H8 and SH Series

SuperH™ RISC Engine

Linkage Editor

User Manual

HITACHI ®

Preface

This manual explains how to use the H Series Linkage Editor, version 5 (Ver.5).

The nine sections and one appendix of this manual cover the following subject matter:

Section 1	Overview
Section 2	Linkage Editor Functions
Section 3	Executing the Linkage Editor
Section 4	Linkage Editor Options and Subcommands
Section 5	Input to the Linkage Editor
Section 6	Output from the Linkage Editor
Section 7	Error Messages
Section 8	Restrictions
Section 9	Object Format Conversion
Appendix A	Example of Use of Linkage Editor

Installation of the Linkage Editor is covered in the Installation Guide supplied with the Linkage Editor.

HITACHI

Users are encouraged to consult the user's manuals for other programs in the H Series cross-software system. Revelant manuals include:

- H Series Librarian User's Manual
- H8/300 Series Cross Assembler User's Manual
- H8/500 Series Cross Assembler User's Manual
- H32 Series Cross Assembler User's Manual
- SH Series Cross Assembler User's Manual
- H8/300 Series C Compiler User's Manual
- H8/500 Series C Compiler User's Manual
- H32 C Compiler User's Manual
- SH Series C Compiler User's Manual

HITACHI

Notes:

- The following symbols have special meaning in this manual

  <item>: <specification item>

  { }: One of the items between brackets is to be selected.

  [ ]: The enclosed item is optional (i.e. can be omitted).

  ...: The preceding item can be repeated.

  Δ: Blanks space(s) or tab(s)

  RET: Press the return (Enter) key.

UNIX is an operating system administrated by the UNIX System Laboratories (United States).

MS-DOS is an operating system administrated by the Microsoft Corporation (United States).

4

HITACHI

Contents

HITACHI

HITACHI

Section 1 Overview

The growing need for large-scale, complex microcomputer programs has led to the common practice of developing a program in separate parts and using a high-level language. In generating a program in this fashion, a compiler or an assembler is used to convert source programs into object modules. After that, a linkage editor is employed to link and edit the modules into one load module file.

The H Series Linkage Editor inputs object module files output by an assembler or C compiler, links and edits them, and generates a single load module file.

Figure 1-1 illustrates the program development procedure using the Linkage Editor.
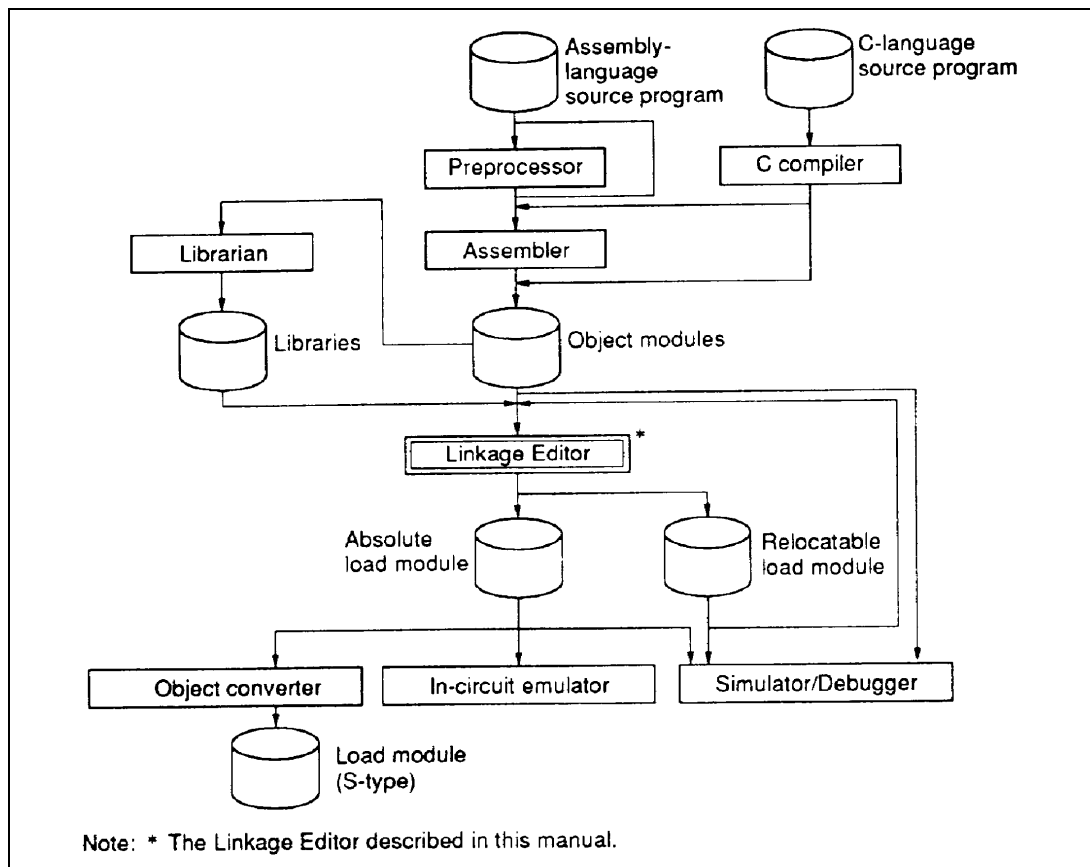
Figure 1-1 Program Development Procedure

HITACHI

.The linkage editor has the following features:

(1) Linkage can be executed by command-line specifications or by subcommands. These two methods allow flexible control over the Linkage Editor to match the desired application.
(2) The load module file output by the Linkage Editor can be re-input and re-edited to generate a new load module file.
(3) Data used by a simulator/debugger or in-circuit emulator in symbolic debugging can be specified for inclusion in the load module file.

1.1 Linkage Editor Functions

The Linkage Editor provides the following five basic functions.

Module Linkage: The module linkage function links and edits object modules output by a compiler or assembler.

Address Resolution: The address resolution function determines absolute addresses for external reference symbols so that references can be made between modules. It also determines absolute addresses for relative addresses.

Load Module File Re-input: The re-input function enables a load module file output by the Linkage Editor to be input again.

Multilinkage: The multilinkage function enables the linkage process to be carried out multiple times during one execution of the Linkage Editor.

Debugging Support: The debugging support function allows display of interim linkage results and provisional correction of errors.

1.2 Object Module and Load Module

An object module is output as a result of compiling or assembling a source program. A load module is obtained by using the Linkage Editor to link object modules.

There are two load module formats: absolute and relocatable. An absolute load module has been assigned absolute addresses, and is in executable form. It does not contain relocation information for relinking and relocation. A relocatable load module has been assigned relative addresses and contains relocation information. This information enables the relocatable load module to be re-input into the Linkage Editor for relinking and relocation. The load module format is selected by the FORM option or subcommand. For details on the FORM option and subcommand, refer to section 4.5.3, "FORM -- Specifies Output Load Module File Format."

8

HITACHI

Object modules, absolute load modules, and relocatable load modules are collectively referred to as modules in this manual.

Modules are either page type or non-page type, depending on the H series microcomputer. The two types differ as to the method of assigning addresses when modules are linked. H8/500 Series modules are page type, whereas H8/300 Series, H32 Series, and SH Series modules are non-page type. When the Linkage Editor is used to link modules, page type and non-page type modules may not be linked together.

1.3 Unit and Section

A unit in a module refers to a compile unit or assembly unit. An object module output by a compiler or assembler consists of a single unit. A load module which represents multiple object modules that have been linked by the Linkage Editor contains more than one unit.

A unit is divided into sections. The Linkage Editor processes one section at a time.

The interrelation among module, unit, and section is illustrated in figure 1-2.
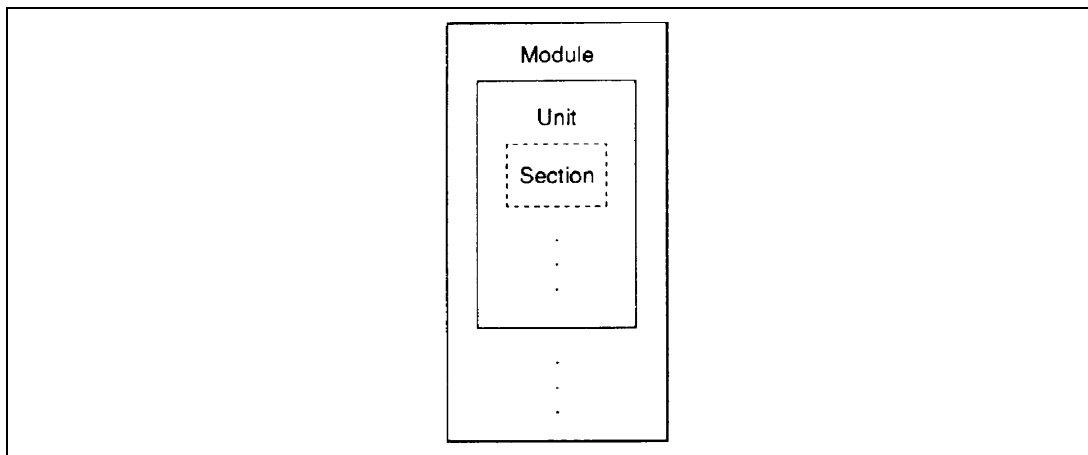
Figure 1-2 Interrelation among Module, Unit, and Section

A section has a name for identification, an attribute describing its content and usage, and a format: either absolute or relocatable. Even if two sections have the same name, they are treated as separate sections when their attributes or formats are different.

Section attributes and formats are classified as follows.

HITACHI

(1) Attributes

   — Code: An area containing instructions or constants.
   — Data: A variable area with values that are changed by the program.
   — Stack: A stack or work area which cannot be initialized.
   — Common: A variable area used in common by multiple modules.
   — Dummy: Used, for example, to define the structure of a variable area; does not generate
     any actual object code.

(2) Formats

   — Absolute: A section in which absolute addresses have already been assigned.
   — Relocatable: A section in which absolute addresses have not yet been assigned.

HITACHI

Section 2 Linkage Editor Functions

This section gives a more detailed description of the basic functions provided by the Linkage Editor. The following discussion and examples will make reference to various options and subcommands used to control the Linkage Editor. Additional information on these options and subcommands can be found in section 3, "Executing the Linkage Editor," and section 4, "Linkage Editor Options and Subcommands."

2.1 Module Linkage

The Linkage Editor reads modules from specified input files and links these modules to generate one load module. Modules are linked by each section, a section being the smallest complete part making up a module.

2.1.1 Section Linkage

A section is linked only if it is relocatable. Since absolute sections have already been assigned absolute addresses, no further linking is performed. Relocatable sections are linked according to the procedure described below.

(1) Grouping of sections with the same name

Sections having the same name but found in more than one unit are grouped.
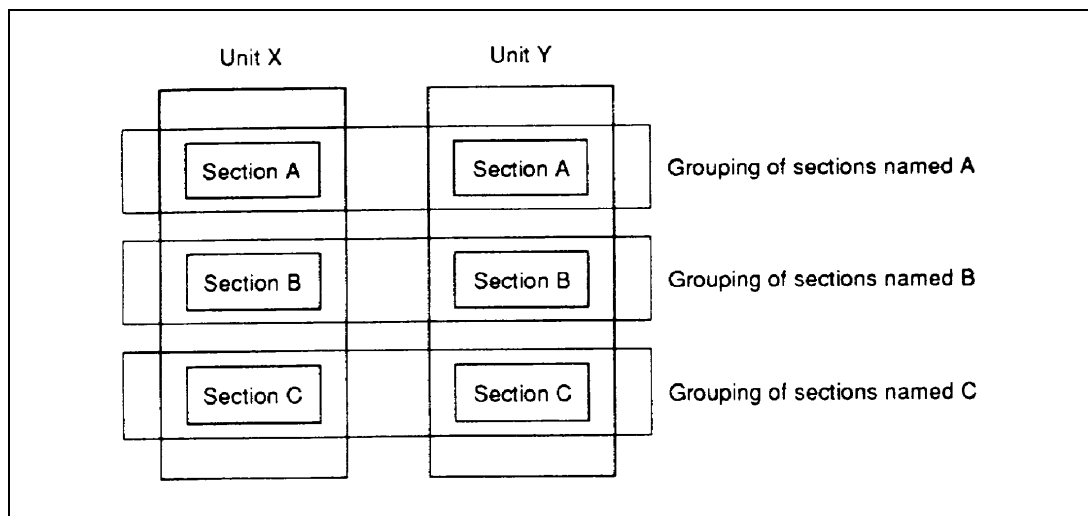
Figure 2-1 Grouping Sections Having the Same Name

HITACHI

A warning message is output when sections have the same name but different attributes. Such sections are then processed as separate sections.

(2) Linking of sections with the same name

Sections having the same name are linked in one of three ways, depending on their attributes.

(a) Simple linkage

Sections with the code, data, or stack attribute and having the same name are allocated consecutively, in the order in which the modules were input.
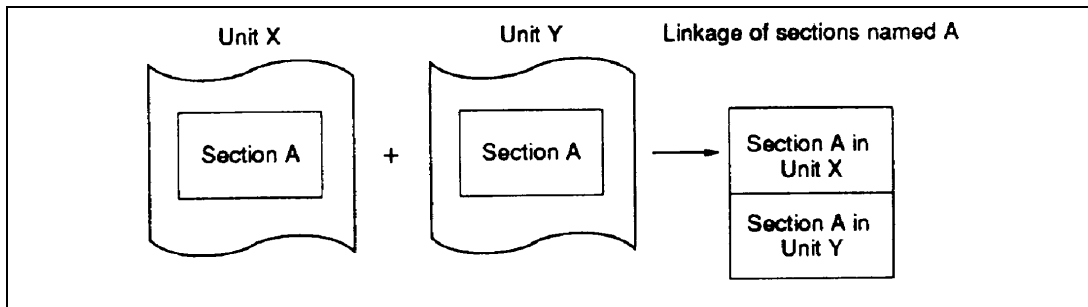
Figure 2-2 Simple Linkage

(b) Common linkage

Sections with the common attribute and having the same name are allocated at the same address. The address area allocated is equal to the size of the largest section.
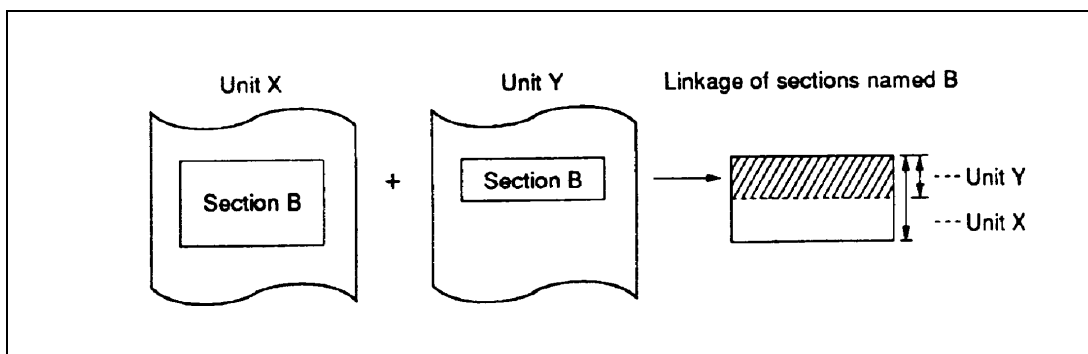
Figure 2-3 Common Linkage

HITACHI

(c) Dummy linkage

Sections with the dummy attribute are not linked, because they do not have any actual existence in the object module file.
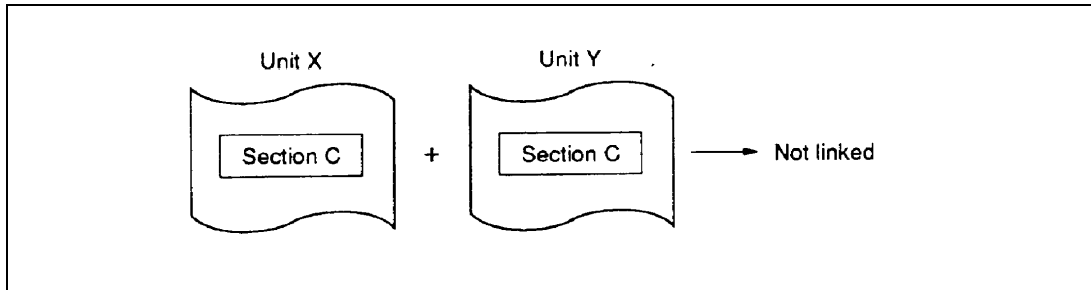
Figure 2-4 Dummy Linkage

(3) Linking of different sections

If a section linking order is specified when the Linkage Editor is executed, sections are linked in that order. If the section linking order is not specified, sections are linked in the order in which they were input.

HITACHI

(a) With a specified linkage order
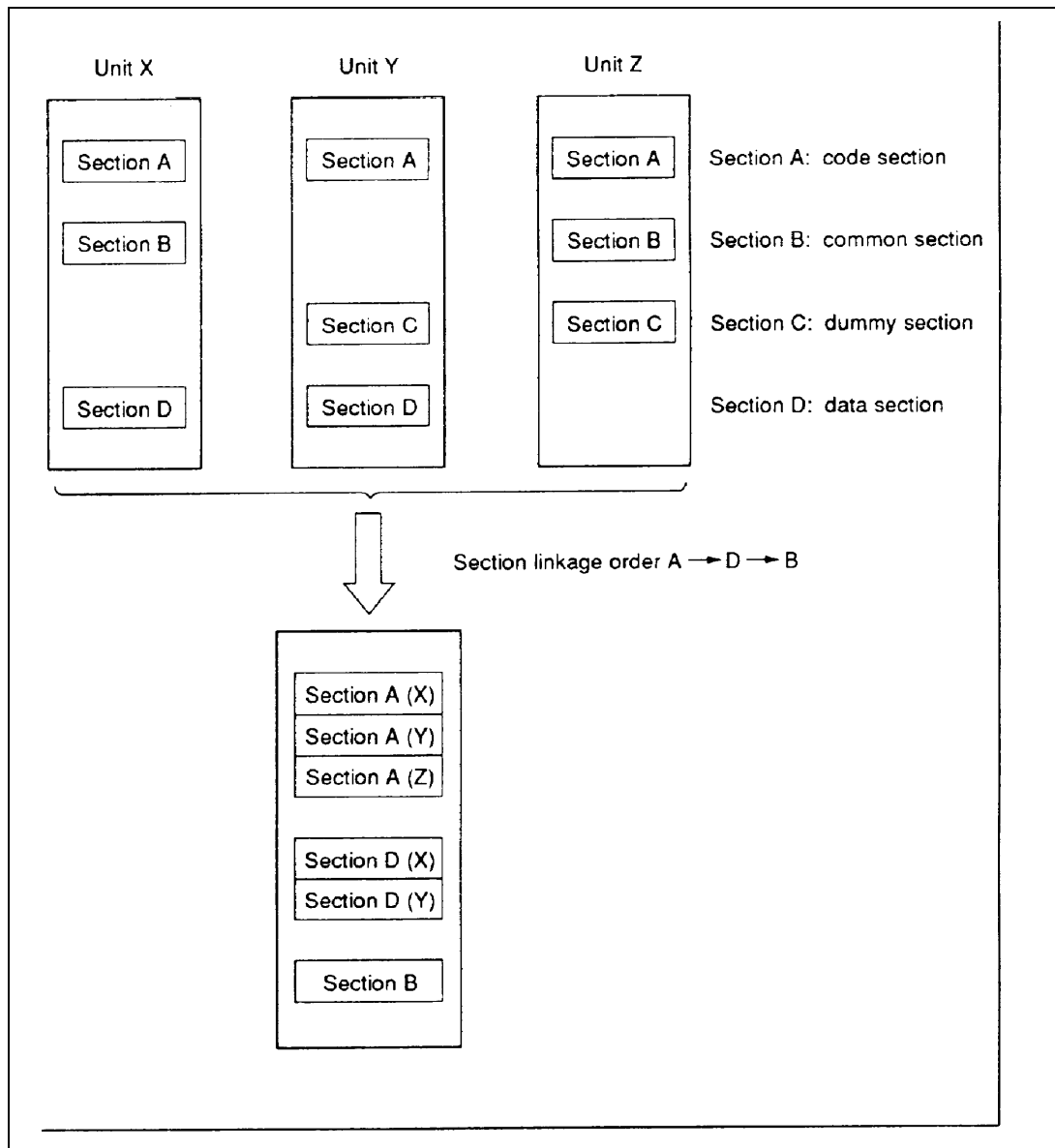
Figure 2-5 Example of Section Linkage with a Specified Linkage Order

HITACHI

The section linkage order can be specified only when the load module output by the Linkage Editor has the absolute format. The linkage order is specified using the START option or subcommand.

(b) Without a specified linkage order
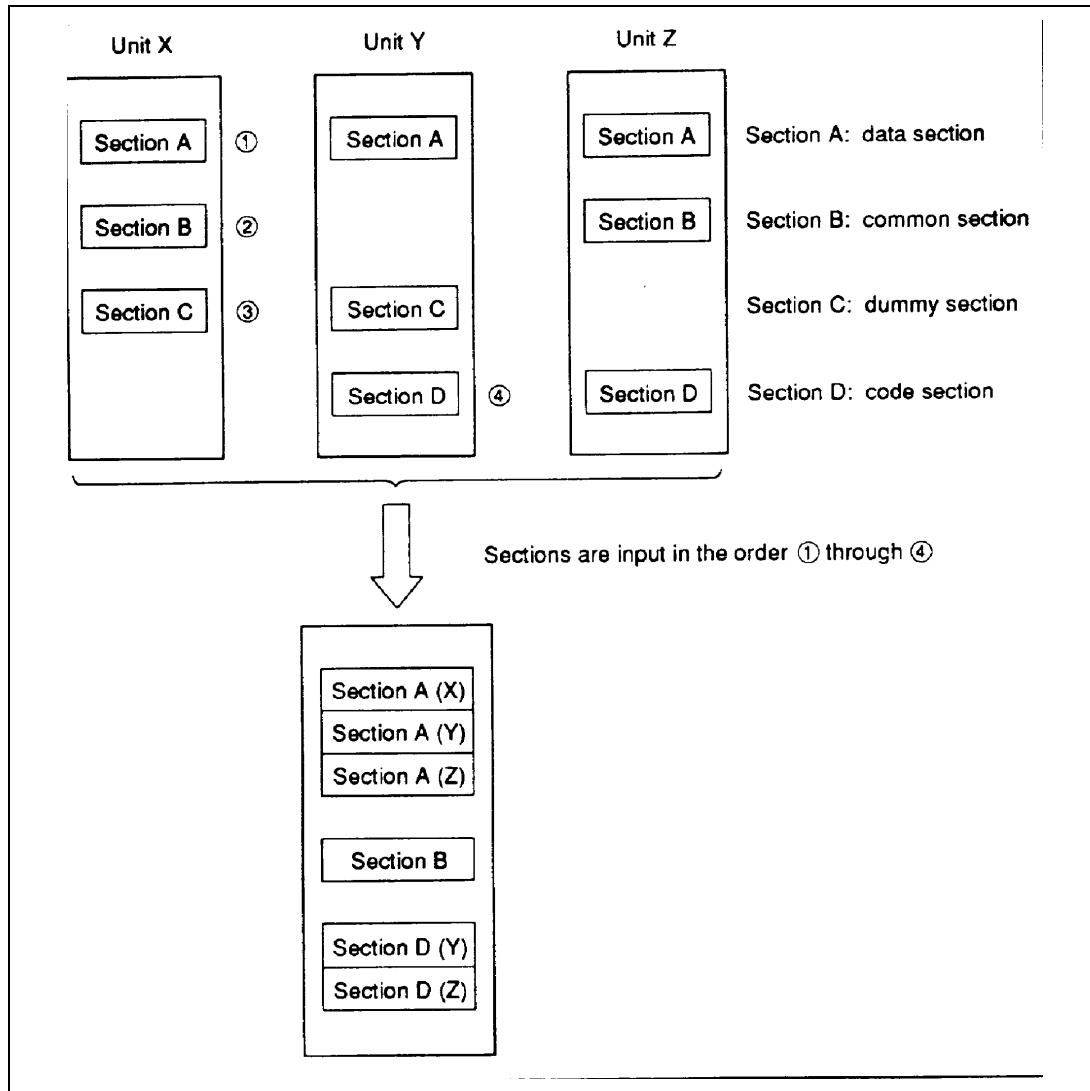
Figure 2-6 Example of Section Linkage without a Specified Linkage Order

HITACHI

Sections having the same name but different attributes are linked in the order in which they are input.
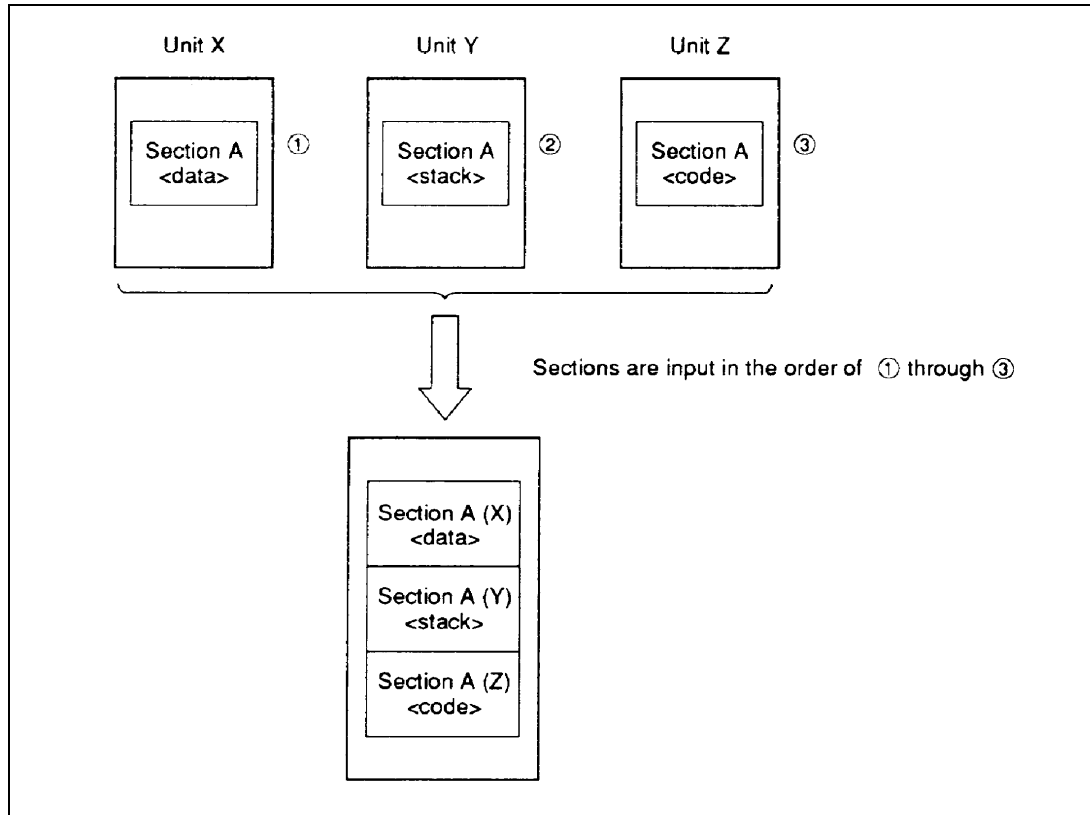
Figure 2-7 Example of Section Linkage for Same Section Name but Different Attributes

(4) Address assignment

Addresses are assigned to each section. Absolute addresses are assigned when the output load module file has the absolute format. The section linkage order and start address can be specified using the START option or subcommand. Absolute addresses are assigned to each section in order, beginning with the start address. If no start address is specified, absolute addresses are assigned beginning from address zero.

If sections with absolute format are linked to sections with relocatable format, the same absolute address may be assigned to more than one section. In that case, the Linkage Editor displays a warning message.

16

HITACHI

When page type modules are linked, if addresses are assigned section by section, one section may overlap a page boundary. In this case the Linkage Editor will display a warning message. However, executing a load module one of whose sections overlaps a page boundary is extremely troublesome. For this reason the Linkage Editor is provided with an autopaging function, which prevents any section in a unit from overlapping the page boundary by allocating the section to the top of the next page. Use of this function is designated by means of the AUTOPAGE option or subcommand. The different methods of assigning addresses to page type modules are shown in figure 2-8 (neither autopaging nor start address specified), figure 2-9 (autopaging specified, start address not specified), and figure 2-10 (autopaging and start address specified).

When the output load module file has the relocatable format, addresses in each section are assigned relative to the beginning of the section. The output format is specified using the FORM option or subcommand.

HITACHI
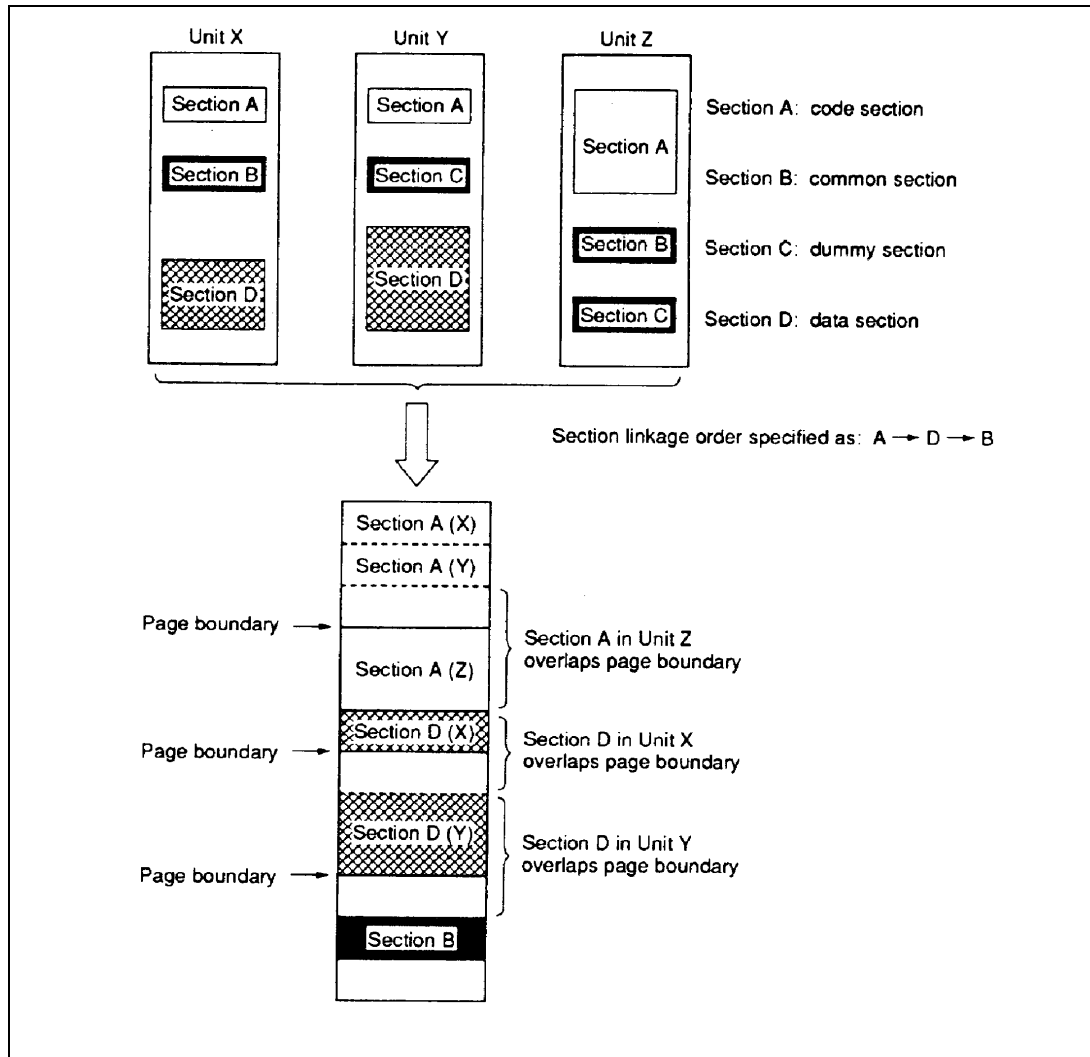
Figure 2-8 Linking of Page Type Modules
(Neither Autopaging nor Start Address Specified)

HITACHI
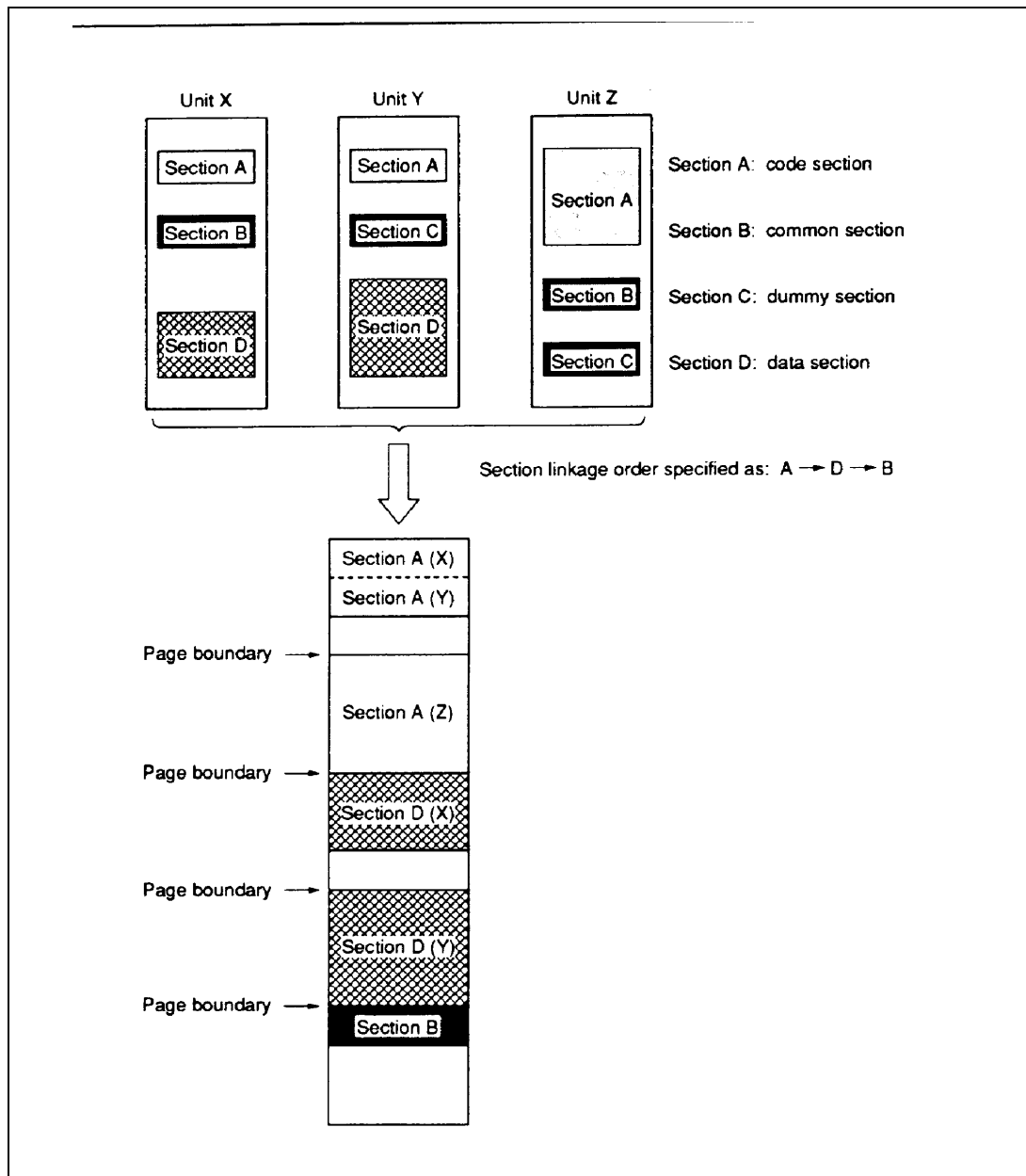
**Figure 2-9 Linking of Page Type Modules
(Autopaging Specified, Start Address Not Specified)**

HITACHI
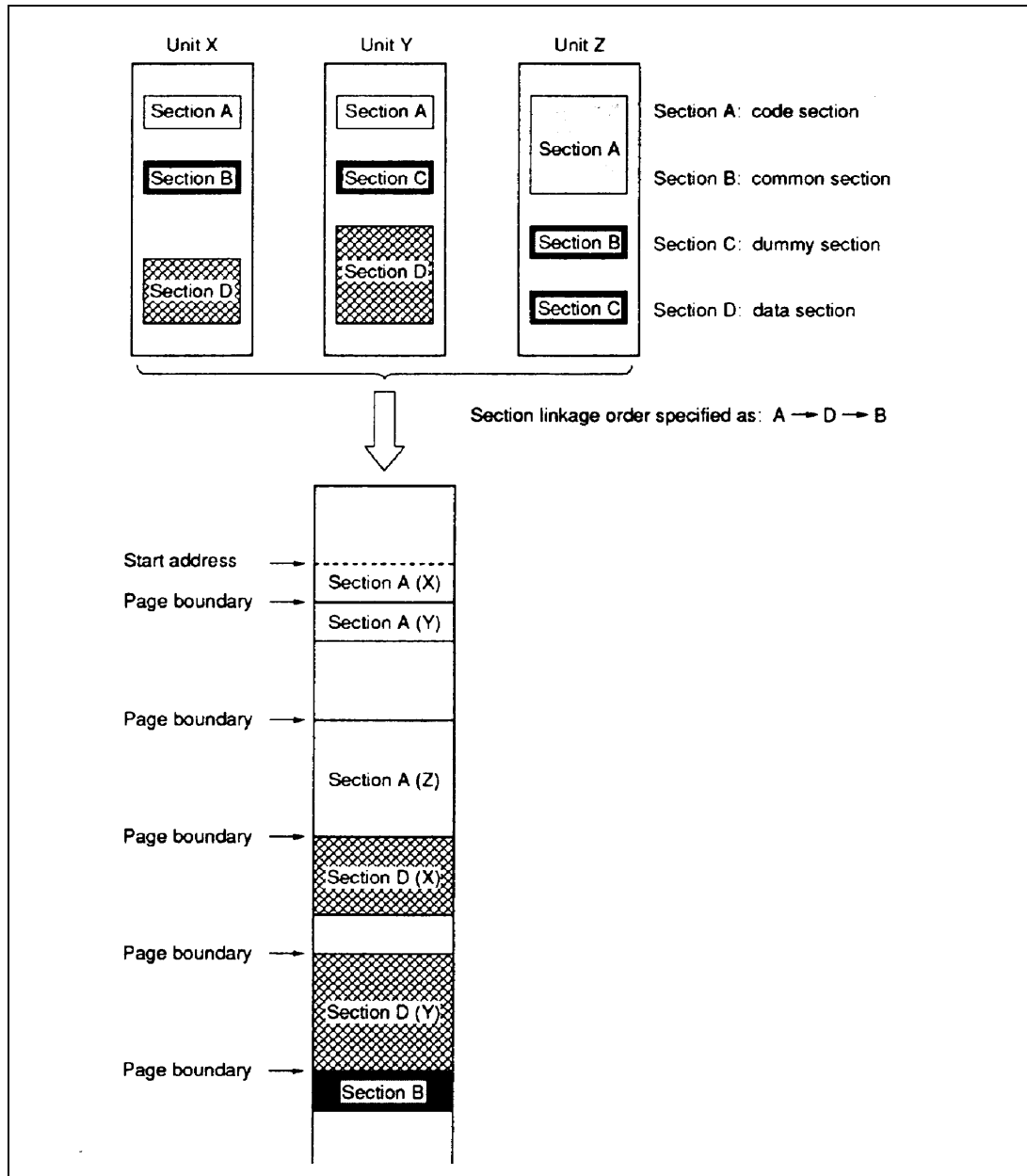
Figure 2-10 Linking of Page Type Modules
(Autopaging and Start Address Specified)

HITACHI

2.1.2 Inclusion from Library Files

The Linkage Editor can link object modules and relocatable load modules input from library files created with the H Series Librarian, and include these modules in the output load module. Inclusion from library files can be accomplished in either of the following two ways.

(1) **Inclusion by Specifying the Module Name:** Particular modules in a library file can be included by specifying the library file name and module name when input file names are specified. Input file names are specified on the command line or by the INPUT subcommand.

(2) **Automatic Inclusion:** After all specified modules have been input, the Linkage Editor begins resolving external reference symbols (after this, external reference symbol is called "import symbol"). If an import symbol is not defined in any of the modules, the Linkage Editor searches the specified library files. If it finds a module defining the unresolved import symbol, the Linkage Editor automatically inputs and links this module. If the unresolved import symbol is not defined in any of these library files, the Linkage Editor searches one or more default library files defined in advance by the user. Again, if it finds a module defining the unresolved import symbol, the Linkage Editor automatically inputs and links this module.

If no module in the default libraries defines the unresolved import symbol, an undefined import symbol error occurs.

A detailed explanation of default libraries is given in section 5.4, "Default Library Files."

Library files are classified into system library files and user library files. The Linkage Editor first searches user library files. When modules containing externally defined symbols (after this, externally defined symbol is called "export symbol") of the same name exist both in a specified system library file and in a user library file, the module in the user library file is linked. The order in which two or more user library files or system library files are searched depends on the order in which they are specified.

For a library file can contain both page type and non-page type modules. If both types of modules are input into the Linkage Editor at the same time, an error will occur. Care must therefore be taken both when creating library files and when specifying them.

Library files are specified using the LIBRARY option or subcommand. On the designation of library files as system files or user files, see the H Series Librarian User's Manual.

An example of the order of module linking when library files are specified is given below.

HITACHI

(1) Object modules a and b are input by the INPUT subcommand.
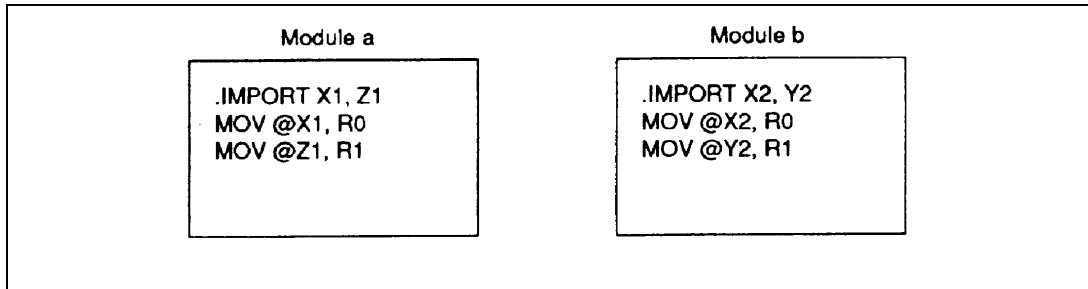
Figure 2-11 Example of Module Linking (Input Object Modules)

(2) Library files lib1, lib2, and lib3 are input in that order by the LIBRARY subcommand.
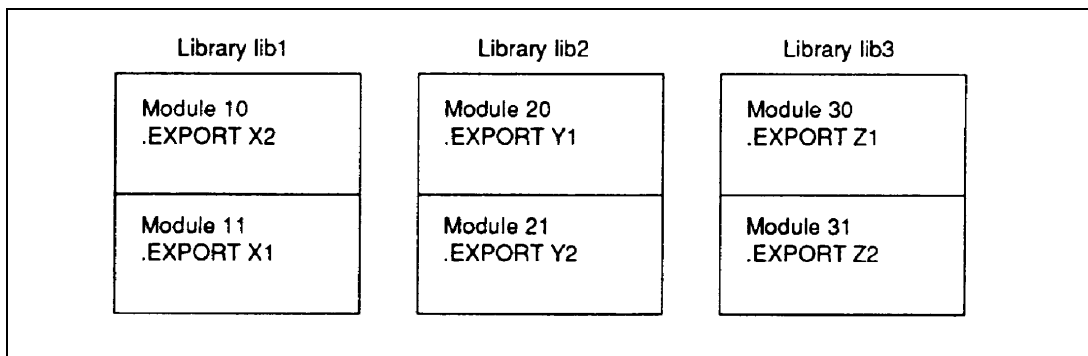
Figure 2-12 Example of Module Linking (Input Library Files)

(3) The Linkage Editor first collects all import symbols declared in the input files, then searches for export symbols in the first specified library. If a symbol is found, the module defining it is linked.

If two or more symbols are declared in separate modules in the same library, the modules are linked in their order of appearance in the library. If a symbol is not found in that library, the next specified library is searched.

In the above example, modules are linked in the following order.

HITACHI

```
┌─────────────┐
│  Module a   │
├─────────────┤
│  Module b   │
├─────────────┤
│  Module 10  │
├─────────────┤
│  Module 11  │
├─────────────┤
│  Module 21  │
├─────────────┤
│  Module 30  │
└─────────────┘
```
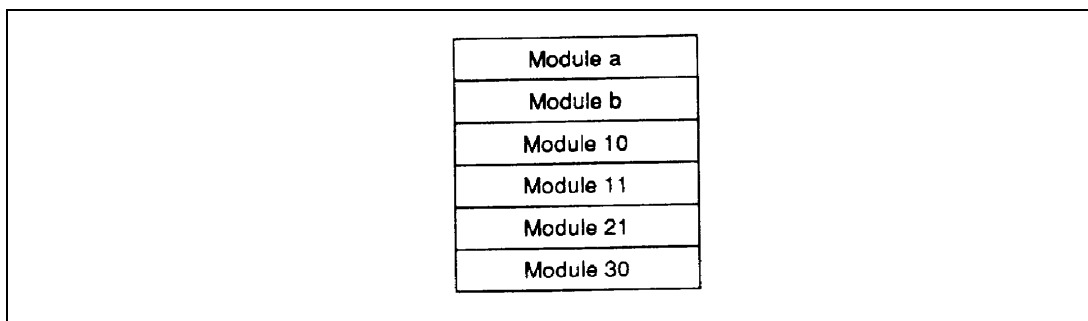
Figure 2-13 Example of Module Linking (Output Load Module)

2.1.3 Exclusion of Module Linking

An option or subcommand can be used to select whether or not to link modules that define non-referenced import symbols. In the following coding example symbol abc is declared as an import symbol, but is not referenced in any executable statement. If exclusion is specified, the module defining symbol abc in a library file will not be linked.

```
.IMPORT xyz, abc
MOV.W @xyz, R0
    .
    .
    .
.END
```
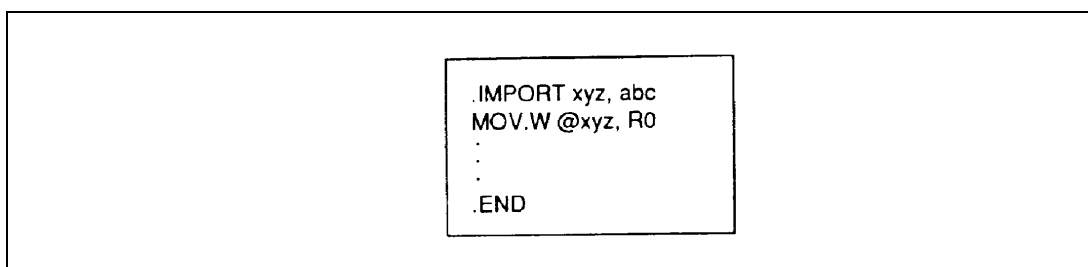
Figure 2-14 Example of Module Containing Non-Referenced Import Symbol

In a C-language program, import symbols are described by an extern declaration, but these symbols are not necessarily referenced. (For example, a large number of non-referenced import symbols are declared in stdio.h.) The exclusion function can reduce program size by excluding unnecessary modules. Exclusion of such modules is specified by the EXCLUDE option or subcommand.

HITACHI

2.2 Address Resolution

When a source program is assembled, the absolute addresses of certain symbols cannot be decided. These include symbols imported from another module and symbols in relocatable sections of the same module. The Linkage Editor determines absolute addresses for these symbols and sets the absolute addresses to the reference positions.

2.2.1 Import Symbol Resolution

When importing symbols from a separate module, the assembler outputs import information in the object program. It also declares export of symbols that can be imported in other modules. As a result, export information is output in the object program. The Linkage Editor relates this import and export information. In addition, it uses address information specified by options or subcommands to determine absolute addresses for the export symbols, and replaces corresponding import symbols with the absolute addresses.

The example given in figure 2-15 illustrates how import symbols are resolved. The modules, sections, and subcommands used in the figure are explained below.

(1) Module a
   — This module consists of one section, section X, having a size of 5000 (hexadecimal) bytes.
   — Symbol S4 in module b is imported at position A1.
   — Symbol S2 in module b is imported at position A2.

(2) Module b
   — This module consists of sections X and Y.
   — The size of section X is 2000 (hexadecimal) bytes.
   — The size of section Y is 3000 (hexadecimal) bytes.
   — S1 is the start of section Y. S2 is located 1000 (hexadecimal) bytes from S1.
   — S3 is the start of section X. S4 is located 1200 (hexadecimal) bytes from S3.

(3) Module c
   — This module consists of one section, section Z, having a size of 4000 (hexadecimal) bytes.
   — Symbol S3 in module b is imported at position C1.
   — Symbol S1 in module b is imported at position C2.

HITACHI

(4) Subcommands

```
INPUTΔa,b,c
STARTΔX,Y,Z(10000)
EXIT
```

Three modules a, b, and c are input to the Linkage Editor. Sections are linked in the order X, Y, Z. The start address is 10000 (hexadecimal).
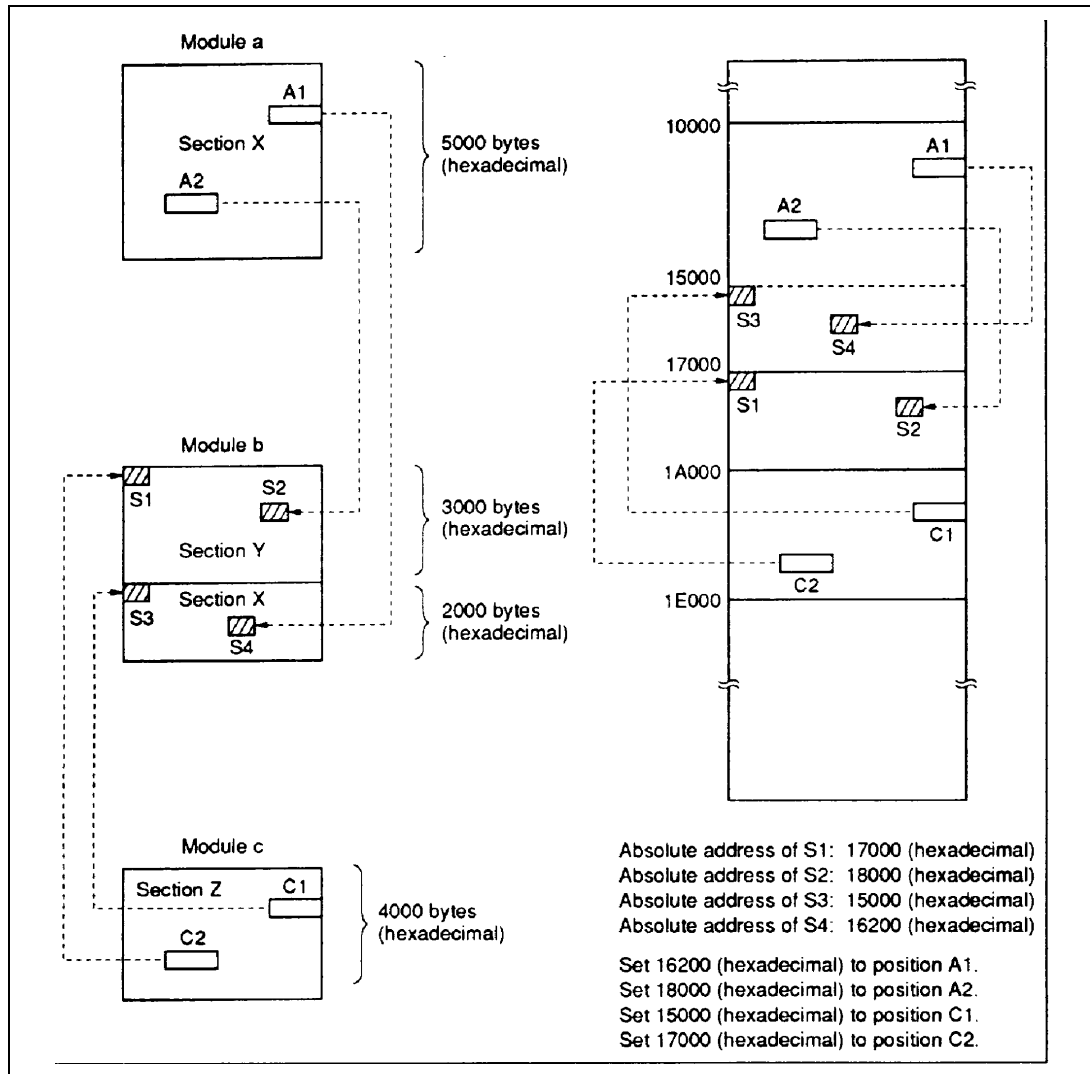
Figure 2-15 Resolution of Import Symbols

HITACHI

2.2.2 Address Resolution within a Module

When a symbol defined in a relocatable section of a module is referenced within the same module, the assembler expresses the symbol address as a relative address from the start of the section. The Linkage Editor uses this relative address value and address information specified by options or subcommands to decide the absolute address. It then replaces the relative address with the absolute addresses.

The example given in figure 2-16 illustrates the resolution of addresses within a module. The modules, sections, and subcommands used in the figure are explained below.

(1) Module a
— This module consists of one section, section X, having a size of 5000 (hexadecimal) bytes.

(2) Module b
— This module consists of sections X, Y, and Z.
— The size of the section X is 6000 (hexadecimal) bytes.
— The size of the section Y is 1000 (hexadecimal) bytes.
— The size of the section Z is 2000 (hexadecimal) bytes.
— B1 references S1.
— B2 references S3.
— B3 references S2.
— S1 is located 3000 (hexadecimal) bytes from the start of section X.
— S2 is located 4500 (hexadecimal) bytes from the start of section X.
— S3 is located 5000 (hexadecimal) bytes from the start of section X.

(3) Subcommands
```
INPUTΔa, b
STARTΔX, Y, Z(10000)
EXIT
```

26

HITACHI

Two modules a and b are input to the Linkage Editor. Sections are linked in the order X, Y, Z. The start address is 10000 (hexadecimal).

Logical address of S1: 18000 (hexadecimal)
Logical address of S2: 19500 (hexadecimal)
Logical address of S3: 1A000 (hexadecimal)

Set 18000 (hexadecimal) to position B1.
Set 1A000 (hexadecimal) to position B2.
Set 19500 (hexadecimal) to position B3.
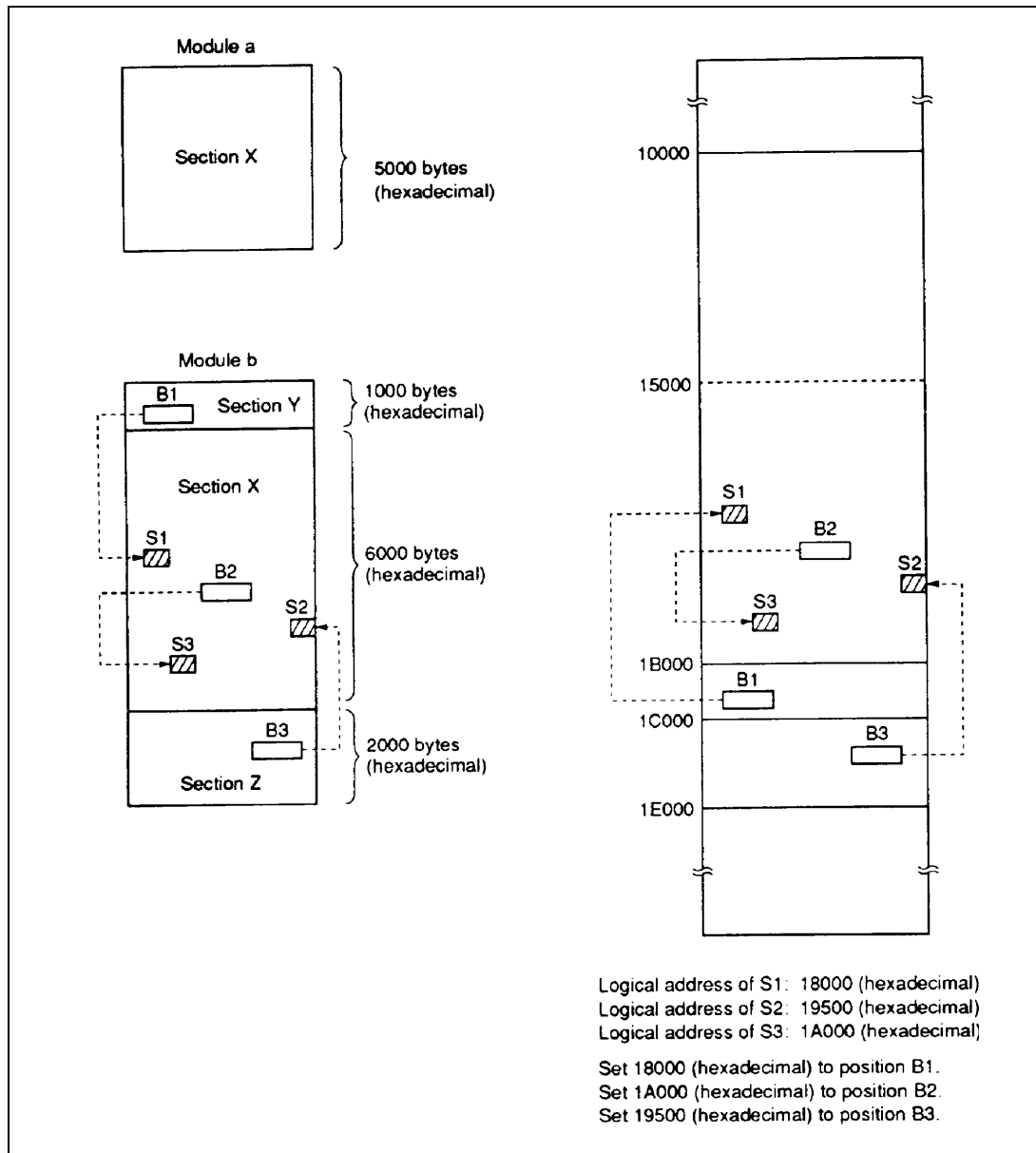
Figure 2-16 Address Resolution within a Module

HITACHI

2.2.3 Suppressing the Listing of Unresolved Symbols

For a relocatable load module, the display of unresolved symbol names can be suppressed. This can be selected by the UDF option or subcommand.

2.3 Load Module File Re-Input

Load module files have to be recreated using the Linkage Editor when a program has been modified or import symbols remain unresolved. The re-input function eliminates the need to specify each object module separately. By simply specifying the existing load module file and the object module files that were modified (or the object module files containing the export symbols), this function will recreate the load module file.

If modules are to be replaced, the re-input function carries out the replacement on a unit basis. A detailed explanation of unit replacement is given in section 2.3.1, "Automatic Unit Exchange."

The load module file to be re-input can be specified on the command line or using the INPUT subcommand.

Only load module files in relocatable format can be re-input. The FORM option or subcommand is used to specify the relocatable format when creating a load module file.

An overview of the load module file re-input function is shown in figure 2-17.

HITACHI
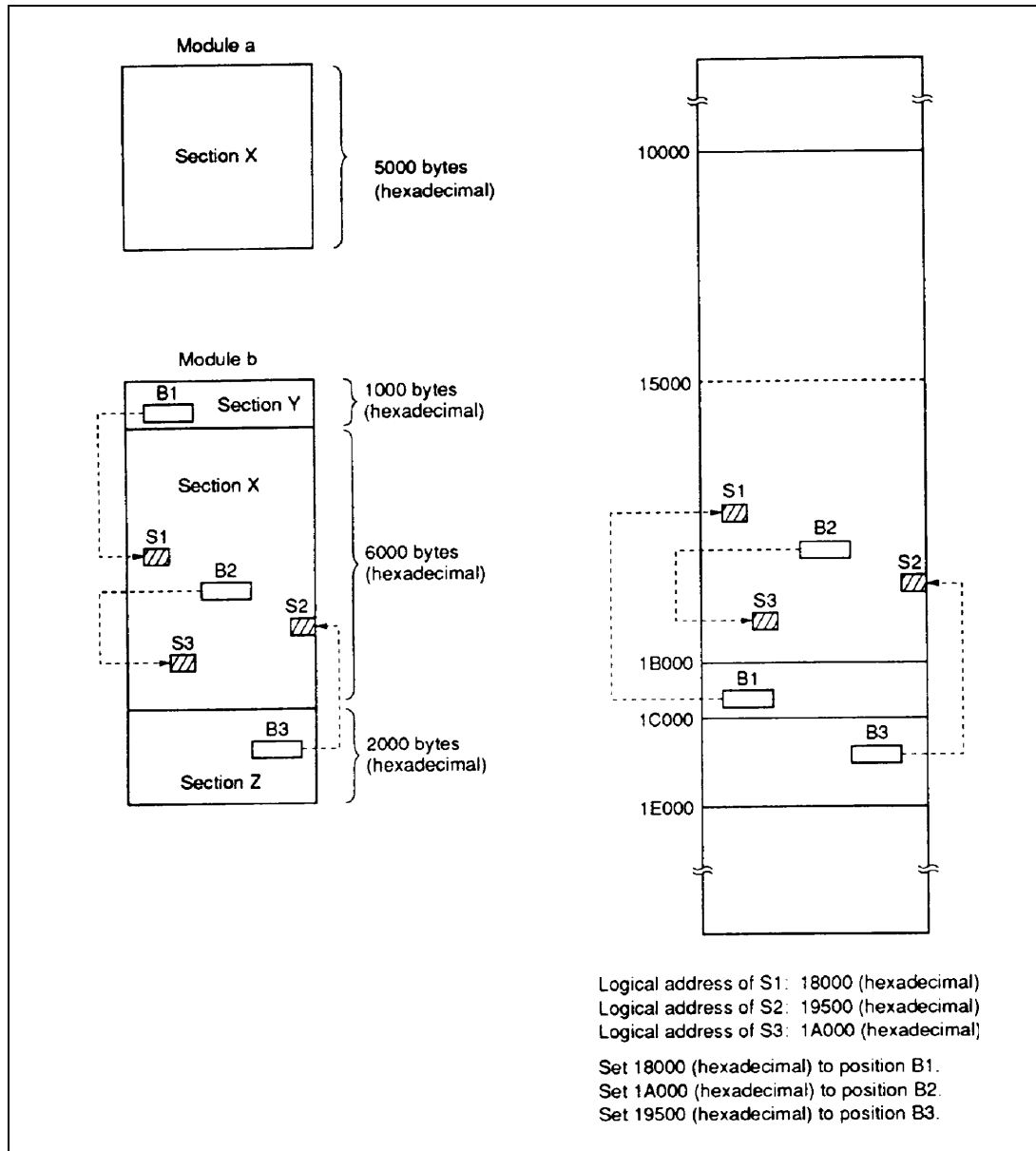
Figure 2-17 Load Module File Re-Input Function

Load module file a and object module files b and c are input to the Linkage Editor, which outputs a new load module file d. Load module file d consists of units U1, U2, U3, U4, U5, and U6.

HITACHI

2.3.1 Automatic Unit Exchange

When the Linkage Editor finds units with the same name in two or more modules, it gives inclusion priority to the unit in the module that was specified first. To replace units in a load module file, first specify files containing the replacement units, then specify the relevant load module file. This will produce the same result as using the EXCHANGE subcommand. This function is called automatic unit exchange.

By using automatic unit exchange, new load module files can be created by simply changing the specified order of file input. This feature is convenient when it is necessary to modify programs frequently, such as during debugging.

An example of the procedure for automatic unit exchange is shown in figure 2-18.

HITACHI
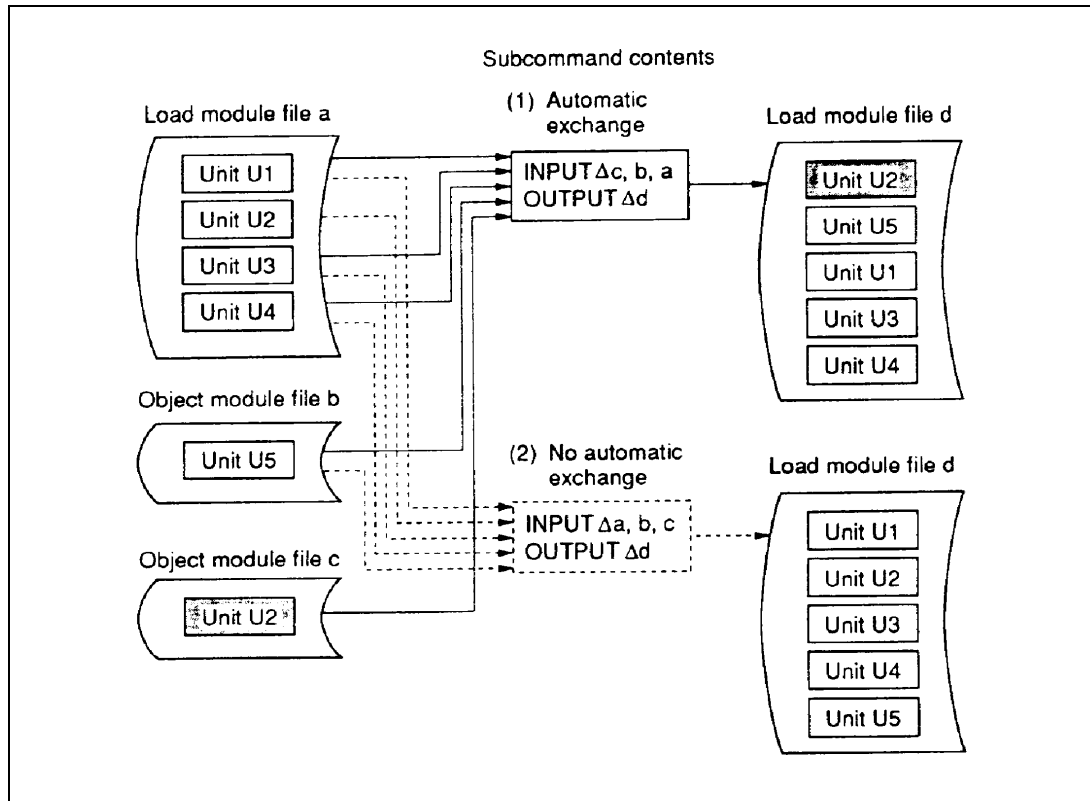
Figure 2-18 Automatic Unit Exchange

(1) Automatic Exchange: Object module files c and b and load module file a are input in that order. Unit U2 in load module file a is not included by the Linkage Editor since unit U2 in load module file c has already been input.

(2) No Automatic Exchange: Load module file a and object module files b and c are input in that order. Unit U2 in load module file c is not included by the Linkage Editor since unit U2 in load module file a has already been input.

HITACHI

2.3.2 Forced Unit Exchange

In addition to using automatic unit exchange, the EXCHANGE subcommand can also be employed to specify the units to be replaced. This function is called forced unit exchange.

By specifying the following subcommands, the result of forced unit exchange will be the same as that of the automatic unit exchange shown in figure 2-18.
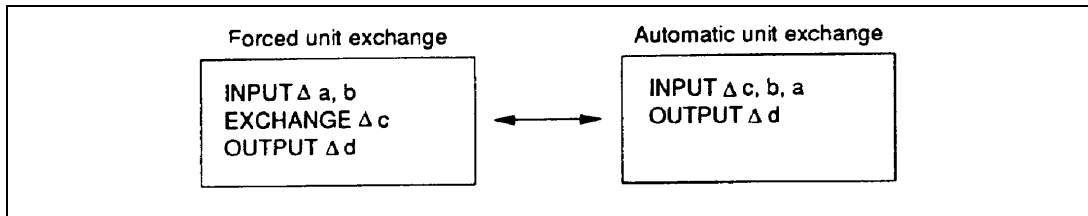
In this example of forced unit exchange, the Linkage Editor inputs units U1, U2, U3, and U4 in load module file a and unit U5 in object module file b, then forcibly replaces the unit U2 already input with unit U2 in object module file c. Load module file d output by the Linkage Editor contains units U1, U3, and U4 from file a, unit U5 from file b, and unit U2 from file c. Thus load module file d has the same unit configuration as load module file d shown in the example of automatic file exchange in figure 2-18.

2.4 Multilinkage

The Linkage Editor can handle up to 256 input files in one link ge process. When there are multiple input files, one way to link them is to re-input the load module file. The multilinkage function allows several linkage processes to be completed with just one execution of the Linkage Editor, instead of executing it separately for each linkage process.

The END subcommand indicates the end of one linkage process when multilinkage is performed. The end of the final linkage process, however, is specified by the EXIT subcommand.

An example of a multilinkage operation is shown in figure 2-19.

HITACHI

Note: When the default library is used during multi-linkage process, the modules in the default library are linked in the first linkage process. When the modules must be linked in the final linkage process, specify the NOLIBRARY command in the processes except the final process.
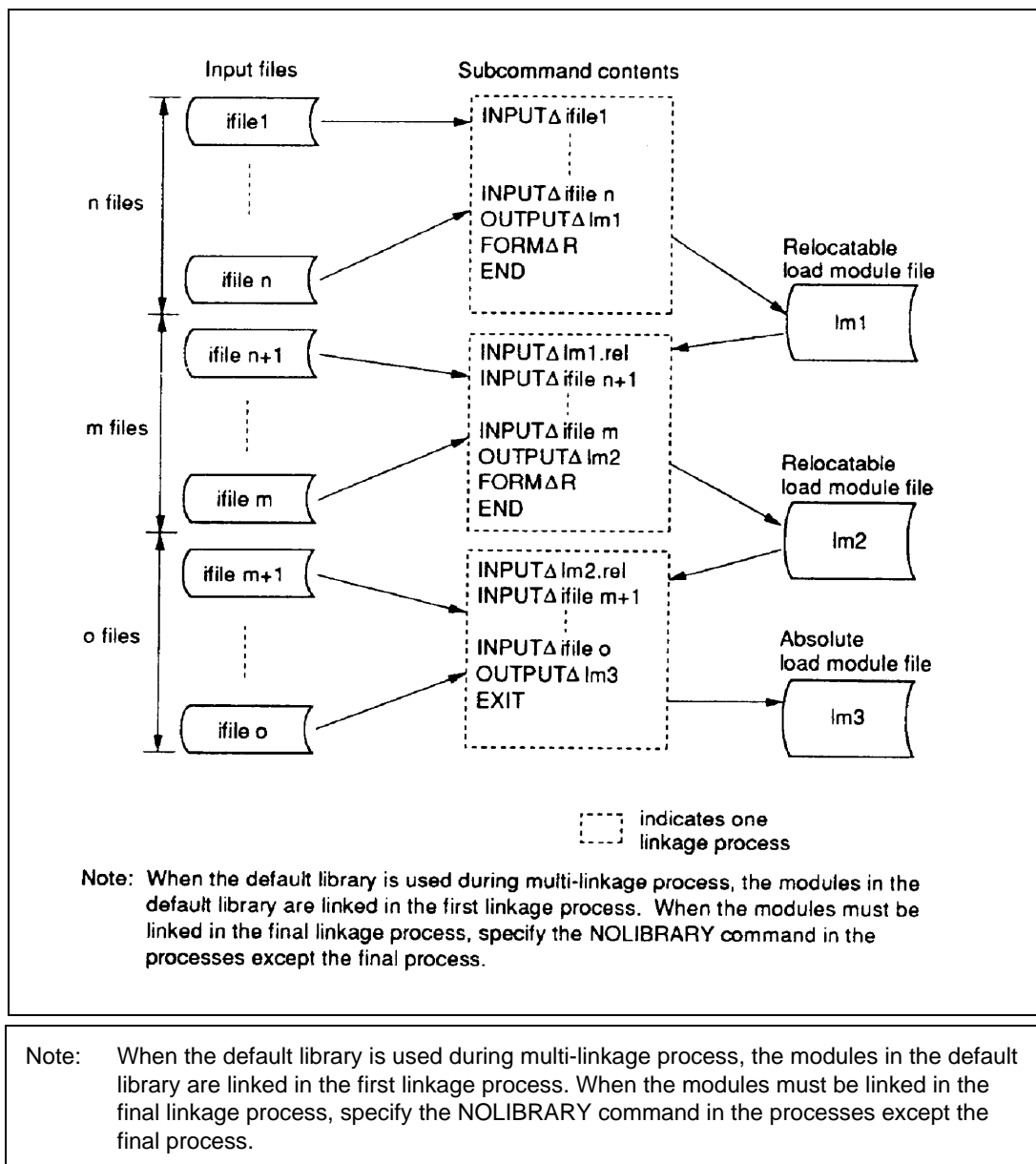
Figure 2-19 Multilinkage Function

2.5 Debugging Support

Debugging support functions are used to confirm the interim linkage results at the program debugging stage and to make provisional recovery from errors in load module files. Debugging support functions include displaying interim linkage information as well as defining, changing, and deleting export and import symbol names. A brief explanation of each function is given below.

(1) Display of Interim Linkage Information: This function is used during subcommand input when it is desired to see information about the load module being processed by the Linkage Editor. Specifying the LIST subcommand outputs interim linkage information to the standard output device.

Three types of linkage information are displayed.

(a) Linkage map

(b) Unresolved import symbols

(c) Export symbols

(2) Change and Deletion of Unit Names, Export Symbol Names, and Import Symbol Names: These functions make it possible to change or delete any duplicated names of units, export symbols, and import symbols. Noted that names of import symbols cannot be deleted.

Names are changed by the RENAME subcommand and are deleted by the DELETE subcommand.

(3) Forced Definition of Import Symbols: This function is used to define provisional values for import symbols. The values defined with this function are valid only for the linkage operation being processed.

The forced definition of these symbol values is specified using the DEFINE option or subcommand.

HITACHI

2.6 Address Check

When an absolute load module is created with the Linkage Editor, addresses must be assigned to sections in accordance with the target CPU memory map. If not, the load module cannot be loaded to memory.

The address check function provided with the Linkage Editor confirms the validity of section address assignments on the basis of CPU memory map information (hereinafter called "CPU information"). This CPU information is read from a specified file.

To execute an address check, the CPU option or subcommand is used to specify the CPU information file. The CPU information file is created using the CPU information analysis program (CIA) included in the H8/300 series Simulator/Debugger. Note that the CPU information analysis program is not available for CPUs other than the H8/300 and SH series; thus the address check function can be used only with this series.

Regarding the method of creating a CPU information file, refer to the H8/300 Series or SH Series Simulator/Debugger User's Manual or the SH Series Simulator/Debugger User's Manual.

2.7 Support of Storing Program in ROM

When a user program is coded in C language and the load module is to be stored in ROM, data sections having initial value (D sections) will also be stored in ROM. To assist the user, the Linkage Editor carry out the following operations.

HITACHI

(1) An area of the same size as the D section (called the D' section) is reserved in the RAM area of the output load module. The memory map of the load module looks like this:
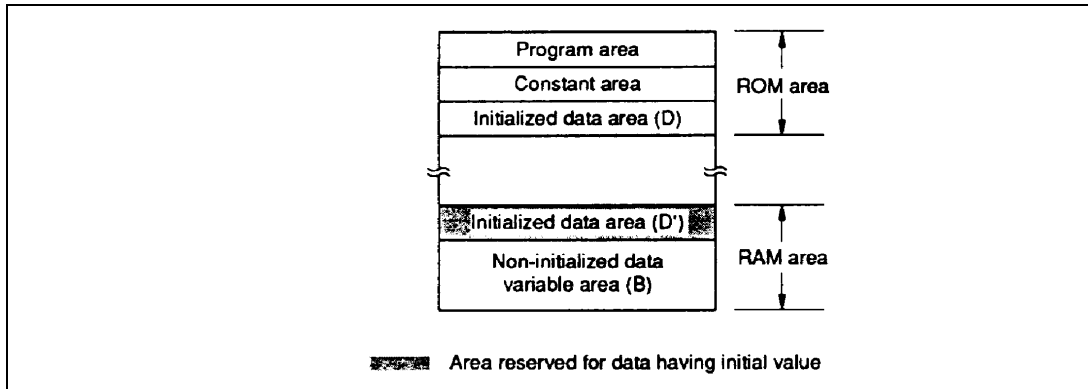
Figure 2-20 Memory Map for Storing Program in ROM

(2) When a variable declared in the D section is referenced, its address is changed to point to the RAM area. The variable address becomes:

Starting address of D section + relative address within section

The ROM ability support function changes this to:

Starting address of D' section + relative address within section

Example: MOV @a, R0

The address of symbol "a" declared in the D section becomes (x) + (y) as shown below. This address is also stored on the object code.
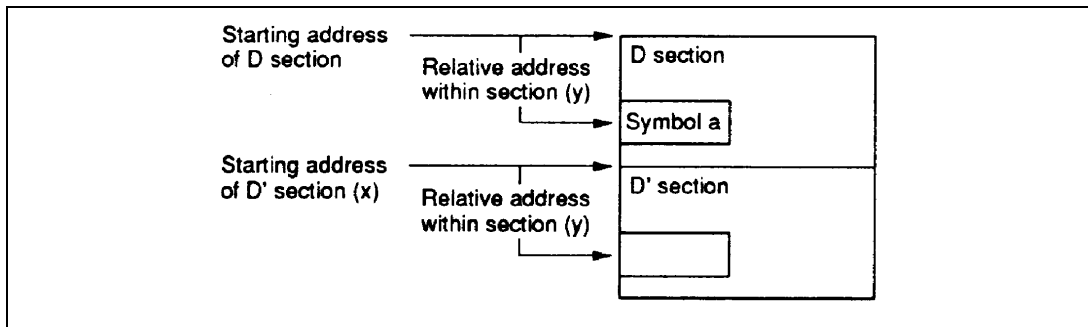
Figure 2-21 Symbol Address for Storing Program in ROM

HITACHI

(3) Data is copied from ROM to RAM in the start-up routine.

The copy process is included in the start-up routine. The procedure for including this process is fcdescribed in the C Compiler User's Manual.

HITACHI

Section 3 Executing the Linkage Editor

To execute the Linkage Editor, first starts Linkage Editor by entering a command line. This command line specifies the names of files to be input, and also specifies options giving various instructions to the Linkage Editor. If these instructions are sufficient, the Linkage Editor can be executed using the command line alone. If further instruction are needed, they can be given in subcommands.

Command Line Specification: This method executes linkage simply by specifying the input files and options on the command line. It is used when only a few files are to be input and the linkage operation is relatively straightforward.

Specifying Subcommands: This method, in addition to a command line, uses subcommands to control the Linkage Editor. The subcommands specify files to be input and output, and execution control parameters for the Linkage Editor. This method is used when a large number of files or modules are specified, when the order in which sections are to be linked specified, or when multilinkage function is used. There are two ways of specifying subcommands: One is direct input from the keyboard or other input device in interactive mode and the other is input from a subcommand file.

UNIX System: File names are specified in the following format:

Path name   main file name   file type
①   ②   ③

①   Path name
Specify the directory path of the directory containing the file, using slashes (/) to delimit directory names. The default value is the current directory.

②   Main file name
Specify the name of the file.

③   File type
Specify the type of file separated from the main file name by a period (.).

The general rules of file naming for the Linkage Editor conform to the operating-system (OS) rules.

The OS shell (command interpreter) checks the command line before passing control to the Linkage Editor. Use characters that the OS allows on the command line.

HITACHI

Example: /usr/tool/prog.typ
                    ↑   ↑ ↑————— File type
                    | |————————— Main file name
                    |————————————————— Path name

HITACHI
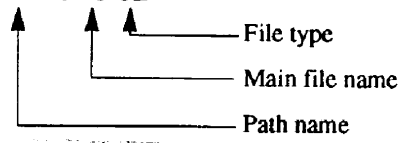
PC System: Before using this Linkage Editor, set the MS-DOS configuration file (CONFIG.SYS) with the editor as follows.

FILES=20

SHELL=a:¥command.com a:¥ /p

---

① The number of files that is allowed to open at one time during Linkage Editor operation.

② Directory path specification that is required when COMMAND.COM is re-loaded.

If both the input file and the output file have the same name, the input file data will be damaged. Take care that the input and output files do not have the same name.

3.1 Command Line Format

The following format is used for the Linkage Editor command line.

UNIX System:

```
lnkΔ [<input file name>[,<input file name>]...]
[[Δ]-<option name>[[Δ]-<option name>...]]RET
```

PC System:

```
LNK[<input file name> [(<module name>[,<module name>...])]
[,<input file name>[(<module name>[,<module name>...])]]...]
[[Δ]/<option name> [[Δ]/<option name>...]]RET
```

(1) Command Name: "lnk" is input to start up the Linkage Editor.

(2) Input File Names: Names of files to be input in the Linkage Editor are specified. These may be object module files or relocatable load module files. When more than one file is specified, the names are delimited by a comma (,).

If the file type is not specified with the input file name, the Linkage Editor automatically assumes that the type is ".obj." In MS-DOS system, up to 256 files can be input in one linkage process. However the maximum number of characters that can be input in a line is 255, and therefore 256 files can not be specified in one line. When many files must be specified, use the subcommand.

(3) Module Name (Only for PC System): When the library file is specified as the input file, specify the module name. The module name must be preceded by the library file name and

40

HITACHI

must be specified in the parentheses. If the input file is the object module file or the load module file, do not specify the module name.

HITACHI

(4) Option Names: <u>Each option name must be preceded by a hyphen (-).</u> When an option name follows an input file name or another option name, one or more spaces or tabs may be inserted to delimit the names, or they may be entered continuously. Option names are described in detail in section 4, "Linkage Editor Options and Subcommands."

(5) Specifying the Mode of Execution: Command line specification determines whether linkage is to be executed by the command line only or subcommands are to be used as well.

    (a) Specifying execution by command line: If one or more input files are specified on the command line and no subcommand file is specified, module linkage will be executed according to the command line only.

    (b) Specifying subcommands: If no input files are specified on the command line, or a subcommand file is specified, control of the Linkage Editor will be controlled by the subcommands.

3.2 Executing by Command Line

In this method, input files are specified on the command line, and the Linkage Editor executes module linkage according to the information specified in the command line alone. Output files and other instructions to the Linkage Editor are specified in the form of options. Command line execution is sufficient for performing linkage operations when the number of input files is small, and when there is no need for detailed instructions to the Linkage Editor such as regarding the order in which sections are to be linked. Examples of execution by command line only are given below. For details on options in these examples, see section 4, "Linkage Editor Options and Subcommands."

EXAMPLE 1 (UNIX system):

```
% lnkΔadd, sub, mul, divΔ-OUTPUT=arithΔ-ENTRY=main RET
```

Four files "add.obj," "sub.obj," "mul.obj," and "div.obj" are to be input to the Linkage Editor. These are to be linked and output as absolute load module file "arith.abs." Exports symbol "main" is specified as the start address for execution of the output load module file. No linkage list is to be output.

EXAMPLE 2 (MS-DOS system):

```
>LNKΔMAIN, KEY, DISPLAY, PRINT/OUTPUT=CALC/PRINT=CALC/FORM=R/DEBUG RET
```

Four files "MAIN.OBJ," "KEY.OBJ," "DISPLAY.OBJ," and "PRINT.OBJ" are to be input to the Linkage Editor. These are to be linked and output as relocatable load module file "CALC.REL." Debugging information is to be incorporated in this load module file. Linkage list "CALC.MAP" is to be output.

HITACHI

3.3 Controlling by Subcommands

When a large number of files or modules must be input, or when complex section linking is to be performed, the command line alone may not be sufficient to contain all the specifications. In such cases, subcommands are used to control the Linkage Editor. Subcommands may be entered one at a time in interactive mode, from the keyboard or other standard input device, or a subcommand file consisting of a group of subcommands may be created in advance, and subcommands may be entered from this subcommand file.

(1) Interactive Mode: Can be used when the number of subcommands is relatively small. This method is also useful when the Linkage Editor is employed during program debugging, where it is desired to check interim linkage results or make provisional recovery from errors.

(2) Subcommand File: A subcommand file is used to control the Linkage Editor when the number of subcommands is large, or the procedures to be carried out are mostly routine. A subcommand file is used by specifying the SUBCOMMAND option on the command line. The name of the subcommand file to be input is specified as a parameter of the SUBCOMMAND option.

The Linkage Editor can use a subcommand file even when subcommands are input interactively. Specify the SUBCOMMAND subcommand with the subcommand file name as a parameter.

Note: In PC system, input all command lines and subcommand lines by capital letters.

HITACHI

3.3.1 Executing in Interactive Mode

In this method, subcommands required for Linkage Editor operations are input directly from the standard input device. Execution proceeds by this method when no input files are specified on the command line and the SUBCOMMAND option is not specified. It is used when the number of subcommands to be input is relatively small, or when it is desired to confirm linkage results while inputting subcommands, as in the first stage of program debugging. When the debugging support function is used, this method of execution is the most suitable.

An example showing input of subcommands in interactive mode is given below. Functions of the subcommands listed here are detailed in section 4, "Linkage Editor Options and Subcommands."

EXAMPLE:

```
%lnk RET...............................①
: INPUTΔmain RET......................②
: INPUTΔsend, receive, exchange RET...③
: INPUTΔaccount RET...................④
: LIBRARYΔsyslib RET.................⑤
: PRINTΔ # RET.......................⑥
: FORMΔR RET.........................⑦
: EXIT RET...........................⑧
```

① Command line, starting up the Linkage Editor in interactive mode.
② Inputs object module file "main.obj."
③ Inputs three object module files "send.obj," "receive.obj," and "exchange.obj."
④ Inputs object module file "account.obj."
⑤ Inputs library file "syslib.lib."
⑥ Outputs linkage list to standard output device.
⑦ Creates a load module in relocatable format.
⑧ Outputs load module file "main.rel" and ends the linkage operation.

HITACHI

3.3.2 Executing from a Subcommand File

In this method, a subcommand file is used which has been created in advance and which contains the subcommands necessary for Linkage Editor operations. This subcommand file is specified as a parameter of the SUBCOMMAND option or subcommand. This method is used when the number of subcommands to be specified is large, or the same linkage process is carried out repeatedly routine. It saves trouble to input subcommands from the keyboard each time.

A subcommand file is created using an editor. An example of executing from a subcommand file is given below. Functions of the subcommands listed here are detailed in section 4, "Linkage Editor Options and Subcommands."

EXAMPLE 1 (UNIX system):

```
%lnkΔ-SUBCOMMAND=prglnk.sub RET............................①
```

Contents of subcommand file "prglnk.sub":

```
OUTPUTΔfunction .........................................②
INPUTΔsin, cos, tan......................................③
INPUTΔasin, acos, atan...................................④
INPUTΔhsin, hcos, htan...................................⑤
INPUTΔlog, log10.........................................⑥
FORMΔA...................................................⑦
EXIT.....................................................⑧
```

① Command line, starting up the Linkage Editor and entering subcommands from subcommand file "prglnk.sub."
② Names the output file as "function." Either ".rel" or ".abs" is assumed, because the file type is omitted.
③ Inputs object module files "sin.obj," "cos.obj," and "tan.obj."
④ Inputs object module files "asin.obj," "acos.obj," and "atan.obj."
⑤ Inputs object module files "hsin.obj," "hcos.obj," and "htan.obj."
⑥ Inputs object module files "log.obj" and "log10.obj."
⑦ Creates a load module in absolute format. The file type for the output file name becomes ".abs."
⑧ Outputs load module file "function.abs" and ends the linkage operation.

HITACHI

EXAMPLE 2:

```
%Ink RET............................①
:SUBCOMMAND pgmlnk.sub RET...........②
```

① Command line, starting up the Linkage Editor. Module linkage is executed interactively, because no parameters are specified.

② Inputs subcommands from "pgmlnk.sub."

<u>If there is no EXIT subcommand in the subcommand file, the Linkage Editor waits for further subcommand input.</u>

3.4 Linkage Editor Termination

When terminated, the Linkage Editor returns an error level to the system as a return code. Execution of a command file can be controlled by this return code.

The return code has the following values, depending on the error level.

Normal termination   0

Warning   0

Error   1

Fatal error   1

HITACHI

Section 4 Linkage Editor Options and Subcommands

Options and subcommands specify file names and give the Linkage Editor various instructions, such as the order in which sections are to be linked. Options and subcommands have four types of functions: File control, memory allocation, execution control, and debugging support. These functions may be used independently or in combination to edit load modules in various ways.

(1) File Control Functions: File control functions specifies input files and output files to the Linkage Editor. Input files include object module files, relocatable load module files and library files. Output files are load module files and list files.

(2) Memory Allocation Functions: Memory allocation functions can tell the Linkage Editor the order in which sections are to be linked and give their starting addresses. They can also specify the address at which the output load module is to start executing. These functions are used to change the order in which sections are linked, or to create a load module that is to execute from a specified address.

(3) Execution Control Functions: Execution control functions specify the form in which the Linkage Editor is to input and output information, and end Linkage Editor operations. These are used to input subcommands from a subcommand file, or to incorporate debugging information in a load module.

(4) Debugging Support Functions: Debugging support functions display contents of a load module during a linkage operation, or change information such as export and import symbol names, etc. These are useful at the program debugging stage, for confirming interim linkage results, or for provisional recovery from errors.

Options and subcommands have the same names and have equivalent functions, but are specified using different formats. Moreover, there are some specifications which can be made only with either subcommands or options. Section 4.1, "Option and Subcommand Formats," and section 4.2, "List of Options and Subcommands," should accordingly be read carefully.

For details on the functions and means of specifying each option and subcommand, refer to sections 4.3, "File Control," through 4.6, "Debugging Support."

HITACHI

4.1 Option and Subcommand Formats

Each option or subcommand consists of a <u>name and parameters, which together must not exceed 255 characters.</u>

(1) Option and Subcommand Structure:

    (a) Name: The name part gives the name of the option or subcommand. For details, see section 4.2, "List of Options and Subcommands."

    (b)   Parameters: The parameter part gives information such as the name of files on which the option or subcommand operates, and address values. There are different requirements and methods of specification depending on the option or subcommand. See sections 4.3, "File Control," 4.4, "Memory Allocation," 4.5, "Excution Control," and 4.5, "Debugging Support."

    Options and subcommands differ as to the way of separating the name from the parameters. <u>Options use an equals sign (=), while subcommands use one or more spaces or tabs.</u>

    Option format

```
<Name>=<parameters>
```

    Subcommand format

```
<Name>Δ<parameters>
```

    EXAMPLES:

```
-OUTPUT=loadf...................option

OUTPUTΔloadf...................subcommand
```

In these examples, "OUTPUT" is the name, and "loadf" is the parameter.

(2) Continuation Specification for a Subcommand: When a subcommand is too long to be specified on one line, a continuation specifier is used. This is an ampersand (&) at the end of the line. It must always be placed in between two parameters; if it is placed within a parameter, it will be interpreted as part of the parameter. If a character (other than a space or tab) is typed after the ampersand, an error will occur and the subcommand will not be continued.

If continuation is specified in interactive mode, a hyphen (-) appears as a prompt for further input.

48

HITACHI

EXAMPLES:

```
: INPUTΔ obj00, lib (mod0,mod1), & RET
-obj01, obj02 RET            →Continuation specifier
: INPUTΔ obj00, lib (mod0, mod1), ob& RET
```

: Not a continuation line

Processed under the file name ob& due to specification within parameter

(3) Specifying Comments in a Subcommand File: A comment specifier is used when it is to add notes or other comments in a subcommand file. The specifier is a semicolon (;) placed on a subcommand line, indicating that the rest of the line is a comment. At least one space or tab must set off the semicolon from the subcommand name or parameter.

When a semicolon is placed at the beginning of a subcommand line, the entire line is taken as a comment.

EXAMPLES:

```
; EXAMPLE OF LINKAGE SUBCOMMAND
```
...............The entire line is a comment.

```
LIBRARYΔsyslibΔ; INDICATES LIBRARY FILE
```
..........."INDICATES LIBRARY FILE" is a comment.

```
INPUTΔobject.rel;abc
```
           "object.rel;abc" is treated as one parameter.

HITACHI

4.2 List of Options and Subcommands

There are 15 options and 23 subcommands. The options and subcommands are listed in table 4-1.

Options and subcommands may be written either in upper-case or lower-case letters.

HITACHI

Table 4-1 List of Options and Subcommands

No.	Type	Option/ Subcommand Name	Function	Option	Sub- command	Section
1	File control	INPUT	Specifies input file	No	Yes	4.3.1
		OUTPUT* (NOOUTPUT	Specifies output file	Yes	Yes	4.3.2
		LIBRARY (NOLIBRARY)*	Specifies library file	Yes	Yes	4.3.3
		PRINT (NOPRINT)*	Specifies list file	Yes	Yes	4.3.4
		EXCLUDE (NOEXCLUDE)*	Excludes modules from linking	Yes	Yes	4.3.5
2	Memory allocation	START	Specifies section starting address and linking order	Yes	Yes	4.4.1
		ENTRY	Specifies execution start address	Yes	Yes	4.4.2
		AUTOPAGE (NOAUTOPAGE)*	Specifies automatic paging	Yes	Yes	4.4.3
		CPU	Specifies address check	Yes	Yes	4.4.4
		ROM	Specifies support of storing program in ROM	Yes	Yes	4.4.5
3	Execution control	EXCHANGE	Substitutes units	No	Yes	4.5.1
		SUBCOMMAND	Specifies subcommand file	Yes	Yes	4.5.2
		FORM	Specifies format of output load module file	Yes	Yes	4.5.3
		DEBUG (NODEBUG)*	Specifies output of debugging information	Yes	Yes	4.5.4
		END	Terminates subcommand input	No	Yes	4.5.5
		EXIT	Terminates linkage operation	No	Yes	4.5.6
		ABORT<t>	Aborts linkage operation	No	Yes	4.5.7
		ECHO*(NOECHO)	Specifies subcommand file echo-back	Yes	Yes	4.5.8
		UDF* (NOUDF)	Specifies display of	Yes	Yes	4.5.9

HITACHI

No.	Type	Option/ Subcommand Name	Function	Option	Sub- command	Section
			undefined symbols			

Notes: 1. The shortest permissible abbreviated forms are underlined.

2. Yes and No in the table indicate whether an item can be used as an option or subcommand.

3. An asterisk indicates the default option or subcommand.

HITACHI

Table 4-1 List of Options and Subcommands (cont)

No.	Type	Option/ Subcommand Name	Function	Option	Sub- command	Section
4	Debugging support	<u>LIS</u>T	Displays interim linkage information	No	Yes	4.6.1
		<u>RE</u>NAME	Changes name of unit, export symbol, or import symbol	No	Yes	4.6.2
		<u>DEL</u>ETE	Deletes unit or export symbol	No	Yes	4.6.3
		<u>DEF</u>INE	Forcibly defines import symbol	Yes	Yes	4.6.4

Notes: 1. The shortest permissible abbreviated forms are underlined.
2. Yes and No in the table indicate whether an item can be used as an option or subcommand.
3. An asterisk indicates the default option or subcommand.

(1) **Negative Form of Options and Subcommands:** For some options and subcommands a negative form starting with "NO" may be specified. Parameters cannot be specified with negative-form options and subcommand. There are eight negative option/subcommand forms, as follows:

(a) NOOUTPUT............Suppresses output of load module file

(b) NOLIBRARY...........Specifies non-use of a library file

(c) NOPRINT.................Suppresses output of a list file

(d) NOEXCLUDE.........Specifies linking of modules

(e) NOAUTOPAGE......Suppresses automatic paging

(f) NODEBUG...............Suppresses output of debugging information

(g) NOECHO.................Suppresses echo-back of a subcommand file

(h) NOUDF...................Suppresses display of undefined symbols

(2) **Option Default:** When an option is omitted, the following are the default choices.

(a) OUTPUT (no parameters)

(b) NOLIBRARY

(c) NOPRINT

(d) NOEXCLUDE

(e) NOAUTOPAGE

(f) FORM=A

(g) NODEBUG

HITACHI

(h) ECHO
(i) UDF

HITACHI

(3) Abbreviating Option and Subcommand Names: Names of options and subcommands may be abbreviated to the point where the name can still be distinguished from other names. For example, consider the name "DEBUG."

D........................Cannot be distinguished from DELETE or DEFINE, so an error occurs

DE.....................Cannot be distinguished from DELETE or DEFINE, so an error occurs

DEB..................Recognized as DEBUG

DEBU...............Recognized as DEBUG

DEBUG............Recognized as DEBUG

DEBUGS..........No such name, so an error occurs

(4) Range of Validity of Options: When only a command line is specified, linkage is executed based only on the options specified. When subcommands are specified, options specified in the command line remain valid up to the first END subcommand specified (or up to the EXIT subcommand when no END is specified). However, if subcommands are specified which conflict with the function of an option, an error message is displayed, the option becomes invalid, and execution proceeds according to the subcommand specification. After the first END subcommand, all subsequent subcommand specifications are valid.

EXAMPLE:

%lnkΔ -NOOUTPUT RET}	.} The NOOUTPUT option is in effect,
.	}so no output file is created.
.	}
:END RET	}
.	}
..	}
:OUTPUT Δ loadfile RET	} The OUTPUT subcommand is now valid,
.	}so output file "loadfile.abs" is created.
.	}

HITACHI

In the following sections the format below is used to describe each option and subcommand.

No. ————————————————————————— **INPUT**   ← Heading for each option or subcommand

| **Format** | Name | Option | Subcommand | Negative Form | |
| | Parameters | | | | |

← Option or subcommand name, and format for specifying parameters; underline indicates shortest abbreviation

Function ←— Summary of option or subcommand functions

Explanation ←— Detailed description of functions, and restrictions

Examples ←— Examples of option or subcommand specifications

Note: The examples are written for UNIX system, write slash (/) instead of hyphen (-) for MS-DOS system, and write all options and subcommands in capital letters.

HITACHI

4.3 File Control

4.3.1 INPUT - Specifies Input Files   INPUT

Format	Name	Option	Subcommand	Negative Form
		None	INPUT	None
	Parameters	<Input file name>[(<module name>[,<module name>...])]		
		[,<Input file name> [(<module name>[,<module name>...])]...]		

Function   Specifies files and modules to be input.

Explanation (1) Outline of functions:

- The files specified by parameters, or the specified modules in those files, are input to the Linkage Editor.
- Three kinds of files can be specified object module files, load module files, and library files.
- Modules may be specified only for library files, in which case only the specified modules from the library file will be input.
- If the file type is omitted from a file name, the Linkage Editor will automatically assume the type as follows.

    No module name specified............".obj"

    Module name specified...............".lib"

(2) Restrictions in use:

- Among load module files, only relocatable load modules may be specified. If an absolute load module is specified, an error will occur and the file will not be input.
- If a module other than that in a library file is specified, an error will occur and the file will not be input.
- The maximum number of input files that may be treated in one linkage process is 256, including library files. If more than 256 files are specified, an error will occur, and only the first 256 files specified will be input. To process more than 256 files, use the multilinkage function.
- Page type and non-page type modules may not be input at the same time. If both types of modules are input together, an error will occur and the Linkage Editor will stop execution.

Examples   INPUTΔmain

    Inputs the object module file "main.obj."

HITACHI

INPUTΔfunclib (sin, cos), tan.o

Inputs the modules "sin" and "cos" from library file "funclib.lib," and
inputs the object module file "tan.o."

HITACHI

4.3.2 OUTPUT-Specifies an Output File   OUTPUT

Format	Name	Option	Subcommand	Negative Form
		OUTPUT	OUTPUT	NOOUTPUT
	Parameters	[<Output file name>]		

Function   Specifies a load module output file name.

Explanation

(1) Outline of functions:

- Outputs the load module generated by the Linkage Editor to the specified file.
- If the file type is omitted from the file name, the Linkage Editor will automatically assign a file type according to the format of the load module file, as follows.
- Absolute format .... ".abs"
- Relocatable format .... ".rel"
- The format of the load module file is specified using the FORM option or subcommand. If no specification is made, absolute format is used.
- If no output file name is specified using the OUTPUT option or subcommand, the output file is given the name of the first specified input file plus the above file type.
- If the NOOUTPUT option or subcommand is specified, no load module file will be output.

(2) Restrictions in use:

- No parameters may be specified with the NOOUTPUT option or subcommand.
- If an output file name is specified, it must be different from all input file names.

Examples   -OUTPUT=prgload

Outputs load module file "prgload.abs" (or "prgload.rel").

-OUTPUT

Outputs load module file with the name of the first specified object module file plus ".abs" (or ".rel").

OUTPUTΔmain.10

Outputs load module file "main.10."

Format	Name	Option		Subcommand	Negative Form
		LIBRARY		LIBRARY	NOLIBRARY
	Parameters	<Library file name>[,<library file name>...]			

Function   Specifies input library files.

Explanation

(1) Outline of functions:

- Specifies library files which the Linkage Editor is to search if there are unresolved import symbols after linkage operations among specified input files are completed.

- If both user library files and system library files are specified, the Linkage Editor will search the user library files first.

- If no file type is specified with the library file name, the Linkage Editor automatically assumes this to be ".lib."

- If the NOLIBRARY option or subcommand is specified, there will be no input from a library file (including default libraries). When linkage is controlled by subcommand specification, however, the range of validity of this option is limited. For details see "Range of Validity of Options" under section 4.2.

(2) Restrictions in use:

- Only library files created using the H Series Librarian may be input to the Linkage Editor.

- The maximum number of input files that may be treated in one linkage operation is 256, including library files. If more than 256 files are specified, an error will occur, and only the first 256 files specified will be input. To process more than 256 files, use the multilinkage function.

- Page type and non-page type modules may not be input at the same time. If both types of modules are input together, an error will occur and the Linkage Editor will stop execution.

- No parameters may be specified with the NOLIBRARY option or subcommand.

Examples   -LIBRARY=syslib.

                    Specifies library file "syslib."

                LIBRARYΔsystem, debug

                    Specifies library files "system.lib" and "debug.lib."

HITACHI

4.3.4 PRINT-Specifies a List File
PRINT

Format	Name	Option	Subcommand	Negative Form
		PRINT	PRINT	NOPRINT
	Parameters	{<List file name>} {# }		

Function   Specifies a list file for output of linkage list.

Explanation

(1) Outline of functions:

- Outputs a linkage list to the specified list file.
- If the parameter "#" is specified, the list file is output to the standard output device.
- If no PRINT option or subcommand is specified, or if the NOPRINT option or subcommand is specified, the linkage list will not be output.
- If no file type is specified with the list file name, the Linkage Editor will automatically assume this to be ".map."
- On the contents of the linkage list, see section 6.1, "Linkage Lists."

(2) Restrictions in use:

- No parameters may be specified with the NOPRINT option or subcommand.

Examples   -PRINT=linkage

   Outputs a linkage list to list file "linkage.map."

PRINTΔearth.prn

   Outputs a linkage list to list file "earth.prn."

HITACHI

4.3.5 EXCLUDE-Excludes Modules from Linking EXCLUDE

Format	Name	Option	Subcommand	Negative Form
		<u>EXCL</u>UDE	<u>EXCL</u>UDE	<u>NOEX</u>CLUDE
	Parameters	None		

Function   Specifies that modules defining non-referenced import symbols should not be linked.

Explanation

(1) Outline of functions:
- If an import symbol is not referenced, the module defining it is not linked.
- When the NOEXCLUDE option or subcommand is specified, modules defining non-referenced import symbols are linked. The defining modules are also linked if the EXCLUDE option or subcommand is omitted.

(2) Restrictions in use:
- The EXCLUDE subcommand cannot be used after input files have been specified by the INPUT or EXCHANGE subcommand.
- The EXCLUDE option or subcommand can be specified only when the output load module is in absolute format. When the multilinkage function is used to create an absolute load module in the final linkage process, if the default library function is also used, the modules from the default library will be included in the first linkage process. If you want the default library to be included in the last linkage process, specify the NOLIBRARY subcommand for the intermediate linkage processes.

Examples   -EXCLUDE

If an import symbol is not referenced, the module defining it is not linked.

HITACHI

4.4 Memory Allocation

4.4.1 START-Specifies Start Address and Linkage Order of Sections   START

Format	Name	Option	Subcommand	Negative Form
		START	START	None
	Param-eters	Option	<Section name>[,<section name>...][/[<page address>:]<start address>]	
		UNIX:	[,<section name>[,<section name>...][/[<page address>:]<start address>]...]	
		MS-DOS:	<Section name>[,<section name>...][([<page address>:]<start address>)] [,<section name>[,<section name>...][([<page address>:]<start address>)]...]	
		Sub-com-mand	<Section name>[,<section name>...] [{ }]	{ /[<page address>:]<start address> }
				{ ([<page address>:]<start address>)}
			[,<section name>[,<section name>...] [{ }]...]	{ /[<page address>:]<start address> }
				{([<page address>:]<start address>)}

Function   Specifies the order in which sections are linked, and their start addresses.

Explanation

(1) Outline of functions:
- Sections are allocated from the specified address and in the specified order.
- If the start address is not specified and only the section linkage order is specified, and sections are assigned addresses starting from zero.
- Page address may be specified only for page type modules. If the page address is not specified, it is assumed to be zero.
- The page address and start address are specified in hexadecimal notation.
- When sections not specified in the parameters are input, those sections are assigned after the series of sections with the highest specified start address.
- If no START option or subcommand is specified, sections will be allocated to addresses starting from zero in the order of appearance.
- The START option or subcommand can be specified more than once.

(2) Restrictions in use:
- If the load module to be output is in relocatable format, the START option or subcommand may not be used.

HITACHI

- If a page address is specified for non-page type modules, an error will occur and the Linkage Editor will stop execution.
- Hexadecimal numbers must start with numbers 0 through 9.

    EX: 0ABCD..............proper designation

    ABCD ....incorrect designation

- Page addresses may be assigned in the range from 0 through 0FF (hexadecimal).
- The range of start addresses that may be specified varies with the H Series model.

    H8/500 Series: 0 through 0FFFF (hexadecimal)

    H8/300 Series: 300HA[*] 0 through 0FFFFFF (hexadecimal)

    Others 0 through 0FFFF (hexadecimal)

    H32 Series: 0 through 0FFFFFFFF (hexadecimal)

    SH Series: 0 through 0FFFFFFFF (hexadecimal)

Note: [*]"300HA" indicates the advanced mode of H8/300 Series.

HITACHI

Examples   -START=CODE, DATA, BSS, STACK

> Sections are linked in the order "CODE," "DATA," "BSS," "STACK," and are allocated to addresses starting from zero.

-START=CONTROL, BANK0, BANK1/OF00

> Sections are linked in the order "CONTROL," "BANK0," "BANK1," and are allocated to addresses starting from 0F00 (hexadecimal).

STARTΔRAM0, RAM1 (8000), ROM1, ROM2 (1000), ROM0

> Sections "RAM0" and "RAM1" are linked in that order and are allocated addresses starting from 8000 (hexadecimal). Sections "ROM1" and "ROM2" are linked in that order and are allocated to addresses starting from 1000 (hexadecimal). Section "ROM0" is allocated to addresses starting from zero.

HITACHI

4.4.2 ENTRY-Specifies Execution Start Address   ENTRY

Format

Name	Option	Subcommand	Negative Form
	ENTRY	ENTRY	None
Parameters	<Export symbol>		

Function   Specifies the start address for executing a load module.

Explanation

(1) Outline of functions:

- Sets the address value of an export symbol as the execution start address of a load module to be output.
- If no ENTRY option or subcommand is specified and the output load module format is absolute, the execution start address becomes the start address of the first code section in the output load module.

(2) Restrictions in use:

- If an ENTRY option or subcommand is specified more than once, the last specified address is valid.

Examples   -ENTRY=PRG_ENT

The address of export symbol "PRG_ENT" is set as the execution start address.

ENTRYΔMAIN

The address of export symbol "MAIN" is set as the execution start address.

HITACHI

4.4.3 AUTOPAGE-Specifies Autopaging Function AUTOPAGE

Format	Name	Option	Subcommand	Negative Form
		AUTOPAGE	AUTOPAGE	NOAUTOPAGE
	Parameters	None		

Function Specifies autopaging in assignment of addresses to page type modules.

Explanation

(1) Outline of functions:

- When a page type module is linked, addresses are assigned by automatic paging.
- If the AUTOPAGE option or subcommand is not specified, or if the NOAUTOPAGE option or subcommand is specified, addresses are not assigned by automatic paging.

(2) Restrictions in use:

- The AUTOPAGE option or subcommand may not be specified when linking non-page type modules are linked. Such specification will result in an error, and the Linkage Editor will stop execution.
- If the NOAUTOPAGE option or subcommand is specified when page type modules are linked, sections may overlap page boundaries. If overlap occurs, the Linkage Editor displays a warning.

Examples   AUTOPAGE

Addresses are assigned by autopaging.

-NOAUTOPAGE

Addresses are assigned without regard to page boundaries.

HITACHI

4.4.4 CPU-Specifies Address Check Using a CPU Information File CPU

Format	Name	Option		Subcommand	Negative Form
		<u>C</u>PU		<u>C</u>PU	None
	Parameters	<CPU information file name>			

Function   Specifies execution of an address check using a CPU information file.

Explanation

(1) Outline of functions:

- The validity of addresses assigned to each section is checked, based on CPU information. In the following cases the section address assignment is regarded as invalid, and the Linkage Editor displays a warning. The sections, however, are output to the load module file without changing the addresses.
- (a)   When sections are assigned addresses in areas other than memory.
- (b)   When one section is assigned to addresses overlapping memory areas having different memory types and attributes.
- If no file type is specified with the CPU information file, the Linkage Editor will automatically assume this to be ".cpu."

(2) Restrictions in use:

- In the following cases the Linkage Editor displays a warning message, and the CPU option or subcommand is invalid.
- (a)   Relocatable format is specified as the load module output format with the FORM option or subcommand.
- (b)   The information format of the CPU information file is invalid.
- (c)   A CPU information file is specified for linkage processing of object modules not for the H8/300 or SH series.
- When a CPU option or subcommand is specified more than once, a warning message is displayed, and only the last-specified file is valid.

Examples   -CPU=cinf

Inputs CPU information file "cinf.cpu."

CPUΔc300.inf

Inputs CPU information file "c300.inf."

HITACHI

4.4.5 ROM-Specifies Support of Storing Program in ROM<t>ROM

Format	Name	Option	Subcommand	Negative Form
		ROM	ROM	None
	Parameters	UNIX:	<Section 1>/<Section 2>[,<Section 1>/<Section 2>...]	
		PC: (<Section 1>,<Section 2>)[(<Section 1>,<Section 2>),...]		
		<Section 1>: section name of source initialized data area in ROM		
		<Section 2>: section name of destination initialized data area in RAM		

Function   Reserves a RAM area for updating initialized data values stored in ROM.

Explanation

(1) Outline of functions:

- In the output load module, a section with the same section size as the specified section 1 is reserved as section 2. Section 2 has the same section attributes as section 1.
- References to symbols declared in section 1 are relocated to addresses in section 2. Specify a relocatable section as section 1.
- Up to ten pairs of section is 1 and section 2 pairs can be specified.
- For details of the support of storing program in ROM, see section 2.7, "Support of Storing Program in ROM."

(2) Restrictions in use:

- The ROM option or subcommand cannot be specified when the output load module has the relocatable format.
- If two sections have the same name and this name is specified as section 1, the section input first is selected.
- An error occurs if section 1 does not exist.
- A dummy section cannot be specified as section 1.
- When an existing section is specified as section 2, the following conditions must be satisfied.

  (a) The size of section 2 in each unit is 0.

  (b) Section 2 is the relocatable section.

  (c) Both section 1 and section 2 have the same attribute.

HITACHI

- The ROM option and subcommand are available in Linkage Editor Ver. 4 and later.

Examples   UNIX:-ROM=D/RAM_SCT

PC:/ROM=(D, RAM_SCT)

> Section RAM_SCT, equal in size to section D, is reserved in the output load module. References to symbols allocated to section D are relocated to addresses on RAM_SCT.

HITACHI

4.5 Execution Control

4.5.1 EXCHANGE-Forcibly Replaces Units   EXCHANGE

Format	Name	Option		Subcommand	Negative Form
		None		EXCHANGE	None
	Parameters	<Input file name>[(<unit name>[,<unit name>...])]			

Function   Replaces units in an input file by units of the same name in the load module being processed by the Linkage Editor.

Explanation

(1) Outline of functions:

- Units in the specified input file are replaced by units of the same name in the load module being processed by the Linkage Editor.
- An object module file or load module file may be specified as the input file.
- If a load module is specified as the input file without specifying unit names, all the units in that load module file will be usable for replacement.
- If no file type is given with the input file name, the Linkage Editor will automatically assume ".obj" as the file type.
- Replacement of units takes place after all input files have been included. If more than one EXCHANGE subcommand is specified, units will be replaced in the order of specification.

(2) Restrictions in use:

- An absolute load module may not be specified. If an absolute load module is specified, an error will occur, and the file will not be input.
- A library file may not be specified as the input file. If a library file is specified, an error will occur, and the file will not be input.

Examples   EXCHANGEΔdatain

        Replaces units in the object module file "datain.obj" by units of the same name in the load module file being processed.

      EXCHANGEΔfunction.rel(tan,atan)

        Replaces the units "tan" and "atan" in relocatable load module file "function.rel" by units of the same name in the load module file being processed.

HITACHI

4.5.2 SUBCOMMAND-Specifies a Subcommand File<t>SUBCOMMAND

Format	Name		Option	Subcommand	Negative Form
			SUBCOMMAND	SUBCOMMAND	None
	Parameters <Subcommand file name>				

Function   Specifies a subcommand file for input.

Explanation

(1) Outline of functions:

- Subcommands are input from the specified subcommand file.
- If the SUBCOMMAND option is not specified on the command line, and no input file is specified there, the Linkage Editor will link modules according to the subcommands input in interactive mode.
- If the SUBCOMMAND option is not specified on the command line but one or more input files are specified there, the Linkage Editor will link modules according to the command line specification.

(2) Restrictions in use:

- When a subcommand and file is specified on the command line together with input files or other options, the subcommand file is executed as the last option, regardless of its specification position. For example:

  UNIX: lnk <u>in1, in2</u> - <u>SUB = linkage.sub</u> - <u>FORM = R</u>

  ①   ②   ③

  PC: LNK <u>In1, In2</u> / <u>SUB = LINKAGE.SUB</u> / <u>FORM = R</u>

  ①   ②   ③

This command line is interpreted and executed in the order ③, ②, ①. If FORM=A is specified in linkage.sub, FORM=A is valid (because it is interpreted afterward).

- The SUBCOMMAND subcommand cannot be specified in a subcommand file.

Example.   -SUBCOMMAND=linkage.sub

               Subcommand file "linkage.sub" is input, and the Linkage

               Editor links modules according to the contents of this file.

HITACHI

4.5.3 FORM-Specifies Output Load Module File Format   FORM

Format	Name	Option	Subcommand	Negative Form
		FORM	FORM	None
	Parameters	{A}		
		{R}		

Function   Specifies the output load module file format as either absolute or relocatable.

Explanation

(1) Outline of functions:

- If parameter "A" is specified, the load module file will be output in absolute format.
- If parameter "R" is specified, the load module file will be output in relocatable format.
- If no FORM option or subcommand is specified, the load module will be output in absolute format.

(2) Restrictions in use:

- The parameter "R" cannot be specified when the ROM or START option or subcommand is specified.

Examples   -FORM=R

The load module file is output in relocatable format.

FORMΔA

The load module file is output in absolute format.

HITACHI

4.5.4 DEBUG-Specifies Output of Debugging Information DEBUG

Format	Name	Option	Subcommand	Negative Form	
		DEBUG	DEBUG	NODEBUG	
	Parameters	None			

Function   Specifies incorporation of debugging information in the output load module file.

Explanation

(1) Outline of functions:

- Incorporates debugging information in the output load module file. This information is required for symbolic debugging using the Simulator/Debugger.
- If no DEBUG option or subcommand is specified, or if the NODEBUG option or subcommand is specified, debugging information will not be incorporated in the output load module file.

(2) Restrictions in use:

- If the NOOUTPUT option or subcommand is specified, specification of a DEBUG option or subcommand has no meaning.

Examples   DEBUG

Debugging information is incorporated in the output load module file.

-NODEBUG

Debugging information is not incorporated in the output load module file.

HITACHI

4.5.5 END-Specifies End of Subcommand Input   END

Format	Name	Option	Subcommand	Negative Form
		None	END	None
	Parameters	None		

Function Temporarily ends input of subcommands and begins linkage operation (after which subcommand input is resumed).

Explanation

(1) Outline of functions:

- Temporarily ends input of subcommands and begins a linkage operation. After the linkage operation is completed, the Linkage Editor is initialized and subcommand input is resumed.
- When the multilinkage function is used to perform multiple linkage operations with one execution of the Linkage Editor, the END subcommand indicates the end of one linkage process.
- When the multilinkage function is not used, or when specifying the end of the final linkage process in a multilinkage operation, use the EXIT subcommand in place of the END subcommand.

(2) Restrictions in use:

- If, for a single linkage process, the END subcommand is specified without specifying input files, an error will occur.

Example   END

Temporarily ends subcommand input and begins a linkage operation.

HITACHI

4.5.6 EXIT-Specifies End of Linkage Operation

Format	Name	OptionSubcommand	Negative	Form
		None	<u>EXIT</u>	None
	Parameters	None		

Function   Ends subcommand input and begins linkage operation (subcommand input is not resumed).

Explanation Outline of functions:

- Ends subcommand input and begins linkage operation. After the linkage operation is completed, ends the Linkage Editor execution.
- When execution is controlled from a subcommand file, if no EXIT subcommand is specified, the Linkage Editor waits for further subcommand input.
- If, for a single linkage process, the EXIT subcommand is specified without specifying input files, an error will occur.

Example   EXIT

- Ends subcommand input and begins linkage operation.

HITACHI

4.5.7 ABORT-Specifies Forced End of Linkage Operation<t>ABORT

Format	Name	Option	Subcommand	Negative Form
		None	ABORT	None
	Parameters	None		

Function   Specifies forced end of linkage operation.

Explanation Outline of functions:

- Forcibly ends Linkage Editor operation.
- The ABORT subcommand is useful to interrupt Linkage Editor operation when a mistake such as subcommand input mistake has been made.

Example   ABORT

        Brings Linkage Editor execution to a forced end.

HITACHI

4.5.8 ECHO-Specifies Subcommand File Echo-Back<t>ECHO

Format	Name	Option	Subcommand	Negative Form	
		ECHO	ECHO	NOECHO	
	Parameters	None			

Function Can select whether or not to suppress echo-back of subcommands when a subcommand file is executed.

Explanation Outline of functions:

- The ECHO option or subcommand displays subcommands on the console when a subcommand file is executed. Subcommands are displayed even if the ECHO option or subcommand is not specified.
- The NOECHO option or subcommand suppresses display of subcommands on the console when a subcommand file is executed.

Examples   -ECHO

> Subcommands executed will be displayed on the console when a subcommand file is executed.

78

HITACHI

4.5.9 UDF - Specifies Display of Undefined Symbols   UDF

Format	Name	Option	Subcommand	Negative Form
		UDF	UDF	NOUDF
	Parameters	None		

Function   Specifies whether to display a warning message when an undefined symbol remains.

Explanation

(1) Outline of functions:

- Warning message 105 is displayed if an undefined symbol remains when a relocatable load module is created. This message is also displayed if an undefined symbol remains when the UDF option or subcommand is omitted.

- When the NOUDF option or subcommand is specified, a warning message is not displayed if there is an undefined symbol when a relocatable load module is created.

(2) Restrictions in use:

- The NOUDF option or subcommand is ignored when an absolute load module is created.

Examples   -FORM=R-NOUDF

A warning message will not be displayed if there is an undefined symbol when the relocatable load module is created.

HITACHI

4.6 Debugging Support

4.6.1 LIST-Displays Interim Linkage Information LIST

Format	Name	Option	Subcommand	Negative Form
		None	<u>LIST</u>	None
	Parameters	{M}		
		{U}		
		{X}		

Function   Displays linkage information of an input file.

Explanation

(1) Outline of functions:

- Outputs linkage information to the standard output device concerning the files currently being input.
- Content of the displayed information depends on the specified parameters, as follows.

    M........Displays a link map

    U........Displays unresolved import symbols

    X........Displays export symbols

(2) Restrictions in use:

- To display linkage information according to the input files, the information displayed is restricted as follows.
- When parameter M is specified.

    The start address of a relocatable section is always 0.

- When parameter U is specified

    The display shows import symbols for which there is no corresponding export symbol in the input files specified in INPUT subcommands up to the location of the LIST subcommand.

Examples   LISTΔM

    Displays a linkage map for the load module being processed.

LISTΔU

    Displays unresolved import symbols in the load module being processed.

HITACHI

4.6.2 RENAME-Changes the Names of Units, Export Symbols, or Import Symbols
RENAME

Format	Name	Option		Subcommand	Negative Form
		None		<u>RENAME</u>	None
	Parameters	{UN=<unit name 1> (<unit name 2>)}			
		{ER=<unit name>.<import symbol 1>}			
		{(<import symbol 2>)}			
		{ED=<unit name>.<export symbol 1>}			
		{(<export symbol 2>)}			
		{UN=<unit name 1>(<unit name 2>)}			
		{ER=<unit name>.<import symbol 1>}			
		{ , } { (<import symbol 2>) } ...}			
		{ED=<unit name>.<export symbol 1>}			
		{(<export symbol 2>)}			

Function   Changes the names of units, export symbols or import symbols in input files.

Explanation (1) Outline of functions:

- Changes the names of each specified unit, export symbol, or import symbol in input files to the name designated in parentheses ("()").
- In the case of a unit, the unit name specified following "UN=" is changed to the unit name in parentheses.
- In the case of an import symbol, the symbol name specified following "ER=" is changed to the name in parentheses. The import symbol name is preceded by the name of the unit in which the symbol exists, and is set off from the unit name by a period (.).
- In the case of an export symbol, the symbol name specified following "ED=" is changed to the name in parentheses. The export symbol name is preceded by the name of the unit in which the symbol exists, and is set off from the unit name by a period (.).

HITACHI

Explanation (2) Restrictions in use:

- The RENAME subcommand will affect the input files specified only in the first INPUT subcommand after the RENAME subcommand.
- Only the following five subcommands may be specified immediately after the RENAME subcommand:

  (a)   INPUT subcommand

  (b)   EXCHANGE subcommand

  (c)   RENAME subcommand

  (d)   DELETE subcommand

  (e)   ABORT subcommand

When more than one RENAME subcommands are specified, or when RENAME and DELETE subcommands are specified together, operation takes place in the order of specification.

Examples   RENAMEΔUN=datalist (dataIst1)

Unit "datalist" is renamed as "data1st1."

RENAMEΔED=cnt1.TRUNK   (P_TRUNK),ER=cnt11.REC_DATA (RECV_DATA)

Export symbol "TRUNK" in unit "cntl" is changed to "P_TRUNK." Likewise, import symbol "REC_DATA" in unit "cntl1" is changed to "RECV_DATA."

HITACHI

4.6.3 DELETE-Deletes Units or Export Symbols   DELETE

Format	Name	Option		Subcommand	Negative Form
		None		<u>DELETE</u>	None
	Parameters	{UN=<unit name>}			
		{ED=<unit name>.<export symbol name>}			
		(,{UN=<unit name>}...)			
		[,{ED=<unit name>.<export symbol name>} ...]			

Function   Specifies deletion of units or export symbols from input files.

Explanation

(1) Outline of functions:

- Deletes the specified units or export symbols from input files.

- In the case of a unit, the unit specified following "UN=" is deleted.

- In the case of an export symbol, the symbol specified following "ED=" is deleted. The export symbol name is set off by a period (.) from the name of the unit in which it exists.

(2) Restrictions in use:

- The DELETE subcommand will not affect input files already specified. This subcommand must be specified prior to specification of the input files in which the name of the unit or externally defined symbol to be deleted is found.

- The following five subcommands may be specified immediately after the DELETE subcommand:

    (a)   INPUT subcommand

    (b)   EXCHANGE subcommand

    (c)   DELETE subcommand

    (d)   RENAME subcommand

    (e)   ABORT subcommand

- When RENAME and DELETE subcommands are specified together, operation takes place in the order of specification.

Examples   DELETEΔUN=snap_unit

                    Deletes unit "snap_unit."

            DELETEΔUN=dummy,ED=main.DUMMY_ENTER

HITACHI

Deletes unit "dummy." Also, deletes export symbol "DUMMY_ENTER" in unit "main."

HITACHI

4.6.4 DEFINE-Forcibly Defines an Import Symbol   DEFINE

Format	Name	Option	Subcommand	Negative Form
		DEFINE	DEFINE	None
	Param-eter	Option UNIX:	<Import symbol name>/	{<numeric value>} {[<page address>:]<address>} {<export symbol name>}
			[,<import symbol name>/	{<numeric value>} {[<page address>:]<address>}...] {<export symbol name>}
		MS-DOS:	<Import symbol name>(	{<numeric value>} {[<page address>:]<address>}) {<export symbol name>}
			[,<Import symbol name>(	{<numeric value>} {[<page address>:]<address>})...] {<export symbol name>}
		Sub-com-mand	<Import symbol name>	{/}{<numeric value>}{ } { } {[<page address>:]<address>}{ } {(}{<export symbol name>} {)}\
			[,<import symbol name>	{/}{<numeric value>}{ } {[<pageaddress>:]<address>}...] {(}{<export symbol name>}{)}

Note: Specify a right parenthesis ")" only if the preceding numeric value, address, or externally
   defined symbol name is delimited by a left parenthesis "(."

Function   Specifies forced definition of import symbols.

Explanation

(1) Outline of functions:
- Forcibly defines each specified import symbol with the specified numeric value,
  address or export symbol value.
- Page address can be specified only for page type modules. If the page address is not
  specified, zero is assumed.
- Numeric values, page addresses, and addresses are specified in hexadecimal notation.

HITACHI

(2) Restrictions in use:

- When the assigned value is that of an export symbol, it must be one that has already been defined.
- If a page address is specified for non-page type modules, an error will occur and the Linkage Editor will stop execution.
- Hexadecimal numbers must start with the numbers 0 through 9.
- The range of page addresses is 0 through 0FF (hexadecimal).
- The range of addresses that may be specified varies with the H Series model.

    H8/500 Series: 0 through 0FFFF (hexadecimal)

    H8/300 Series: 300HA[*] 0 through 0FFFFFF (hexadecimal)

    Others 0 through 0FFFF (hexadecimal)

    H32 Series: 0 through 0FFFFFFFF (hexadecimal)

    SH Series: 0 through 0FFFFFFFF (hexadecimal)

Note: [*] "300HA" indicates the advanced mode of H8/300 Series.

Explanation

- Values defined by the DEFINE subcommand cannot be used in relocatable load modules.
- When the EXCLUDE option or subcommand is specified, non-referenced import symbols specified by the DEFINE subcommand are ignored.

Examples   -DEFINE=PORT10/0E8

> Defines undefined import symbol "PORT10" as a symbol having the value 0E8 (hexadecimal).

DEFINEΔMAIN_RTN (PRG_EXIT)

> Defines undefined import symbol "MAIN_RTN" as having the same value as export symbol "PRG_EXIT."

HITACHI

Section 5 Input to the Linkage Editor

5.1 Object Module Files

The Linkage Editor can accept as input the object module files output by the H Series C Compiler or Assembler.

5.2 Relocatable Load Module Files

Relocatable load module files output by this Linkage Editor may be re-input. Absolute load module files may not be re-input.

5.3 Library Files

Library files created using the H Series Librarian may be input to the Linkage Editor. Modules in library files may be specified individually, or the LIBRARY option or subcommand may be used to input modules contained in library files automatically. See further under section 4.3.3, "LIBRARY-Specifies Library Files."

5.4 Default Library Files

A library file created by the H Series Librarian can be input implicitly without specifying the LIBRARY option or subcommand. This is called the default library function.

A default library is input when the following three conditions are satisfied:

- A logical name reserved as a default library name is assigned to the library file before the library files is input to the Linkage Editor.
- The NOLIBRARY option or subcommand is not specified.
- An unresolved import symbol remains after the libraries specified by the LIBRARY option or subcommand have been searched.

The Linkage Editor inputs the library files assigned to the following logical names in the order 1, 2, 3, and searches for modules that define unresolved import symbols.

1. HLNK_LIBRARY 1

2. HLNK_LIBRARY 2

3. HLNK_LIBRARY 3

HITACHI

The user can specify library files corresponding to these logical names by using the setenv command for UNIX system and the SET command for PC system of the operating system of the operating system.

EXAMPLE:

UNIX system:

```
% setenv HLNK_LIBRARY1 user.lib
```

PC system:

```
> SET HLNK_LIBRARY1=USER.LIB
```

User library user.lib is assigned to the logical name HLNK_LIBRARY1.

HITACHI

Section 6 Output from the Linkage Editor

6.1 Linkage Lists

When the PRINT option or subcommand or the LIST subcommand is specified, the contents of a
load module file being processed are output to the standard output device or to a file, as follows.

(1) Input information   (PRINT only)
(2) Link map list   (PRINT or LIST M)
(3) Export symbol list   (PRINT or LIST X)
(4) Unresolved import list   (PRINT or LIST U)
(5) RENAME/DELETE list   (PRINT only)
(6) DEFINE list   (PRINT only)

The formats in which these lists are output are shown below.

HITACHI

(1) Input Information: Information input as command line parameters, interactive mode
subcommands, or subcommand files is output in the format shown in figure 6-1.

```
                    H SERIES LINKAGE EDITOR Ver. 5.0B
        LINK COMMAND LINE
        LNK -sub=func.sub
                     ①

        LINK SUBCOMMANDS

            inp main
            rename ed=sin.sin0(sin1)
            delete ed=sin.sin3
            inp sin
            define undef1(100),undef2(sin1)
            print fmap
            inp cos
            inp tan                                            ②
            inp calc.lib(division)
            form a
            out func
            rom (SECT1, SEC1N)
            exit
            ** sin0 IS RENAMED TO sin1
            ** sin3 IS DELETED
            ** 105 UNDEFINED EXTERNAL SYMBOL (division.undef3)
```

Figure 6-1 Typical Output of Input Information

① Shows the character string input on the command line.

② Shows the character strings input as subcommands in interactive mode, or input from a
subcommand file. Also shows error messages or informative messages in response to this
input.

HITACHI

(2) Link Map List:

(a) When the PRINT option or subcommand is specified, information on each section is output in the format shown in figure 6-2.

```
        H SERIES LINKAGE EDITOR Ver. 5.0B                          PAGE: 1


                    ***    LINKAGE EDITOR LINK MAP LIST    ***

 SECTION NAME                        START    -  END              LENGTH
                                                         UNIT NAME      MODULE NAME


 ATTRIBUTE : CODE   NOSHR   ROM
                ②       ③      ④
 SECT1                               H'00000000 - H'00000004    H'00000005
    ①                                           ⑤              main    ⑥         main
                                                                 ⑦                 ⑧
                                     H'00000006 - H'00000017    H'00000012
                                                                 sin               sin
                                     H'00000018 - H'00000019    H'00000002
                                                                 cos               cos
                                     H'0000001a - H'0000002d    H'00000014
                                                                 tan               tan
                                     H'0000002e - H'00000043    H'00000016
                                                                 division          division


 * TOTAL ADDRESS *                   H'00000000 - H'00000043    H'00000044
                                                ⑨                        ⑩
```

Figure 6-2 Typical Link Map List Output Using PRINT

HITACHI

(b) When parameter "M" is specified in the LIST subcommand, information on each file is
output in the format shown in figure 6-3.

```
        H SERIES LINKAGE EDITOR Ver. 5.0B                    PAGE: 1


                      ***   LINKAGE EDITOR LINK MAP LIST   ***


   FILE NAME    :   main.obj
                      ⑪
   MODULE NAME  :   main
                      ⑧
   UNIT NAME    :   main
                      ⑦


        SECTION NAME      ATTRIBUTE
                          START       -  END          LENGTH


   SECT1                  CODE NOSHR
     ①                    H'00000000 - H'00000004   H'00000005
                               ⑤                        ⑥
```

Figure 6-3 Typical Link Map List Output Using LIST

① Shows section names in the order in which sections are linked.

② Shows the attribute as follows.

DATA: data or common section

CODE: code section

DUMMY: dummy section

STACK: stack section

RESV: reserved

UNDEF: undefined

*****: unused

③ Shows the following link attributes.

SHR: common link

NOSHR: simple link

DUMMY: dummy link

UNDEF: link attribute undefined

*****: unused

92

HITACHI

④ Displayed for a section related to the support of storing program in ROM.

ROM ROM section (section 1 in the ROM option or subcommand)

RAM RAM section (section 2 in the ROM option or subcommand)

HITACHI

⑤ Shows start address and end address of the object in hexadecimal notation. In the case of
   page type modules, the page address and address are separated by a colon (:) as follows.

H'xxxx : xxxx
                ———— address
                ———— page address
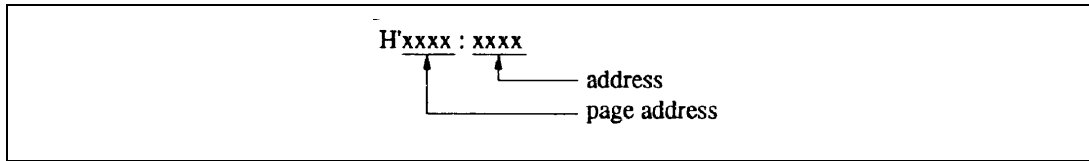
⑥ Shows size of object in hexadecimal notation.

⑦ Shows unit name.

⑧ Shows module name.

⑨ Shows start address and end address of the section.In the case of page type modules, the page
   address and address are separated by a colon (:) as follows.

H'xxxx : xxxx
                ———— address
                ———— page address
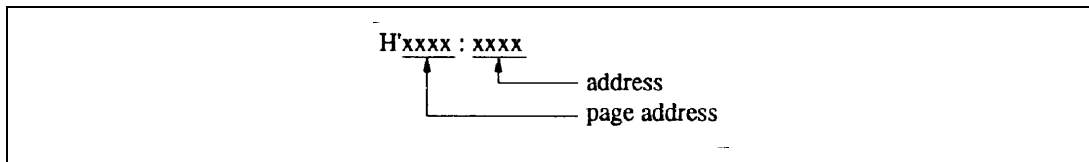
⑩ Shows total size of the section.

(11) Shows the file name (LIST only).

HITACHI

(3) Export Symbols List: This list is output when there are export symbols.

(a) When the PRINT option or subcommand is specified, a list is output in the format shown in figure 6-4.

```
        H SERIES LINKAGE EDITOR Ver. 5.0B              PAGE: 1


     ***   LINKAGE EDITOR EXTERNALLY DEFINED SYMBOLS LIST    ***


 SYMBOL NAME                         ADDR              TYPE

   cos1                            H'0000000a          EQU
   sin1                            H'00000000          DAT
   sin2                            H'00000011          DAT
    ①                                 ②                ③
```

Figure 6-4 Typical Export Symbol List Output Using PRINT

(b) When paramenter "X" is specified by the LIST subcommand, a list is output as shown in figure 6-5.

```
        H SERIES LINKAGE EDITOR Ver. 5.0B              PAGE: 1


     ***   LINKAGE EDITOR EXTERNALLY DEFINED SYMBOLS LIST    ***


 SYMBOL NAME                         ADDR              TYPE

   cos1                            H'0000000a          EQU
   sin1                            H'00000000          DAT
   sin2                            H'00000011          DAT
    ①                                 ②                ③
```

Figure 6-5 Typical Export Symbol List Output Using LIST

① Shows export symbols in alphabetical order.
② Shows the value of each export symbol in hexadecimal notation. In the case of page type modules, the page address and address are separated by a colon (:) as follows.

```
        H'xxxx : xxxx
              │    │
              │    └────── address
              └────────── page address
```
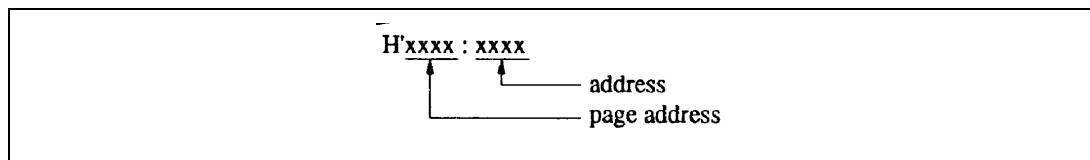
③ Shows the type of symbol as follows.

DAT: data/variable name

EQU: symbol name defined as constant value

ENT: entry name

***: undefined/unused

(4) Unresolved Import Symbol List: This list is output only when there are undefined symbols remain.

(a) When the PRINT option or subcommand is specified, a list is output in the format shown in figure 6-6.

```
        H  SERIES  LINKAGE  EDITOR Ver.  5.0B            PAGE: 1

    ***    LINKAGE  EDITOR  UNRESOLVED  EXTERNAL  REFERENCE  LIST    ***

  FILE NAME    : calc.lib
                    ①
  MODULE NAME : division
                    ②
  UNIT NAME    : division
                    ③

                        SYMBOL   NAME                TYPE
                        undef3                       ***
                          ④                           ⑤
```

Figure 6-6 Typical Unresolved Import Symbol List Output Using PRINT

(b) When parameter "U" is specified by the LIST command, a list is output as shown in figure 6-7.

HITACHI

```
              H SERIES LINKAGE EDITOR Ver. 5.0B              PAGE: 1


       ***    LINKAGE EDITOR UNRESOLVED EXTERNAL REFERENCE LIST    ***

       FILE NAME      : calc.lib
                            ①
       MODULE NAME    : division
                            ②
       UNIT NAME      : division
                            ③


       SYMBOL NAME                  TYPE

         undef1                      ***
         undef2                      ***
         undef3                      ***
           ④                         ⑤
```

Figure 6-7 Typical Unresolved Import Symbol List Output Using LIST

① Shows name of file containing undefined symbol.

② Shows name of module containing undefined symbol.

③ Shows name of unit containing undefined symbol.

④ Shows undefined symbol names in alphabetical order.

⑤ Shows undefined symbol attributes as follows.

   DAT: data/variable name

   ENT: entry name

   ***: undefined/unused

HITACHI

(5) RENAME/DELETE List: When RENAME or DELETE subcommands are used to change the name of units or symbols or delete units or symbols, specification of the PRINT option or subcommand results in output of a list in the format shown in figure 6-8.

```
        H SERIES LINKAGE EDITOR Ver. 5.0B              PAGE: 1


                 ***    LINKAGE EDITOR RENAME/DELETE LIST    ***


FILE NAME   : sin.obj
                 ①
UNIT NAME   : sin
                 ②


        FROM NAME          TO NAME              TYPE       RENAME/DELETE

        sin0               sin1                 ED         RENAME
        sin3                ④                   ED         DELETE
         ③                                      ⑤            ⑥
```

Figure 6-8 Typical RENAME/DELETE List

① Shows names of files containing the unit or symbol to be renamed or deleted in the order input.

② Shows the unit name. If the unit was renamed or deleted, the old unit name is shown.

③ Shows the name before changed.

④ Shows the name after changed. No name is shown in case of a DELETE.

⑤ Shows the type specified by subcommand, as follows.

   UN: unit name

   ED: export symbol

   ER: import symbol

⑥ Shows whether the subcommand was a RENAME or a DELETE.

HITACHI

(6) DEFINE List: When an import symbol is forcibly defined using the DEFINE option or subcommand, specification of the PRINT option or subcommand results in output of a list in the format shown in figure 6-9.

```
            H SERIES LINKAGE EDITOR Ver. 5.0B            PAGE: 1

                 ***    LINKAGE EDITOR DEFINE LIST    ***

    UNDEFINED SYMBOL      DEFINED SYMBOL        DEFINED VALUE

    undef1                                      H'00000100
    undef2                sin1                  H'00000000
       ①                   ②                       ③
```

Figure 6-9 Typical DEFINE List

① Shows forcibly defined symbol name.
② Shows the name of the export symbol which is specified.
③ Shows the value of the defined symbol in hexadecimal notation. In the case of page type modules, the page address and address are separated by a colon (:) as follows.

```
    H'xxxx : xxxx
              └────── address
        └──────────── page address
```
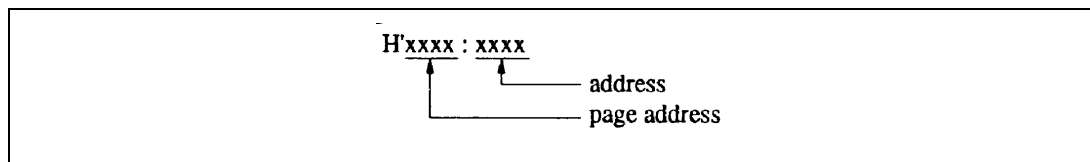
6.2 Load Module File

The Linkage Editor links a number of object modules or relocatable load module files and outputs them as a single load module file. Depending on the specification made with the FORM option or subcommand, the load module file is output in either absolute or relocatable format. A detailed explanation of the FORM option and subcommand is given in section 4.5.3, "FORM-Specifies Output Load Module File Format."

HITACHI

6.3 Console Messages

The Linkage Editor shows the following messages on the standard output device.

(1) Opening Message: This is displayed when Linkage Editor command name "LNK" is input.

```
H SERIES LINKAGE EDITOR Ver. 5.0B
Copyright (C) Hitachi, Ltd. 1989
Licensed Material of Hitachi, Ltd.
```

(2) Normal Completion Message: This is displayed when the load module file editing has been completed normally.

```
LINKAGE EDITOR COMPLETED
```

(3) Abort Message: This is displayed when the load module file editing is ended before completion, due either to an error or to specification of an ABORT subcommand.

```
LINKAGE EDITOR ABORT
```

(4) Subcommand Request Prompt: In interactive mode, a colon (:) indicates that the Linkage Editor is waiting for subcommand input.

```
:
```

(5) Subcommand Continuation Prompt: When continuation of a subcommand is specified during interactive mode execution, a hyphen (-) indicates that the Linkage Editor is waiting for continuation of the input.

```
-
```

HITACHI

(6) Informative Message: Informative messages indicate the result of Linkage Editor processing, for example when units are replaced or when an export symbol is renamed. The messages are output in the following format.

```
                    ** Δ <information>

                     ↑

                    1st column
```
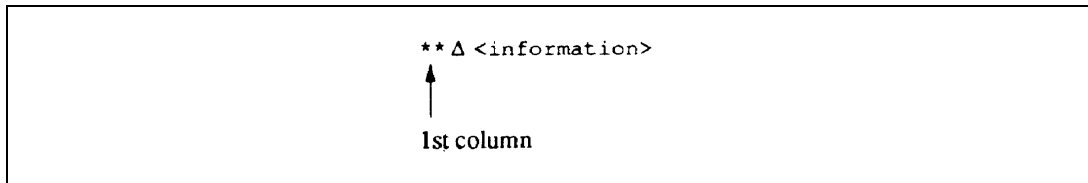
A list of informative messages is given in table 6-1.

Table 6-1 List of Informative Messages

No.	(Informative message)
	(Meaning of message)
1	<Unit name 1> IS REPLACED WITH <unit name 2>(<file name>)
	<Unit name 1> has been replaced by <unit name 2> from <file name>.
2	<External name 1> IS RENAMED TO <external name 2>
	Name of <external name 1> has been changed to that of <external name 2>.
3	<External name> IS DELETED
	<External name> has been deleted.
4	DUPLICATE UNIT-(<unit name>) IN (<file name>) IS DELETED
	More than one units of the same name <unit name> have been found, and the unit of that name in <file name> has been deleted.
5	<External name> CANNOT DEFINED
	<External name> could not be found, and therefore could not be forcibly defined.
6	<External name>/<unit name> CANNOT RENAMED
	<External name> or <unit name> could not be found, and therefore could not be renamed.
7	<Externally defined name>/<unit name> CANNOT DELETED
	<Externally defined name> or <unit name> could not be found, and therefore could not be deleted.
8	<Unit name> CANNOT REPLACED
	<Unit name> could not be found, and therefore could not be replaced.

HITACHI

Section 7 Error Messages

When options or subcommands are specified incorrectly, or if an error is detected during the linkage process, an error message is output. The Linkage Editor outputs error messages in the following form.

```
**Δ <Error number> Δ <error message>[(<additional information>)]
 ▲
 │
1st column
```

Error Number: The first digit indicates the level of the error (xx represent the second and third digits).

> 1xx: Warning   Processing of the particular module is skipped.

> 2xx: Error In the case of input from the command line or a subcommand file, processing is stopped. In interactive mode, processing of the subcommand is stopped when the error is detected, and the next subcommand is requested.

> 3xx: Fatal error   Processing is stopped.

A list of errors is given below in tables 7-1, 7-2, and 7-3 in the following format.

Error Number	Error Message	Additional Information
	Nature of Error	
	Linkage Editor actions and corrective actions	

Notation used in table: --: No additional information

HITACHI

Table 7-1 List of Warning Messages

HITACHI

101	DUPLICATE OPTION/SUBCOMAND	Option/subcommand name
	The same option or subcommand was specified more than once.	
	Only the last-specified option or subcommand is valid.	
102	TOO LONG IDENTIFIER UP TO 32	Name
	Name of a unit, section, or symbol over 32 characters was specified.	
	Name is valid up to 32nd character. The rest is ignored.	
104	DUPLICATE SYMBOL	Symbol name
	The same export symbol is defined more than once.	
	Only the first appearing symbol is valid.	
105	UNDEFINED EXTERNAL SYMBOL	Unit name, Symbol name
	An undefined symbol was imported.	
	The import is invalid, and zero is assumed as the value.	
106	REDEFINED SYMBOL	Symbol name
	A previously defined symbol was defined using the DEFINE subcommand or option.	
	The DEFINE specification is invalid.	
107	SECTION ATTRIBUTE MISMATCH	Section name
	Two sections with the same name but different attributes or boundary alignment were input.	
	The sections are processed as separate sections.	
108[*]	RELOCATION SIZE OVERFLOW	Unit name, Section name-offset value
	Relocation result exceeds the relocation size.	
	Result is rounded off to fit the relocation size.	
109	ENTRY POINT MULTIPLY DEFINED	--
	Execution start addresses were specified in more than one object modules.	
	The first appearing execution start address is valid.	
110	SECTION ADDRESS EXCEED PAGE BOUNDARY	Section name
	A section overlaps a page boundary.	
	Specify AUTOPAGE option or subcommand.	
111	DUPLICATE SECTION NAME	Section name
	Same section name was specified in options or subcommands.	

HITACHI

	The first section is valid.	
112	ILLEGAL CPU INFORMATION FILE FORMAT	--
	The file format of the CPU information file is not correct.	
	The CPU option or subcommand specification is invalid.	

HITACHI

Table 7-1 List of Warning Messages (cont)

113	CONFLICTING DEVICE TYPE	--
	The specified CPU information file is for a different CPU from that for which the input object module is intended.	
	The CPU information file specification is invalid.	
114	SECTION IS NOT IN SAME MEMORY AREA	Section name: xxxx-yyyy
	A section overlaps different memory areas. Addresses xxxx to yyyy are not allocated to one memory area.	
	The section is output to the load module without change.	
115	INACCESSIBLE ADDRESS RANGE	Section name
	A section was assigned to a memory area that cannot be used.	
	The section is output to the load module without change.	
116	INVALID CPU OPTION/SUBCOMMAND	--
	The CPU option or subcommand was specified for a relocatable load module file.	
	The CPU option or subcommand specification is invalid.	
117	ADDRESS SPACE DUPLICATE	--
	Sections overlap.	
	The load module is output as is.	
118	INVALID UDF OPTION/SUBCOMMAND	--
	The NOUDF option or subcommand was specified for an absolute output load module.	
	The NOUDF option or subcommand is invalid.	
119	RELOCATION VALUE IS ODD	Unit name, section name-offset value
	Relocation value for the displacement is odd.	
	The LSB is rounded down to fit to the relocation size.	

Note:* Warning message 108 is output if a data size designated at assembly or compile is exceeded as a result of address resolution by the Linkage Editor.

Example:

Assembler source program (example.src)

HITACHI

	.section	secl,	code	In this example the value assigned
label	.equ	$		to the label during linking is 1000,
	.			which exceeds the 1-byte data size,
	.			and therefore warning message 108
	.			is output. The upper byte (10) of
	.data.b	label	; 1 byte data	1000 is disregarded, leaving 0
	.			(zero) as the data value.

Assembly and linkage commands

```
asm8 example
lnk example-start=secl/1000
```

HITACHI

Table 7-2 List of Error Messages

201	ILLEGAL SUBCOMMAND/OPTION	--
	An illegal subcommand (or option) was specified.	
	Specify a valid subcommand (or option).	
202	SYNTAX ERROR	--
	Syntax of the specified subcommand and (or option) is incorrect.	
	Check the syntax and re-specify the subcommand (or option).	
203	TOO LONG SUBCOMMAND LINE	--
	Length of the subcommand entry exceeds 255 characters.	
	Re-specify, keeping the length within 255 characters.	
204	ILLEGAL SUBCOMMAND SEQUENCE	--
	Order of subcommand specification is invalid.	
	Check the order of subcommand specification and re-specify.	
207	ILLEGAL SECTION NAME	Section name
	The specified section name is invalid.	
	Specify a proper section name.	
208	ILLEGAL SYMBOL NAME	Symbol name
	The specified symbol name is invalid.	
	Specify a proper symbol name.	
210	TOO MANY INPUT FILES	--
	Attempt was made to input more than 256 input files at one time.	
	Create a relocatable load module file, then specify the remaining input files by re-inputting the load module file.	
211	CANNOT FOUND FILE	File name
	The specified file cannot be found.	
	Check the specified file name, then re-specify.	
212	CANNOT FOUND UNIT	Unit name
	The specified unit cannot be found.	
	Check the specified unit name, then re-specify.	
213	CANNOT FOUND MODULE	Module name
	The specified module cannot be found.	
	Check the specified module name, then re-specify.	
214	DUPLICATE START ADDRESS SPECIFIED	--
	The same start address was specified more than once.	
	Change the start address, then re-input.	

HITACHI

HITACHI

Table 7-2 List of Error Messages (cont)

216	PAGE ADDRESS EXCEEDED	--
	A page address exceeds the permitted range.	
	Check the page address and re-specify.	
217	SUBCOMMAND COMMAND IN SUBCOMMAND FILE	--
	The SUBCOMMAND subcommand appeared in a subcommand file.	
	Remove the SUBCOMMAND subcommand from the subcommand file.	
219	INVALID ADDRESS	address
	The specified address exceeds the permitted range.	
	The specified address exceeds the address range of the specified device. Check the value of the specified address, then re-execute.	
220	TOO MANY ROM COMMANDS	--
	More than 10 pairs of section names were specified in a ROM subcommand.	
	Specify 10 pairs or less.	

Table 7-3 List of Fatal Error Messages

301	ILLEGAL COMMAND PARAMETER	--
	An improper command parameter was specified.	
	Check the command parameters and re-execute.	
302	CANNOT OPEN FILE	File name
	The file cannot be opened.	
	Check the specified file name. If the file name is correct, the disk may be full, or there may be a disk hardware problem. After checking the problem, re-execute.	
303	CANNOT READ INPUT FILE	File name
	The file cannot be input.	
	Check the specified file name. If the file name is correct, the disk may be full, or there may be a disk hardware problem. After checking the problem, re-execute.	
304	CANNOT WRITE OUTPUT FILE	File name
	The file cannot be output.	
	Check the specified file name. If the file name is correct, the disk may be full, or there may be a disk hardware problem. After checking the problem, re-execute.	
305	CANNOT CLOSE FILE	File name
	The file cannot be closed.	
	Check the specified file name. If the file name is correct, the disk may be full, or there may be a disk hardware problem. After checking the problem, re-execute.	

HITACHI

Table 7-2 List of Error Messages (cont)

306	ILLEGAL FILE FORMAT	File name
	The specified file format is incorrect.	
	Check the file contents and specified file name, then re-execute. This message is output when the object file format is illegal, for example because there are two or more import symbols with the same name in the same unit, or two external symbol names were made identical by the RENAME subcommand.	
307	ILLEGAL RECORD FORMAT	File name
	There is an illegal record in the specified file, or division by zero occurred.	
	Check the source program contents. Re-assemble or re-compile, then re- execute	
308	SECTION ADDRESS OVERFLOW	Section name of the specified device
	The address allocated to a section exceeds the allowable range.	
	The address allocated to the section exceeds the address range of the specified device.	
	Change the section start address or rearrange the user program, then re-execute.	
309	ADDRESS OVERFLOW	--
	The specified address exceeds the address range allowed for the particular CPU.	
	Check the specified address, then re-execute.	
310	MEMORY OVERFLOW	--
	There is no space remaining in the Linkage Editor's usable memory.	
	Expand the memory or alter the user program, then re-execute.	
311	PROGRAM ERROR	nnn
	There is an error in the Linkage Editor program.	
	The Linkage Editor is inoperable. Check the program error number (nnn), then contact your Hitachi representative.	
312	ILLEGAL START ADDRESS ALIGNMENT	Address
	The specified address conflicts with the boundary alignment number of the object module.	
	Check the boundary alignment number of the object module, then re-execute.	

HITACHI

314	CANNOT FOUND SECTION	Section name
	The specified section name cannot be found.	
	Check the section name, then re-specify.	
319	AUTOPAGE SPECIFIED AT NON-PAGE TYPE	--
	The AUTOPAGE option/subcommand was specified when non-page type files were input.	
	Check the input file contents, then re-specify.	

HITACHI

Table 7-3 List of Fatal Error Messages (cont)

321	PAGE ADDRESS OVERFLOW	--
	The page address overflows the allowable range	
	Change the section start address or the user program so that the page address will be within the allowable range of 0-0FF (hexadecimal), then re-execute.	
322	PAGE ADDRESS SPECIFIED AT NON-PAGE TYPE	--
	For a non-page type input file, a page address was specified with the START or DEFINE option/subcommand.	
	Check the specified file name and option or subcommand content, then re-execute.	
323	SECTION SPECIFIED AT ROM OPTION	Section name
	/SUBCOMMAND DOES NOT EXIST	
	A section specified in a ROM command does not exist.	
	Check the section name, and specify again.	
325	ILLEGAL START SECTION	Section name
	A section specified by a START command has an illegal attribute.	
	Check the section attributes, and re-specify.	
326	CANNOT READ	--
	Input failed from a file (including the standard input device).	
	Check the specified file name. If the file name is correct, the disk may be full, or there may be a disk hardware problem. After checking the problem, re-execute.	
327	SYMBOL ADDRESS OVERFLOW	Symbol name
	The address assigned to a symbol exceeded the permitted range for the specified device.	
	Change the section start address or rearrange the user program, then re-execute.	
328	ILLEGAL ROM SECTION	Section name
	Section 2 specified in a ROM subcommand or option is invalid.	
	The size of section 2 is not 0, section 2 is the absolute section or the attribute of section 2 is different from that of section 1. Check the size and attribute of section 2, and re-specify.	

HITACHI

Section 8 Restrictions

Restrictions on the Linkage Editor are shown in table 8-1. If the numerical restrictions are exceeded, linkage operations cannot be performed.

Table 8-1 Restrictions on Linkage Editor Processing

No.	Item Restrictions	Remarks	
1	Number of input files	Max 256	
2	Input file formats	• Object module file output by assembler or compiler • Relocatable load module file • Library file created using librarian	
3	Address/notation	Hexadecimal only	H8/300 Series: (300HA[*]:0-0FFFFFF (Others:0-0FFFF
		Range of specification varies with H Series model	H8/500 Series: 0-0FFFF H32 Series: 0-0FFFFFFFF SH Series: 0-0FFFFFFFF
4	Names of modules, units, sections, symbols	Up to 32 characters	
5	Length of options or subcommands	Up to 255 characters	
6	Number of modules, units, sections, export symbols, import symbols	Max 65,535	Assumes no prior restrictions on memory of system on which Linkage Editor is executed.

Note: [*] "300HA" indicates the advanced mode of H8/300 Series.

HITACHI

Section 9 Object Format Conversion

In order for the load modules output by the Linkage Editor to be input into an emulator or PROM programmer, they must first be converted to S-type object format using the Object Format Converter.

9.1 Executing the Object Format Conversion

The command line format for starting the Object Format Converter is as follows.

```
cnvsΔ<Input file name>[Δ<output file name>] RET
```

Command Name: The Object Format Converter is started up by specifying the command "cnvs."

Input File Name: The name of an absolute-format load module file to be input to the Object Format Converter is specified. Relocatable load module files cannot be specified.

If the file type is omitted from the file name, the Object Format Converter automatically assumes this to be ".abs" when it inputs the file.

Output File Name: The name of the S-type object file to be output by the Object Format Converter is specified. If the file type is omitted from the file name, the Object Format Converter automatically assumes this to be ".mot" when it outputs the file.

Examples of command line specification are given below.

```
% cnvsΔprog1.lmdΔprogl.sty RET........①
% cnvsΔprog1Δprog1 RET...............②
```

① File "prog1.lmd" is input, and file "progl.sty" is output.

② File "prog1.abs" is input, and file "prog1.mot" is output.

The S-type object format is shown in figure 9-1.

(a) Header record
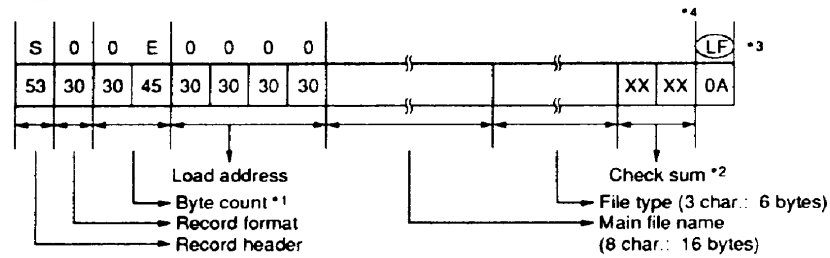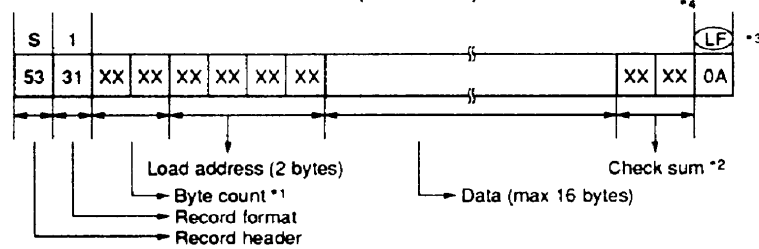
S	0	0	E	0	0	0	0	...	...	XX	XX	LF *3
53	30	30	45	30	30	30	30	...	...	XX	XX	0A

*4

- Load address
- Byte count *1
- Record format
- Record header
- Check sum *2
- File type (3 char.: 6 bytes)
- Main file name (8 char.: 16 bytes)

(b) Data record

(i) When load address is between 0 and 0FFFF (hexadecimal)

S	1	XX	XX	XX	XX	XX	XX	...	XX	XX	LF *3
53	31	XX	XX	XX	XX	XX	XX	...	XX	XX	0A

*4

- Load address (2 bytes)
- Byte count *1
- Record format
- Record header
- Check sum *2
- Data (max 16 bytes)

(ii) When load address is between 10000 and 0FFFFFF (hexadecimal)

S	2	XX	XX	XX	XX	XX	XX	XX	XX	...	XX	XX	LF *3
53	32	XX	XX	XX	XX	XX	XX	XX	XX	...	XX	XX	0A

*4

- Load address (3 bytes)
- Byte count *1
- Record format
- Record header
- Check sum *2
- Data (max 16 bytes)

(iii) When load address is between 1000000 and 0FFFFFFFF (hexadecimal)

S	3	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	...	XX	XX	LF *3
53	33	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	...	XX	XX	0A

*4

- Load address (4 bytes)
- Byte count *1
- Record format
- Record header
- Check sum *2
- Data (max 16 bytes)
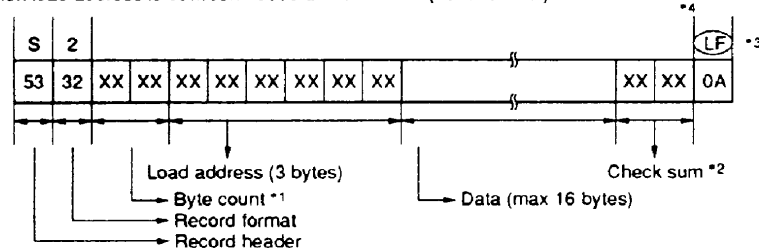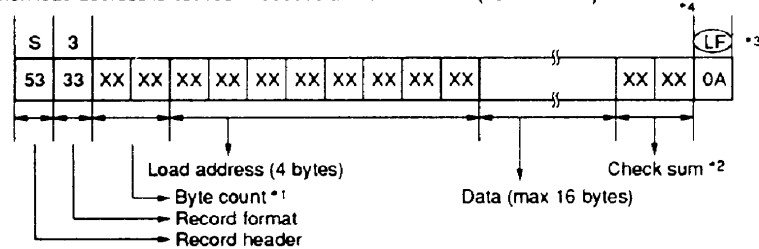
Notes: 1. The byte count is the number of bytes from the load address (or entry address) to check sum.
2. Check sum is the 1's complement of the result of adding the data values from the byte count to that before check sum, in byte units.
3. "LF" indicates the line feed code.
4. In PC system, there is "CR" (0D) before "LF" (0A). "CR" indicates carriage return code.

Figure 9-1 S-Type Object Format

HITACHI

(c) End record

(i) When load address is between 0 and 0FFFF (hexadecimal)
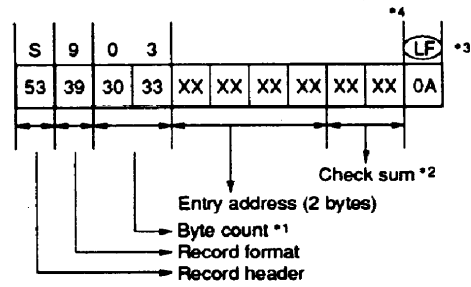
```
 S | 9 | 0 | 3 |                              (LF) *3
53 |39 |30 |33 |XX|XX|XX|XX|XX|XX|0A
```

Check sum *2
Entry address (2 bytes)
Byte count *1
Record format
Record header

(ii) When load address is between 10000 and 0FFFFFF (hexadecimal)

```
 S | 8 | 0 | 4 |                                 (LF) *3
53 |38 |30 |34 |XX|XX|XX|XX|XX|XX|XX|XX|0A
```

Entry address (3 bytes)   Check sum *2
Byte count *1
Record format
Record header

(iii) When load address is between 1000000 and 0FFFFFFFF (hexadecimal)

```
 S | 7 | 0 | 5 |                                    (LF) *3
53 |37 |30 |35 |XX|XX|XX|XX|XX|XX|XX|XX|XX|XX|0A
```
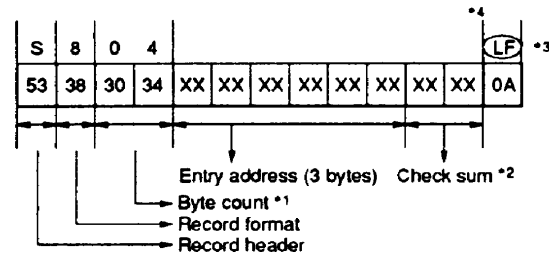
Entry address (4 bytes)   Check sum *2
Byte count *1
Record format
Record header
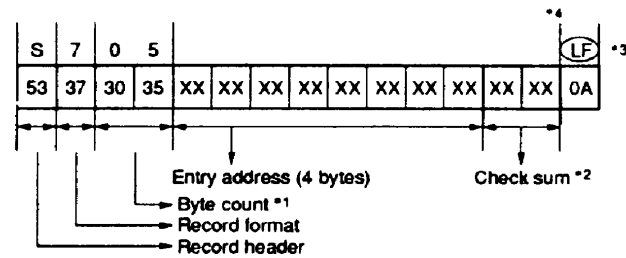
Notes: 1. The byte count is the number of bytes from the load address (or entry address) to the
          check sum.
       2. The check sum is the 1's complement of the sum of the data values from the byte count
          to the byte before the check sum, in byte units.
       3. "LF" indicates the line feed code.
       4. In PC system, there is "CR" (0D) before "LF" (0A). "CR" indicates carriage return code.

Figure 9-1 S-Type Object Format (cont)

HITACHI

9.2 Error Messages

When errors are made in command specification, or when an error is detected during the conversion process, the Object Format Converter outputs error messages in the following format.

```
**Δ <Error number> Δ <error message>[(<additional information>)]
 ▲
 |
 |
1st column
```

A list of error messages is given below in table 9-1 in the following format.

Error Number	Error Message	Additional Information
	Nature of Error	
	Converter actions and corrective actions	

Notation used in table: --: No additional information

HITACHI

Table 9-1 Object Format Converter Error Messages

301	INVALID COMMAND PARAMETER	--
	An improper command parameter was specified.	
	Check the command parameters and re-execute.	
302	FILE NOT FOUND	File name
	The specified file cannot be found.	
	Check the directory and the specified file name, then re-execute.	
303	CANNOT OPEN FILE	File name
	File cannot be opened.	
	Check the specified file name. If the file name is correct, the disk may be full, or there may be a disk hardware problem. After checking the problem, re-execute.	
304	CANNOT READ FILE	File name
	File cannot be input.	
	Check the specified file name. If the file name is correct, there may be a disk hardware problem. After checking the problem, re-execute.	
305	CANNOT WRITE FILE	File name
	File cannot be output.	
	Check the specified file name. If the file name is correct, the disk may be full, or there may be a disk hardware problem. After checking the problem, re-execute.	
306	CANNOT CLOSE FILE	File name
	File cannot be closed.	
	Check the specified file name. If the file name is correct, the disk may be full, or there may be a disk hardware problem. After checking the problem, re-execute.	
307	ILLEGAL FILE FORMAT	File name
	The specified file format is incorrect.	
	Check the file contents, then re-execute.	
308	ILLEGAL FILE NAME	File name
	An illegal file name was specified.	
	Specify a legal file name.	
309	MEMORY OVERFLOW	--
	There is insufficient memory available for use by the Object Format Converter.	
	Expand the memory or revise the user program, then re-execute.	

HITACHI

Appendix A Example of Use of Linkage Editor

In this sample application, the 11 object modules and one library file shown in table A-1 are input into the Linkage Editor.

Table A-1 List of Input Files

No.	File Name	Type of File
1	main.obj	Object module file
2	init.obj	
3	cmndanl.obj	
4	cmndprc.obj	
5	table.obj	
6	term.obj	
7	keyin.obj	
8	file.obj	
9	printer.obj	
10	display.obj	
11	commu.obj	
12	function.lib	Library file

Library file "function.lib" consists of the 14 modules listed in table A-2.

HITACHI

Table A-2 List of Modules in Library File

No.	Module Name
1	mvdata
2	upshft
3	comp
4	expr
5	rmargin
6	lmargin
7	sum
8	number
9	zerosprs
10	ascbin
11	binasc
12	cnvbcd
13	portio
14	dos

Linkage Execution: Input the following command to execute module linkage. In this example, subcommands are input from subcommand file "exlink.sub," and execution is controlled by these subcommands.

lnkΔ-SUBCOMMAND=exlink.sub RET

The contents of subcommand file "exlink.sub" are shown in figure A-1.

HITACHI

```
;
;   First Linkage Process
;
form        r               ; Relocatable Load Module
input       main            ; Input "main.obj"
input       init            ; Input "init.obj"
input       cmndanl         ; Input "cmndanl.obj"
input       cmndprc         ; Input "cmndprc.obj"
input       table           ; Input "table.obj"
input       term            ; Input "term.obj"
library     function        ; Library "function.lib"
output      program1        ; Output "program1.rel"
print       program1        ; Print "program1.map"
end
;
;   Second Linkage Process
;
input       program1.rel    ; Input "program1.rel"
input       keyin           ; Input "keyin.obj"
input       file            ; Input "file.obj"
input       printer         ; Input "printer.obj"
input       display         ; Input "display.obj"
input       commu           ; Input "commu.obj"
library     function        ; Library "function.lib"
                            ; Sequence of Sections
start       program1,program2,function,global,local,f_local,stack_area
output      example         ; Output "example.abs"
print       example         ; Print "example.map"
exit
```

Figure A-1 Subcommand File "exlink.sub"

As figure A-1 shows, two linkage processes are carried out, using the multilinkage function. In the first linkage process, six object module files and the library file are input, and relocatable load module file "program1.rel" and linkage list "program1.map" are output. In the second linkage process, load module file "program1.rel" is re-input, and the remaining object module files are input. The output is absolute load module file "example.abs" and linkage list "example.map."

Linkage list, "program1.map" output in the first linkage process is shown in figure A-2. Linkage list "example.map" output in the second linkage process is shown in figure A-3.

122

HITACHI

```
                            H SERIES LINKAGE EDITOR Ver. 5.0B

     LINK COMMAND LINE

     lnk -subcommand=exlink.sub

     LINK SUBCOMMANDS


     ;
     ;  First Linkage Process
     ;
     form        r              ; Relocatable Load Module
     input       main           ; Input "main.obj"
     input       init           ; Input "init.obj"
     input       cmndanl        ; Input "cmndanl.obj"
     input       cmndprc        ; Input "cmndprc.obj"
     input       table          ; Input "table.obj"
     input       term           ; Input "term.obj"
     library     function       ; Library "function.lib"
     output      program1       ; Output "program1.rel"
     print       program1       ; Print "program1.map"
     end
     ** 105 UNDEFINED EXTERNAL SYMBOL(main.keyin)
     ** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.printer)
     ** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.file)
     ** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.keyin)
     ** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.commu)
     ** 105 UNDEFINED EXTERNAL SYMBOL(cmndprc.display)
     ** 105 UNDEFINED EXTERNAL SYMBOL(term.file)
```

Figure A-2 Linkage List "program1.map" (Input Information)

HITACHI

```
          H SERIES LINKAGE EDITOR Ver. 5.0B                    PAGE: 1

                    ***   LINKAGE EDITOR LINK MAP LIST   ***

SECTION NAME        START      - END          LENGTH
                                              UNIT NAME        MODULE NAME
ATTRIBUTE : CODE NOSHR

program1            H'00000000 - H'00000349  H'0000034a
                                              main             main
                    H'0000034a - H'00000467  H'0000011e
                                              init             initialize
                    H'00000468 - H'0000055d  H'000000f6
                                              cmndanl          command_analize
                    H'0000055e - H'000007e7  H'0000028a
                                              cmndprc          command_process
                    H'000007e8 - H'0000091f  H'00000138
                                              term             terminate
* TOTAL ADDRESS *   H'00000000 - H'0000091f  H'00000920

ATTRIBUTE : DATA NOSHR

local               H'00000000 - H'00001e1f  H'00001e20
                                              main             main
                    H'00001e20 - H'00001e3f  H'00000020
                                              init             initialize
                    H'00001e40 - H'00003c7f  H'00001e40
                                              cmndanl          command_analize
                    H'00003c80 - H'000222bf  H'0001e640
                                              cmndprc          command_process
                    H'000222c0 - H'000222df  H'00000020
                                              term             terminate
* TOTAL ADDRESS *   H'00000000 - H'000222df  H'000222e0

ATTRIBUTE : DATA NOSHR

global              H'00000000 - H'000015cf  H'000015d0
                                              table            global_table
* TOTAL ADDRESS *   H'00000000 - H'000015cf  H'000015d0

ATTRIBUTE : STACK NOSHR

stack_area          H'00000000 - H'001e1fff  H'001e2000
                                              table            global_table
* TOTAL ADDRESS *   H'00000000 - H'001e1fff  H'001e2000
```

Figure A-2 Linkage List "program1.map" (Link Map List)

HITACHI

```
                    H SERIES LINKAGE EDITOR Ver. 5.0B                    PAGE: 2

                        ***    LINKAGE EDITOR LINK MAP LIST    ***

SECTION NAME        START      - END             LENGTH
                                                 UNIT NAME      MODULE NAME

ATTRIBUTE : CODE NOSHR

function            H'00000000 - H'0000001b      H'0000001c
                                        comp                   compare_string
                    H'0000001c - H'0000010f      H'000000f4
                                        expr                   expression
                    H'00000110 - H'00000163      H'00000054
                                        mvdata                 move_data_string
                    H'00000164 - H'00000193      H'00000030
                                        upshft                 upshift_character
* TOTAL ADDRESS * H'00000000 - H'00000193      H'00000194

ATTRIBUTE : DATA NOSHR

f_local             H'00000000 - H'0000000b      H'0000000c
                                        comp                   compare_string
                    H'0000000c - H'0000011b      H'00000110
                                        expr                   expression
                    H'0000011c - H'0000011f      H'00000004
                                        upshft                 upshift_character
* TOTAL ADDRESS * H'00000000 - H'0000011f      H'00000120
```

Figure A-2 Linkage List "program1.map" (Link Map List) (cont)

HITACHI

```
H SERIES LINKAGE EDITOR Ver. 5.0B                          PAGE: 1

***    LINKAGE EDITOR EXTERNALLY DEFINED SYMBOLS LIST    ***

   SYMBOL NAME              ADDR              TYPE

   cmndanl                H'00000000        DAT
   cmndprc                H'00000000        DAT
   cmndtbl                H'000000c8        DAT
   comp                   H'00000000        DAT
   expr                   H'00000000        DAT
   fltbl                  H'000003c8        DAT
   header                 H'00000000        DAT
   init                   H'00000000        DAT
   keybuf                 H'000001c8        DAT
   main                   H'00000000        DAT
   mvdata                 H'00000000        DAT
   prbuf                  H'000014c8        DAT
   recbuf                 H'000013c8        DAT
   stackarea              H'00000000        DAT
   term                   H'00000000        DAT
   upshft                 H'00000000        DAT
```

Figure A-2 Linkage List "program1.map" (Export Symbol List)

HITACHI

```
              H SERIES LINKAGE EDITOR Ver. 5.0B                    PAGE: 1

           *** LINKAGE EDITOR UNRESOLVED EXTERNAL REFERENCE LIST ***

FILE NAME        : main.obj

MODULE NAME      : main
UNIT NAME        : main

                        SYMBOL NAME                      TYPE

                        keyin                            ***

FILE NAME        : cmndprc.obj

MODULE NAME      : command_process
UNIT NAME        : cmndprc

                        SYMBOL NAME                      TYPE

                        commu                            ***
                        display                          ***
                        file                             ***
                        keyin                            ***
                        printer                          ***

FILE NAME        : term.obj

MODULE NAME      : terminate
UNIT NAME        : term

                        SYMBOL NAME                      TYPE

                        file                             ***
```

Figure A-2 Linkage List "program1.map" (Undersolved Import Symbol List)

HITACHI

```
                    H SERIES LINKAGE EDITOR Ver. 5.0B
        LINK COMMAND LINE


        LINK SUBCOMMNADS


        ;
        ; Second Linkage Process
        ;
        input        program1.rel    ; Input "program1.rel"
        input        keyin           ; Input "keyin.obj"
        input        file            ; Input "file.obj"
        input        printer         ; Input "printer.obj"
        input        display         ; Input "display.obj"
        input        commu           ; Input "commu.obj"
        library      function        ; Library "function.lib"
                                     ; Sequence of Sections
        start        program1,program2,function,global,local,f_local,stack_area
        output       example         ; Output "example.abs"
        print        example         ; Print "example.map"
        exit
```

Figure A-3 Linkage List "example.map" (Input Information)

HITACHI

```
                  H SERIES LINKAGE EDITOR Ver. 5.0B              PAGE: 1


                     ***   LINKAGE EDITOR LINK MAP LIST ***


SECTION NAME        START      - END              LENGTH
                                                  UNIT NAME     MODULE NAME
ATTRIBUTE : CODE NOSHR
program1            H'00000000 - H'00000349    H'0000034a
                                               main          program1
                    H'0000034a - H'00000467    H'0000011e
                                               init          program1
                    H'00000468 - H'0000055d    H'000000f6
                                               cmndanl       program1
                    H'0000055e - H'000007e7    H'0000028a
                                               cmndprc       program1
                    H'000007e8 - H'0000091f    H'00000138
                                               term          program1
* TOTAL ADDRESS *   H'00000000 - H'0000091f    H'00000920


ATTRIBUTE : CODE NOSHR
program2            H'00000920 - H'00000b1f    H'00000200
                                               keyin         input_keyboard
                    H'00000b20 - H'00000c47    H'00000128
                                               file          file_io
                    H'00000c48 - H'00000d49    H'00000102
                                               printer       output_printer
                    H'00000d4a - H'00000e61    H'00000118
                                               display       display_console
                    H'00000e62 - H'00001127    H'000002c6
                                               commu         communication
* TOTAL ADDRESS *   H'00000920 - H'00001127    H'00000808


ATTRIBUTE : CODE NOSHR
function            H'00001128 - H'00001143    H'0000001c
                                               comp          program1
                    H'00001144 - H'00001237    H'000000f4
                                               expr          program1
                    H'00001238 - H'0000128b    H'00000054
                                               mvdata        program1
                    H'0000128c - H'000012bb    H'00000030
                                               upshft        program1
                    H'000012bc - H'00001343    H'00000088
                                               lmargin       left_margin
                    H'00001344 - H'00001373    H'00000030
                                               number        numbering_items
                    H'00001374 - H'000013f3    H'00000080
                                               rmargin       right_margin
```

Figure A-3 Linkage List "example.map" (Link Map List)

HITACHI

***   LINKAGE EDITOR LINK MAP LIST   ***

```
SECTION NAME        START       - END            LENGTH
                                                 UNIT NAME    MODULE NAME
ATTRIBUTE : CODE NOSHR
function            H'000013f4 - H'0000140b    H'00000018
                                                 sum          sum_items
                    H'0000140c - H'000014c7    H'000000bc
                                                 zerosprs     zero_suppress
                    H'000014c8 - H'00001533    H'0000006c
                                                 ascbin       ascii_to_binary
                    H'00001534 - H'00001573    H'00000040
                                                 binasc       binary_to_ascii
                    H'00001574 - H'0000163f    H'000000cc
                                                 cnvbcd       convert_to_bcd
                    H'00001640 - H'00001647    H'00000008
                                                 dos          interface_of_dos
                    H'00001648 - H'00001657    H'00000010
                                                 portio       interface_of_port
* TOTAL ADDRESS *   H'00001128 - H'00001657    H'00000530

ATTRIBUTE : DATA NOSHR
global              H'00001658 - H'00002c27    H'000015d0
                                                 table        program1
* TOTAL ADDRESS *   H'00001658 - H'00002c27    H'00015d0

ATTRIBUTE : DATA NOSHR
local               H'00002c28 - H'00004a47    H'00001e20
                                                 main         program1
                    H'00004a48 - H'00004a67    H'00000020
                                                 init         program1
                    H'00004a68 - H'000068a7    H'00001e40
                                                 cmndan1      program1
                    H'000068a8 - H'00024ee7    H'0001e640
                                                 cmndprc      program1
                    H'00024ee8 - H'00024f07    H'00000020
                                                 term         program1
                    H'00024f08 - H'00025127    H'00000220
                                                 keyin        input_keyboard
                    H'00025128 - H'00025307    H'000001e0
                                                 file         file_io
                    H'00025308 - H'0002544b    H'00000144
                                                 printer      output_printer
                    H'0002544c - H'0002554f    H'00000104
                                                 display      display_console
```

Figure A-3 Linkage List "example.map" (Link Map List) (cont)

HITACHI

```
                    H SERIES LINKAGE EDITOR Ver. 5.0B                   PAGE: 3

                       ***   LINKAGE EDITOR LINK MAP LIST   ***

        SECTION NAME         START     - END              LENGTH
                                                          UNIT NAME      MODULE NAME


        ATTRIBUTE : DATA NOSHR
        local               H'00025550 - H'00025713    H'000001c4
                                                          commu          communication
        * TOTAL ADDRESS *   H'00002c28 - H'00025713    H'00022aec


        ATTRIBUTE : DATA NOSHR                      '
        f_local             H'00025714 - H'0002571f    H'0000000c
                                                          comp           program1
                            H'00025720 - H'0002582f    H'00000110
                                                          expr           program1
                            H'00025830 - H'00025833    H'00000004
                                                          upshft         program1
                            H'00025834 - H'00025843    H'00000010
                                                          lmargin        left_margin
                            H'00025844 - H'00025847    H'00000004
                                                          number         numbering_items
                            H'00025848 - H'00025857    H'00000010
                                                          rmargin        right_margin
                            H'00025858 - H'0002587b    H'00000024
                                                          zerosprs       zero_suppress
                            H'0002587c - H'00025883    H'00000008
                                                          ascbin         ascii_to_binary
                            H'00025884 - H'00025887    H'00000004
                                                          binasc         binary_to_ascii
                            H'00025888 - H'000258cf    H'00000048
                                                          cnvbcd         convert_to_bcd
        * TOTAL ADDRESS *   H'00025714 - H'000258cf    H'000001bc


        ATTRIBUTE: STACK NOSHR
        stack_area          H'000258d0 - H'002078cf    H'001e2000
                                                          table          program1
        * TOTAL ADDRESS *   H'000258d0 - H'002078cf    H'001e2000
```

Figure A-3 Linkage List "example.map" (Link Map List) (cont)

HITACHI

```
H SERIES LINKAGE EDITOR Ver. 5.0B                    PAGE: 1

***   LINKAGE EDITOR EXTERNALLY DEFINED SYMBOLS LIST   ***

SYMBOL NAME              ADDR              TYPE

ascbin                  H'000014c8        DAT
binasc                  H'00001534        DAT
cmndanl                 H'00000468        DAT
cmndprc                 H'0000055e        DAT
cmndtbl                 H'00001720        DAT
cnvbcd                  H'00001574        DAT
commu                   H'00000e62        DAT
comp                    H'00001128        DAT
display                 H'00000d4a        DAT
dos                     H'00001640        DAT
expr                    H'00001144        DAT
file                    H'00000b20        DAT
fltbl                   H'00001a20        DAT
header                  H'00001658        DAT
init                    H'0000034a        DAT
keybuf                  H'00001820        DAT
keyin                   H'00000920        DAT
lmargin                 H'000012bc        DAT
main                    H'00000000        DAT
mvdata                  H'00001238        DAT
number                  H'00001344        DAT
portio                  H'00001648        DAT
prbuf                   H'00002b20        DAT
printer                 H'00000c48        DAT
recbuf                  H'00002a20        DAT
rmargin                 H'00001374        DAT
stackarea               H'000258d0        DAT
sum                     H'000013f4        DAT
term                    H'000007e8        DAT
upshft                  H'0000128c        DAT
zerosprs                H'0000140c        DAT
```

Figure A-3   Linkage List "example.map" (Export Symbol List)

Figure A-3 Linkage List "example.map" (Export Symbol List)

HITACHI