

P D P - 1 1  
L I N K - 1 1 L I N K E R  
A N D  
L I B R - 1 1 L I B R A R I A N

Programmer's Manual  
for the  
Disk Operating System

For additional copies, order No. DEC-11-ZLDA-D from Digital Equipment Corporation, Direct Mail, Bldg. 1-1, Maynard, Massachusetts 01754

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Your attention is invited to the last two pages of this document. The "How to Obtain Software Information" page tells you how to keep up-to-date with DEC's software. The "Reader's Comments" page, when filled in and mailed, is beneficial to both you and DEC; all comments received are acknowledged and are considered when documenting subsequent manuals.

Copyright © 1971 by Digital Equipment Corporation

This document is for information purposes and is subject to change without notice.

Associated Documents:

- PDP-11 Disk Operating System Monitor,  
Programmer's Handbook, DEC-11-MWDA-D
- PDP-11 FORTRAN IV,  
Programmer's Manual, DEC-11-KFDA-D
- PDP-11 PAL-11R Assembler,  
Programmer's Manual, DEC-11-ASDB-D
- PDP-11 Edit-11 Text Editor,  
Programmer's Manual, DEC-11-EEDA-D
- PDP-11 ODT-11R Debugging Program,  
Programmer's Manual, DEC-11-OODA-D
- PDP-11 PIP, File Utility Package,  
Programmer's Manual, DEC-11-PIDA-D

The following are trademarks of Digital Equipment Corporation:

DEC	PDP
FLIP CHIP	FOCAL
DIGITAL (logo)	COMPUTER LAB
OMNIBUS	UNIBUS

## PREFACE

This manual describes the features and operation of the Link-11 Linker and the Libr-11 Librarian, both are system programs for the PDP-11 Disk Operating System (DOS). The reader is expected to be familiar with the DOS Monitor and the DOS Assembler; as described in their respective documents listed on the preceding page.

The manual has two major parts:

Part I describes the Link-11 Linker

Part II describes the Libr-11 Librarian

A Master Table of Contents follows this page, and each major part begins with a separate detailed Table of Contents. Chapters are numbered sequentially throughout the manual, and for quick reference purposes the manual is concluded with a comprehensive index.

In addition to the Link-11 Linker and the Libr-11 Librarian, the Disk Operating System software includes:

DOS Monitor  
FORTRAN IV  
PAL-11R Assembler  
Edit-11 Text Editor  
ODT-11R Debugging Program  
PIP, File Utility Package

The following conventions apply to subsequent examples:

1. System program printout is underlined whereas user typed input is not.
2. All command strings are terminated by typing the RETURN key, symbolized as <CR>
3. Elements enclosed in parentheses are optional.



MASTER TABLE OF CONTENTS

PART I	LINK-11 LINKER	
Chapter 1	Introduction	1-1
Chapter 2	Input and Output	2-1
Chapter 3	Operating Procedures	3-1
Chapter 4	Error Handling and Messages	4-1
Chapter 5	Summary of Link-11 Switches	5-1
PART II	LIBR-11 LIBRARIAN	
Chapter 6	Introduction	6-1
Chapter 7	Operating Procedures	7-1



PART I

LINK-11 LINKER

Chapter 1.	Introduction	1-1
1.1	Absolute and Relocatable Program Sections	1-2
1.2	Global Symbols	1-3
1.3	Relinking Link-11	1-3
Chapter 2	Input and Output	2-1
2.1	Input Module	2-1
2.2	Output Module	2-1
2.2.1	Absolute Loader	2-1
2.2.2	DOS Monitor Loader	2-2
2.3	Load Map	2-2
Chapter 3	Operating Procedures	3-1
3.1	Loading	3-1
3.2	Command String	3-1
3.2.1	Switches	3-2
3.2.1.1	Top and Bottom Switches	3-2
3.2.1.2	Undefined Globals Switch	3-3
3.2.1.3	Tapes Switch	3-3
3.2.1.4	Concatenate Switch	3-4
3.2.1.5	ODT Switch	3-4
3.2.1.6	Transfer Address Switch	3-4
3.2.1.7	Exit Switch	3-5
3.2.1.8	Library Switch	3-5
3.2.1.9	General Notes on Switches	3-5
3.3	Library Searches	3-6
3.3.1	User Libraries	3-6
3.3.2	Monitor Library	3-6
3.4	An Example Linking Session	3-7
Chapter 4	Error Handling and Messages	4-1
4.1	Restarting	4-1
4.2	Warning Error Messages	4-1
4.3	Action Request Messages	4-2
4.4	Fatal Linking Error Messages	4-2
Chapter 5	Summary of Link-11 Switches	5-1





## CHAPTER 1

### INTRODUCTION TO LINK-11 LINKER

The PDP-11 Disk Operating System software includes the Link-11 Linker; a system program for linking and relocating user programs assembled by the DOS Assembler. Link-11 enables the user to separately assemble his main program and various subprograms without assigning an absolute address for each segment at assembly time.

The binary output (object module) of each assembly can be processed by Link-11 to:

- Relocate each object module and assign absolute addresses.
- Link the modules by correlating global symbols defined in one module and referenced in another module.
- Produce a load map which displays the assigned absolute addresses.
- Create a load module which can subsequently be loaded (by the Monitor or the Absolute Loader) and executed.

The advantages of using Link-11 include:

- The source program can be divided into segments (usually subroutines) and assembled separately. If an error is discovered in one segment, only that segment needs to be reassembled. Link-11 can then link the newly assembled object module with other object modules.
- Absolute addresses need not be assigned at assembly time; the Linker automatically assigns absolute addresses. This keeps programs from overlaying each other. This also allows subroutines to change size without influencing the placement of other routines.
- Separate assemblies allow the total number of symbols to exceed the number allowed in a single assembly.
- Internal symbols (which are not global) need not be unique among object modules. Thus, naming rules are required for global symbols only when different programmers prepare separate subroutines for a single program.

- Subroutines may be provided for general use in object module form to be linked into the user's program.

Link-11 is designed to run on an 8K PDP-11 with a disk and a teleprinter. DECTape, high-speed paper tape reader and punch and a line printer may be used if available. The DECTape and the high-speed reader/punch significantly speed up the linking process. A line printer provides a fast display device for the load map listing.

The instructions and assembler directives of the PAL-11R Assembler are used herein to describe the operation of Link-11. However, Link-11 can link and relocate any object modules which conform to its linking format (consult your nearest DEC representative for the linking format).

### 1.1 ABSOLUTE AND RELOCATABLE PROGRAM SECTIONS

A program assembled by PAL-11R can consist of an absolute program section, declared by the .ASECT assembler directive, and relocatable program sections, declared by the .CSECT assembler directive; without either the assembler assumes a .CSECT directive. The program and data in a relocatable section are assigned absolute addresses by the Linker such that the relocatable section is normally at the high end of memory. The assignment of addresses can be influenced by command string options (see Section 3.2). The Linker appropriately modifies all instructions and/or data as necessary to account for the relocation of the control section (as declared by .CSECT).

Link-11 handles the absolute section, as well as the named and unnamed control sections. The unnamed control section is internal to each object module. That is, every object module can have an unnamed control section, but the Linker treats them independently. Each is assigned an absolute address such that they occupy mutually exclusive areas of memory. Named control sections, on the other hand, are treated globally. That is, if different object modules each have control sections with the same name, they are all assigned the same absolute load address and the size of the area reserved for loading of the section is the size of the largest section. Thus, named control sections allow the sharing of data and/or programs among object modules. This is similar to the handling and function of COMMON in FORTRAN IV. The names assigned to control sections are global and can be referenced as any other global symbol.

## 1.2 GLOBAL SYMBOLS

Global symbols provide the links, or communication, between object modules. With PAL-11R, these symbols would be created with the .GLOBL assembler directive. Symbols which are not global are called internal symbols. If the global symbol is defined (as a label or by direct assignment) in an object module, it is called an entry symbol and other object modules can reference it. If the global symbol is not defined in the object module, it is an external symbol and is assumed to be defined (as an entry symbol) in some other object module.

As the Linker reads the object modules it keeps track of all global symbol definitions and references. It then modifies the instructions and/or data which reference the global symbols.

## 1.3 RELINKING LINK-11

Link-11 is provided as a system program with the PDP-11 Disk Operating System.

It is available from DEC in the following formats:

- Absolute load module for 8K systems
- Object module for relinking
- ASCII source tapes

These enable you to relink Link-11 using the supplied Link-11 load module to load into any 8K or larger system. The resulting Linker will assume a top of memory corresponding to the system configuration; this can be over-ridden using the T (top) or B (bottom) switches (see Section 3.2.1.1).

The top address assumed by the Linker is:

nnn460

where nnn is explained below.

<u>nnn</u>	<u>for Memory Size</u>
37	8K
57	12K
77	16K
117	20K
137	24K
157	28K



## CHAPTER 2

### INPUT AND OUTPUT

#### 2.1 INPUT MODULE

Link-11's input is the object module. This is the output of the PAL-11R Assembler or any assembler or compiler that can produce an object module. The Linker reads each object module twice, thus, it is a two-pass processor.

On pass 1, the Linker reads each object module to gather enough information so that absolute addresses can be assigned to all relocatable sections and all globals can be assigned absolute values. This information appears in the global symbol directory (GSD) of the object module.

On pass 2, the Linker reads all of each object module and produces the load module. The data gathered on pass 1 guides the relocation and linking process on pass 2.

#### 2.2 OUTPUT MODULE

The normal output of the Linker is a load module which can be loaded and run. A load module consists of formatted binary blocks of absolute load addresses and object data as specified for the Absolute Loader and the DOS Monitor Loader. The first few words of data will be the communications directory (COMD), and will have an absolute load address equal to the lowest relocated address of the program.

##### 2.2.1 Absolute Loader

The Absolute Loader will load the COMD at the specified address, but then the program will overlay the COMD. The overlaying of the COMD by the relocated program allows the Absolute Loader to handle load modules with a COMD. However, a problem arises if a load module is to be loaded by the Absolute Loader and either of the following conditions exist:

1. The object modules used to construct the load module contain no relocatable code, or
2. The total size of the relocatable code is less than the size of the COMD.

In either case, there would not be enough relocatable code to overlay the COMD, which means that the COMD will load into parts of core not intended to be altered by the user. The Linker will select the COMD's load address such that the COMD will be against the current top (see T switch in Section 3.2). If the top is very low, the Linker will not allow the COMD to be loaded below address 0; it will load it at 0.

### 2.2.2 DOS Monitor Loader

The DOS Monitor Loader will load the COMD where the Monitor wants it. The end of the load module will be indicated by a TRA (transfer address) block; that is, a block containing only a load address. The byte count in the formatted binary block will be 6 on this block; on all other blocks the byte count will be greater than 6. If the TRA is not specified by a switch, it is assumed by the Linker to be the first even transfer address encountered. Thus, if four object modules are linked together and if the first and second have a .END statement without an address, the third a .END A, and the fourth a .END B, the transfer address would be A of module three.

### 2.3 LOAD MAP

The load map produced by Link-11 provides several types of information concerning the load module's make-up. The map begins with the transfer address and the low and high limits of the relocatable code. Then there is a section of the map for each object module included in the linking process. Each of these sections begins with the module's name followed by a list of the control sections and the entry points for each control section. For each control section, the base of the section (its low address) and its size (in bytes) is printed to the right of the section name (enclosed in angle brackets). Following each section name is an alphabetically ordered list of entry points and their addresses. A list of any undefined symbols for each object module concludes the load map.

Note that modules are loaded such that if modules A, B, and C are linked together, A is lowest and C is highest in core.

A sample load map is shown below.

LOAD MAP

TRANSFER ADDRESS: 037434

LOW LIMIT: 037406

HIGH LIMIT: 037460

\*\*\*\*\*

MODULE	MOD1	SECTION	ENTRY	ADDRESS	SIZE
		.	ABS.	000000	000000
				037406	000044
			X3	037430	
			X4	037440	
			X5	037452	
			X7	037450	

\*\*\*\*\*

MODULE	MOD2	SECTION	ENTRY	ADDRESS	SIZE
				037452	000006
			X1	037452	
			X2	037452	

\*\*\*\*\*

\*\*\*\*\*

UNDEFINED REFERENCES

X6





## CHAPTER 3

### OPERATING PROCEDURES

#### 3.1 LOADING

Link-11 is loaded into core by typing the following Monitor command.

```
.RUN LINK11 <CR >
```

#### NOTE

In the examples, typing the RETURN, LINE FEED, and SPACE keys are shown as <CR>, <LF>, and <SPACE>, respectively. Also, in the examples, program print-out is underlined whereas user typed input is not.

When the Linker is in core and ready to accept the user's command, it prints the following three lines:

```
LINK-11 V002A  
PASS1  
#
```

The user can now type a command string after the # sign.

#### 3.2 COMMAND STRING

Commands are typed in response to the number sign, #, printed by the Linker. The format of the command string adheres to the requirements of the DOS Command String Interpreter (CSI), as explained in the Disk Operating System Monitor, Programmer's Handbook.

The Linker's file specifications must appear in the following order:

```
#load module output, map output < object modules <CR>
```

A null specification field signifies that the associated output is not desired. A complete specification contains the following information:

```
dev:filnam.ext[uc]/s1:v/s2:v.../sn:v <CR>
```

The default values for each specification are noted below.

	<u>dev</u>	<u>filnam</u>	<u>ext</u>	<u>uic</u>
Load Module	DFØ	**	LDA	This user
Map Output	DFØ*	none	MAP	This user
Object Module	DFØ*	none	OBJ	This user

\*or last device specified on this side of the <  
 \*\*the filename from the first input specification

If a syntactical error is detected in a command string, the Linker will print the command on the teleprinter up to and including the character in error, followed by a question mark, and then a line beginning with the input request character #. The user can then type the command correctly. Link-11 performs this error reporting function on pass 1 only.

### 3.2.1 Switches

There are nine switches associated with the Linker:

/T	Top
/B	Bottom
/U	Undefined globals
/OD	ODT
/TA	Tapes
/CC	Concatenate
/TR	Transfer address
/E	Exit
/L	Library

The letter(s) representing each switch is always preceded by the slash symbol. Switches are not allowed in the output fields of a command string.

#### 3.2.1.1 Top and Bottom Switches

The T and B switches are used to control the placement or relocation of the object program. When neither switch is specified, Link-11 will link the object programs to the top of available core, i.e., immediately below the Absolute and Boot Loaders.

The T switch (top) can be specified with any of the input file specifications. It must be in the following format:

/T:n

where n is an unsigned octal number which defines a new top address. If a bottom is specified, the top switch is ignored.

The B switch (bottom) can be specified with any of the input file specifications. It must be in the following format:

/B:n

where n is an unsigned octal number which defines the bottom address of the object program. The B switch causes a top address to be calculated so that the lowest address in the program will be at location n. If a top is specified, the bottom switch is ignored.

Once a top of core has been calculated with the T or B switch, that value is used until it is changed. Only one T or B switch can be used during any one linking process.

#### CAUTION

The top or bottom value must be an unsigned even octal number. Link-11 detects if the value is odd and gives an error message.

#### 3.2.1.2 Undefined Globals Switch

The U switch is used to obtain a teleprinter listing of all globals still found to be undefined at the time the switch is used. The listed globals will apply only to those files specified prior to the U switch request, including the specification in which the U itself appears.

The U switch can be specified with any or all input file specifications. Its format is:

/U

This switch does not require a value.

#### 3.2.1.3 Tapes Switch

The TA switch is used to specify the number of tapes to be read in during the linking process. Its format is:

/TA:n

where n is an unsigned decimal number, the number of the tapes to be read. If n is too large, the user must load some dummy modules or put a blank tape in the reader for each extra request on both pass 1 and pass 2.

In response to this switch, the Linker requests the DOS Monitor to read n tapes from the specified input device. The Monitor will print:

```
A002    063320
$
```

whenever the device is not ready. The message A002 means that device 063320 (high-speed paper tape reader, represented in Radix-50) is not ready.

The user should place the next tape in the reader and type CO<CR> in response to the \$ printed by the Monitor to continue the linking process. The procedure above is repeated for each tape to be linked.

After pass 1, the Linker will print PASS 2 and the procedure above is repeated, which produces the load module. After completing pass 2, control remains in the Linker for more link requests.

#### 3.2.1.4 Concatenate Switch

The CC switch is used to indicate that the file was formed (by PIP) as a concatenation of several object modules. This switch must be placed with an input file specification. Its format is:

```
/CC
```

This switch does not have a value.

#### 3.2.1.5 ODT Switch

The OD switch is used to link ODT with your object modules. It must appear with ODT's input file specification, in the following format:

```
ODT.OBJ/OD
```

which would identify the file as ODT, for transfer address purposes.

#### 3.2.1.6 Transfer Address Switch

The TR switch can appear with any input file specification. It can be used with no value, or with an octal number or global symbol as its value.

When the TR switch has no value, it indicates that the Linker should take the transfer address (even or odd) of the first object module in the file as the transfer address of the load module. Its format is:

/TR

When it has an octal number as its value, it indicates that the value is the transfer address of the load module. Its format is:

/TR:n

When it has a global symbol as its value, it indicates that the value of the global symbol is the transfer address of the load module. Its format is:

/TR:xxxxxx

When the value is a nonexistent symbol, the transfer address is set to 1.

#### 3.2.1.7 Exit Switch

The E switch should appear with the last input file specification. It indicates the end of input.

#### 3.2.1.8 Library Switch

The L switch is used to indicate that the file is a library. It must appear in an input file specification, if the specification contains a user library. The L switch has no value. Its format is:

/L

#### 3.2.1.9 General Notes on Switches

If a switch appears by itself as a specification (e.g., , ,/CC), it takes the default device and a file  $\emptyset$  file name. Thus, the linking process will be aborted if the default device requires a file.

There are thirty words allowed for switches per input specification. If more are requested, it will result in a S203 error message. The following conditions also result in a S203 error message:

- a. If a switch requires a value and if none appears or more than one appear.
- b. If a switch does not require a value and some value is given.

Leader and trailer are punched on the load module when the output is to paper tape. The low-speed reader (LSR) and low-speed punch (LSP), although able to be used by the Linker, are not serviced by the DOS for binary tapes. Therefore, they are considered illegal devices.

A comprehensive summary of all switches appears in Chapter 5.

### 3.3 LIBRARY SEARCHES

#### 3.3.1 User Libraries

Object modules from the named user libraries built by the Libr-11 Librarian will be relocated and linked by the Linker. The object modules in the libraries have to be ordered; only forward references are allowed.

The libraries are input to the Linker as any other input file. The L switch in the input file specification indicates that this particular file is a library. It has the following format:

```
dev:libnam.ext/L
```

For example, the user could type the following command string to the Linker:

```
#TASKØ1.LDA,LP:<MAIN,MEASUR.LIB/L/E
```

Program MAIN.OBJ would be read in from the disk as the first input file. Any undefined symbols generated by program MAIN.OBJ can be satisfied by the library MEASUR.LIB specified in the second input file. The load module, TASKØ1.LDA would be put on the disk, and a load map would go to the line printer.

#### 3.3.2 Monitor Library

At the end of pass 1, the Monitor library is searched for Monitor routines which were declared as globals in the user program. Satisfying these globals means that the Linker passes the EMT trap numbers of the found routines (in the COMD) to the Monitor so that at load time the Monitor brings the requested routines into core with the user program.

The user libraries are searched first and the Monitor library is searched last.

### 3.4 AN EXAMPLE LINKING SESSION

The Linker is called into core, identifies itself and prints # as explained in Section 3.1. The user could then type the first command string. For example:

```
#PP:,KB:<PR:<CR>
```

which means that he wants the load module to be punched on the high-speed paper tape punch, the load map to be printed on the teleprinter, and his first object module to be read in from the high-speed paper tape reader.

When the first tape is read in and no more file specifications follow, and when no E (Exit) switch has been seen by the Linker, the Linker asks for the next command string by printing another #. The user could now type:

```
#PR:/E<CR>
```

The Linker would then read in the second input tape, encounter the Exit switch, and exit pass 1. The load map would then be printed on the teleprinter.

When the Linker finishes printing the load map, it enters the second pass automatically to read in again the first tape from the paper tape reader. Before putting in the first tape again, wait until the Monitor prints:

```
A002    063320  
$
```

where A002 means device 063320 (high-speed paper tape reader represented in Radix-50) is not ready. The user then places the first tape in the reader, and in response to the \$ printed by the Monitor he types:

```
CO <CR>
```

and the Linker starts processing the first tape. It then starts punching out the load module. When finished reading the first tape, the Monitor prints:

A002      063320  
\$

which again means that the paper tape reader is not ready.

The user then places the second tape in the reader, and in response to the \$ printed by the Monitor, he types:

CO <CR>

and the Linker starts processing the second tape.

When the Linker finishes linking the two input tapes into one load module, it restarts itself and prints on the teleprinter:

LINK-11   V002A  
PASS 1  
#

and waits for another command string.



## CHAPTER 4

### ERROR HANDLING AND MESSAGES

#### 4.1 RESTARTING

The user can restart Link-11 at any time by typing

```
CTRL/C  RESTART  or  CTRL/C  BEGIN
```

which causes Link-11 to re-identify itself and print #.

If the above sequence of keys (CTRL/C, RE(START) command, RETURN key) is typed while a load map is being printed, the load map will be aborted and the Linker will continue.

#### 4.2 WARNING ERROR MESSAGES

The following three types of warnings are printed by Link-11.

a. Non-unique object module name. This error is detected during pass 1 and results in the message:

```
XXXXXX  
W300  0000000
```

where XXXXXX is the non-unique object module name. The module is rejected. The Linker will then continue processing the remaining unsorted file specifications.

b. A byte relocation error. The Linker will try to relocate and link byte quantities. Failure is defined as the high byte of the relocated value (or the linked value) not being all zero. In such a case, the value is truncated to 8 bits and the following message is printed:

```
W301  XXXXXX
```

where XXXXXX is the absolute address where the byte relocation error occurred. The Linker automatically continues.

c. Multiple-defined globals. This results during pass 1, causing the error message:

```
XXXXXX,YYYYYY  
W302  0000000
```

where YYYYYY is the symbol which is multiple defined by object module XXXXXX. The second definition is ignored and the Linker continues.

d. Undefined globals in the load module. This results at the end of pass 1, causing the error message:

```
W322  0000000
```

meaning that there were some global symbols left undefined at the end of pass 1. The Linker then continues.

#### 4.3 ACTION REQUEST MESSAGES

If the object modules are not read in from paper tape in the same order on pass 2 as on pass 1, the Linker will indicate which module should be loaded next by printing:

```
XXXXXX  
A006  0000000
```

meaning, load object module XXXXXX next.

When the message appears on the teleprinter, the Linker halts the creation of the load module temporarily. The tape which is in the paper tape reader is ignored. The Linker waits for the user to remove the incorrect tape and place the correct tape under the reader.

When the user is done he types CO<CR> and the Linker reads in the tape and resumes the creation of the load module.

If input comes from a file-structured device, this action request does not apply.

#### 4.4 FATAL LINKING ERROR MESSAGES

When Link-11 detects a fatal error condition during the linking process, it prints an appropriate error message and then restarts itself. Fatal error messages and their meanings are listed below.

Error Message  
and Format

Meaning

S202	xxxxxx dev:file.ext	EOD or device error on .WRITE or .READ request. xxxxxx = error status byte.
S203	000000	Illegal switch, or too many switches, or illegal switch value, or switch value not given, or switch in an output field.
S204	000000	Too many output files.
S206	000000	Input file not specified in CSI line.
S207	xxxxxx	EOD or device error on .TRAN request. xxxxxx = error status byte.
S210	000000 dev:file.ext	Unrecognized symbol table entry in indicated file.
S211	000000 dev:file.ext	In indicated file, a RLD references a global name which cannot be found in the symbol table.
S212	000000 dev:file.ext	In indicated file, a RLD contains a location counter modification command which is not last.
S213	000000 dev:file.ext	In indicated file, an object module does not start with a GSD.
S214	000000 dev:file.ext	In indicated file, the first entry in a GSD is not the module name.
S215	000000 dev:file.ext	In indicated file, a RLD references a section name that cannot be found.
S216	000000	The TRA specification references a nonexistent module name.
S217	000000	The TRA specification references a nonexistent section name.
S220	000000	An internal jump table index is out of range.
module,symbol S223	000000	Symbol table overflow. Insufficient space in symbol table for indicated symbol of indicated object module.
S223	000000	No more space for CSI input buffer, or Monitor's file manager routine, or for Monitor's library search buffer.
S225	000000	An angle bracket (<) in CSI line, which is not the first.
S243	000000	Angle bracket (<) is missing from the first CSI line.



## CHAPTER 5

## SUMMARY OF LINK-11 SWITCHES

<u>Name</u>	<u>Symbol</u>	<u>Value</u>	<u>Format</u>	<u>Function</u>
Top	T	n (octal number)	/T:n	n becomes top of core
Bottom	B	n (octal number)	/B:n	n becomes bottom of core
Undefined Globals	U	none	/U	List undefined globals on teleprinter
ODT	OD	none	/OD	This file is ODT
Tapes	TA	n (decimal number)	/TA:n	There are n tapes to be read
Concatenate	CC	none	/CC	This file contains concatenated object modules
Transfer Address	TR	none	/TR	Take the transfer address of the first object module of this file as the transfer address
	TR	n (octal number)	/TR:n	Take the octal number as the transfer address
	TR	global symbol	/TR:xxxxxx	Take the value of the global symbol as the transfer address
Library	L	none	/L	This file is a library
Exit	E	none	/E	Exit from current pass; end of linking



PART II

LIBR-11 LIBRARIAN

Chapter 6.	Introduction	6-1
Chapter 7	Operating Procedures	7-1
7.1	Calling Libr-11	7-1
7.2	Command Strings	7-1
7.2.1	Creating a Library	7-1
7.2.2	Updating a Library	7-2
7.2.2.1	To Delete One or More Object Modules	7-2
7.2.2.2	To Insert One or More Object Modules	7-2
7.2.2.3	To Replace One or More Object Modules	7-3
7.2.3	Listing a Library	7-3
7.2.4	Legal File Specification Combinations	7-4
7.2.5	Duplicate Library Names	7-5
7.3	Examples	7-5
7.4	Error Messages	7-8





## CHAPTER 6

### INTRODUCTION TO LIBR-11 LIBRARIAN

The PDP-11 Librarian (Libr-11) is a system program for the Disk Operating System. Libr-11 provides facilities for creating, modifying, deleting, and listing the contents of libraries. A library is a file which consists of one or more object modules. (An object module is the binary output of the PAL-11R Assembler.)

Libr-11 is a valuable program for the DOS user because:

- It eliminates having separate directory entries in a User File Directory (UFD) for each object module.
- It expedites the linking process in conjunction with the Linker's library search capabilities.
- It allows for standardization and controlled updating of frequently used routines, e.g., FORTRAN cosine routine.

The user controls the operation of Libr-11 through command strings typed on the keyboard. Specified in the command strings are such things as devices, library and object module names, and switches which indicate the Libr-11 operation desired. The user can direct Libr-11 to:

- Create a library
- Update a library
- Insert one or more object modules in a library
- Replace one or more object modules in a library
- List the contents of a library
- Delete one or more object modules from a library
- Delete an entire library

A directory listing of the object modules of a library can be obtained merely by specifying the device on which the directory is to appear and the name of the library.

The flexibility of Libr-11 enables the user to specify certain combinations of operations in a single command string. For example,

a library can be modified, renamed, and listed in one command string.

The switch options which direct Libr-11's operations are:

<u>Switch</u>	<u>Operation</u>
/D	Delete object module
/DL	Delete input library
/I	Insert object module
/LO	List object modules
/R	Replace object module

If you type an illegal command string, e.g., illegal format, excessive switches, nonexistent file or object module, etc., Libr-11 will print an appropriate error message on the teleprinter.

The following discussion assumes that the reader is familiar with the DOS Monitor, Edit-11 Text Editor, PAL-11R Assembler, ODT-11R Debugging Program, and Link-11 Linker.

## CHAPTER 7

### OPERATING PROCEDURES, LIBR-11

#### 7.1 CALLING LIBR-11

The Librarian is called into core by typing the RUN command in response to the DOS Monitor's dot or dollar sign. (The Librarian's call name can be determined by listing the system directory using PIP.) The Librarian is often stored as LIBR, and when called it prints its name, version number, and a # sign, and then waits for the user to issue a command string. For example:

```
.RUN LIBR      <CR>
LIBR-11      V001A
#
```

#### 7.2 COMMAND STRINGS

When the Librarian is in core and has printed the # sign, it is ready to accept a user command string. The format is:

```
output library, listing file < input library, input file(s)
```

Libr-11 performs two passes over all input files. For nonfile-structured devices (e.g., paper tape reader), the system will inform the user to reload the device for the second pass. For file-structured devices, both passes are performed automatically without requiring any user intervention.

##### 7.2.1 Creating a Library

```
output library (,listing file) < ,input file(s)
```

A library is created on the device specified in the output library specification and named as specified. The listing file specification is optional and, if present, the contents of the output library will be listed. The format of the listing will be fully discussed later.

An input library need not appear, but the comma and one or more input files must appear (each of which contains one or more object modules\*). For example:

```
#DT1:FIL.LIB< ,FIL.1,FIL.2      <CR>
```

\*Note that an input file of concatenated object modules differs from a library in that it does not have a directory of the object modules that it contains.

would create a library named FIL.LIB on DECTape 1. The library would consist of all object modules in FIL.1 and FIL.2 in that order, and in the order in which the object modules appear in their respective input files.

### 7.2.2 Updating a Library

Libraries can be updated in one of three ways:

#### 7.2.2.1 To Delete One or More Object Modules

```
output library (,listing file) < input library/D:v1:...vn
```

The output library will be created as a result of deleting the object modules named v<sub>1</sub>...v<sub>n</sub> from the input library. The listing file is optional.

The name associated with an object module is the symbol assigned to the module by PAL-11R's .TITLE assembler directive.

The object modules to be deleted must appear in the same order as they appear in the library; their order can be determined from the listing.

Insert and/or Replace operations cannot accompany a Delete request. For example:

```
#DT1:LIBR.1 < DT2:LIBR∅/D:M1:M2 <CR >
```

would create a library named LIBR.1 on DECTape 1 as a result of deleting the object modules M1 and M2 from LIBR.∅ on DECTape 2.

#### 7.2.2.2 To Insert One or More Object Modules

```
output library (,listing file)<input library,input file(s)/I(:v)
```

The output library will be created as a result of inserting the object modules of the input file into the input library. If v is specified, the object module(s) in the input file are inserted starting at position v, otherwise, they are inserted at the end. v is treated as a decimal integer.

If more than one input file is specified for insertion, the positions at which they are to be inserted must appear in non-descending

order. For example:

```
#DT1:LIBR.1<DT2:LIBR.Ø,FIL.1/I:2,FIL.2/I <CR>
```

would create an output library on DECTape 1 as a result of inserting the object modules of FIL.1 into LIBR.Ø, beginning at position 2, and then inserting the object modules of FIL.2 into LIBR.Ø at the end.

Insert and Replace operations can appear in the same command so long as the order restriction is observed.

### 7.2.2.3 To Replace One or More Object Modules

```
output library (,listing file)<input library,input file(s)/R
```

The output library will be created as a result of replacing the object module(s) in the input library by those in the input file(s).

The object modules to be replaced must have the same name as those which replace them, and they must correspond orderwise. For example:

```
#DT1:LIBR.1<DT2:LIBR.Ø,FIL.1/R,FIL.2/R <CR>
```

would create the output library LIBR.1 on DECTape 1 as a result of replacing the object modules in the input library LIBR.Ø with those in FIL.1 and FIL.2.

### 7.2.3 Listing a Library

```
,listing file(/LO)<input library
```

The directory of the input library will be listed. Optionally, the presence of the /LO switch directs the Librarian to produce an object module listing. This is intended as a means to double-check the accuracy of the library; the directory listing must correspond exactly to the object module listing.

The output library will be listed when one was created; otherwise, the input library will be listed. The format of the listing is:

Library Name & Extension

Decimal Order Number	Object Module Name (1st Module)
⋮	⋮
Decimal Order Number	Object Module Name (last module)

For example, if LIB.1 contains object modules M1, M2, and M3 in that order, the command:

```
#,LP:FIL.LST<DT1:LIB.1      <CR>
```

will produce on the line printer:

```
FIL      .LST
000001    M1
000002    M2
000003    M3
```

If the /LO switch appears, for example:

```
#,LP:FIL.LST/LO<DT1:LIB.1    <CR>
```

the listing above would be followed by a form feed and a similar table, except that the name of the second table is always OBJMOD.LST.

The library name that is printed at the head of the listing is the name specified in the listing file specification. For example:

```
#LIB.ABC,LP:NAME<,FIL.1,FIL.2  <CR>
```

The listing would be titled NAME, not the new created library LIB.ABC. When the listing file name is not specified, then the listing would be titled with the name of the newly created file.

#### 7.2.4 Naming Libraries

The output library can have the same name as the input library. In this case, however, the input library has an implied /DL; that is, the input library is deleted. For example:

```
#LIB.1<LIB.1/D:OM1          <CR>
```

is the same as:

```
#LIB.TMP<LIB.1/D:OM1/DL     <CR>
```

and then rename LIB.TMP to LIB.1.

### CAUTION

The user should never name a Library LIBR.TMP. This name is reserved for use by the Librarian.

#### 7.2.5 Legal File Specification Combinations

In a command string, various combinations of file specifications are possible; legal combinations and their operation are shown below.

	<u>Output Library</u>	<u>Listing</u>	<u>Input Library</u>	<u>Input File(s)</u>	<u>Operation</u>	<u>Note</u>
(1)	P	P	P	P	Insert or Replace Object Modules; List Output Library	SE if /D on input library
(2)	P	P	P	NP	Delete Object Modules; List Output Library	SE if /D not on input library
(3)	P	P	NP	P	Create Library; List Output Library	SE if switch on input file
(4)	NP	P	P	NP	List Input Library	SE if /D on input library
(5)	P	NP	P	P	Same as (1) except no listing	Same as (1)
(6)	P	NP	P	NP	Same as (2) except no listing	Same as (2)
(7)	P	NP	NP	P	Same as (3) except no listing	Same as (3)

Legend: P = present  
NP = not present  
SE = syntax error

#### 7.3 EXAMPLES

Assume FIL.1 contains object modules OM1, OM2, and OM3 in that order, FIL.2 contains OM4 and OM5 in that order, FIL.3 contains OM5 and OM3 in that order, and FIL.4 contains OM6. Then:

```
#LIB.1,LP:LIB.1<,FIL.1,FIL.2
```

will create a library named LIB.1 containing object modules OM1, OM2, OM3, OM4, and OM5 in that order. The listing will appear on the line printer as:

```
LIB      .1
ØØØØ1   OM1
ØØØØ2   OM2
ØØØØ3   OM3
ØØØØ4   OM4
ØØØØ5   OM5
```

Files FIL.1 and FIL.2 remain unaltered. The listing is produced after all other actions have been performed. Consequently,

```
#LIB.1,LIB.1 <,FIL.1,FIL.2      <CR>
```

would produce an error message (file already exists) when an attempt is made to write the listing to the disk.

Using the assumption above:

```
#LIB.2<LIB.1/D:OM1:OM4      <CR>
```

will create a library named LIB.2 containing object modules OM2, OM3, and OM5 in that order. No listing is produced and LIB.1 is not deleted.

```
#LIB.3<LIB.2/D:OM3:OM2      <CR>
```

will produce an error message because the modules to be deleted are not in the order in which they appear in the library.

The command string:

```
#,LP:LIB2.LS/LO<LIB.2      <CR>
```

will produce a listing on the line printer which appears as:

```
LIB2      .LS
ØØØØ1     OM2
ØØØØ2     OM3
ØØØØ3     OM5
<form feed>
OBJMOD.LST

ØØØØ1     OM2
ØØØØ2     OM3
ØØØØ3     OM5
```



The command string:

```
#LIB.3<LIB.2/DL,FIL.4/I:2    <CR>
```

will create a library named LIB.3 containing OM2, OM6, OM3 and OM5 in that order. No listing is produced and LIB.2 is deleted.

The command string:

```
#LIB.4<LIB.3,FIL.4/R    <CR>
```

will create a library named LIB.4, which is really LIB.3 with OM6 replaced (i.e., removed from LIB.3 before creating LIB.4).

```
#LIB.5<LIB.4,FIL.3/R    <CR>
```

will produce an error message because the object modules in FIL.3 are not in the same order as in LIB.4.

The command string:

```
#LIB.5<LIB.3/DL,FIL.4/I    <CR>
```

will create a library named LIB.5 containing OM2, OM6, OM3, OM5 and OM6 in that order. No listing is produced and LIB.3 is deleted. Note that a library can contain multiple copies of the same object module, e.g., two OM6 modules, above.

The command string:

```
#LIB.6<LIB.5/D:OM6    <CR>
```

will create a library named LIB.6 containing OM2, OM3, OM5 and OM6 in that order. No listing is produced and LIB.5 is not deleted. When a library contains multiple copies of the same object module, they are deleted one at a time in their order of occurrence.

If the purpose of the previous example were to delete all occurrences of OM6, the command string would have been either:

```
#LIB.6<LIB.5/D:OM6:OM6    <CR>
```

or

#LIB.6<LIB.5/D:OM6/D:OM6 <CR>

#### 7.4 ERROR MESSAGES

Error messages issued by the Librarian are listed below. (See the DOS Monitor document for a listing of all DOS error messages.)

<u>Error Code</u>	<u>Additional Information</u>	<u>Meaning</u>
S202	File Name and Error Status Byte	Fatal I/O error; due to truncated line, checksum, character parity, or device parity error.
S203	File Name	Switch error or semantic error; due to illegal switch, too many switches on a file, or illegal combination of file specifications.
S204		Illegal file specification format; more than two output files specified.
S213	File Name	Error on input file; illegal object module format; first line not a GSD, or EOF prior to reading end module line.
S244	File Name	Out of order; already past requested position for Insert.
S245	File Name	Object module error; object module not found, or /R or /D out of order.
S246		Error on input library; illegal library format, first two lines incorrect.
S247		Listing error; output library cannot be read from output library device, i.e., PP:.

## INDEX

- Absolute program sections, 1-2
- Absolute loader, 2-1
- Action request messages, Linker, 4-2
- Address assignment, 1-2
- Advantages, Link-11, 1-1
- Angle brackets ( <> ), writing convention, 3-1
- Assembler directive
  - .ASECT, 1-2
  - .CSECT, 1-2
  - .TITLE, 7-2
  
- Bottom (B) switches, 1-3, 3-2
  
- Calling Libr-11, 7-1
- Command string
  - Libr-11, 7-1
  - Linker, 3-1
- Communications directory (COMD), 2-1
- Concatenate (CC) switch, 3-4
- Control section, named or unnamed, 1-2
- <CR> symbol, writing convention, 6-2
- Creating a library, 7-1, 7-5
- .CSECT assembler directive, 1-2
- CTRL/C BEGIN, 4-1
- CTRL/C RESTART, 4-1
  
- Deleting object modules, 7-2
- Deletion error, 7-6
- DOS assembler, 1-1
- DOS Monitor loader, 2-2
  
- .END statement, 2-2
- Entry symbols, 1-3
- Error handling, Link-11, 4-1
  - messages, 4-1, 4-2, 4-3
  - syntactical errors, 3-2
- Error messages, Libr-11, 6-2, 7-6, 7-8
- Examples of file specification combinations, 7-5
- Example linking session, 3-7
- Exit (E) switch, 3-5
- External symbols, 1-3
  
- Fatal linking error messages, 4-2, 4-3
- File specification combinations, Libr-11, 7-5
  - Examples, 7-5
- File specifications, Link-11, 3-1
- Format, Libr-11 command string, 7-1
  
- Global symbol directory (GSD), 2-1
- Global symbols, 1-3
  
- Input library, 7-3
- Input module, 2-1
- Inserting object modules, 7-2
- Internal symbols, 1-3
- Introduction to Libr-11 Librarian, 6-1
  
- Legal file specification combinations, Libr-11, 7-5
- Libr-11 Librarian introduction, 6-1
- Library (L) switch, 3-5, 3-6
- Library
  - creation, 7-1
  - file specifications, 7-5
  - listing, 7-3
  - Monitor, 3-6, 3-7
  - naming, 7-4
  - searches 3-6
  - update, 7-2
- LIBR.TMP, 7-5
- Link-11 advantages, 1-1
- Linking session example, 3-7
- Listing, 7-6
- Loader
  - absolute, 2-1
  - DOS Monitor, 2-2
- Loading, Link-11, 3-1
- Load map, Link-11, 2-2
- Load module, 2-1
  
- Modules
  - input, 2-1
  - load, 2-1
  - object, 2-1
  - output, 2-1
- Monitor library, 3-6
  
- Named control section, 1-2
- Naming libraries, 7-4
- Number sign (#), usage by Link-11, 3-1
  
- Object module, 2-1, 6-1
  - deleting, 7-2
  - inserting, 7-2
  - replacing, 7-3
- OBJMOD.LST, 7-4
- ODT switch, 3-4
- Operating procedures
  - Libr-11, 7-1
  - Linker, 3-1

Output library, 7-3  
 Output module, 2-1

Parentheses as writing convention, 6-2  
 Pass 1, 3-4  
 Pass 2, 2-1, 3-4  
 PIP, 7-1  
 Program sections, absolute and relocatable, 1-2

Relinking Link-11, 1-3  
 Relocatable program sections, 1-2  
 Replacing object modules, 7-3  
 Restarting Link-11, 4-1  
 RETURN key (<CR>), 6-2  
 RUN command, 7-1

Sharing data and/or programs among object modules, 1-2  
 Switches, 1-3, 3-2  
   Bottom (B), 1-3, 3-2  
   Concatenate (CC), 3-4  
   Exit (E), 3-5  
   general notes, 3-5  
   Library (L), 3-5, 3-6  
   ODT, 3-4  
   options, Libr-11, 6-2  
   summary, Link-11, 5-1  
   Tape (TA), 3-3  
   Top (T), 1-3, 3-2  
   TTransfer address (TR), 3-4  
   Undefined globals (U), 3-3

Symbols  
   entry, 1-3  
   external, 1-3  
   global, 1-3  
   internal, 1-3

Syntactical error, 3-2

Tape (TA) switch, 3-3  
 .TITLE assembler directive, 7-2  
 Top address, Link-11, 1-3  
 Top of memory, 1-3  
 Top (T) switch, 1-3, 3-2  
 Transfer address (TR) switch, 3-4

Undefined globals (U) switch, 3-3  
 Underlining, writing convention, 3-1, 6-2  
 Unnamed control section, 1-2  
 Updating a library, 7-2  
 User libraries, 3-6, 3-7

Warning error messages, Link-11, 4-1  
 Writing conventions, 6-2

## HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 & PDP-12  
Digital Software News for the PDP-11  
Digital Software News for the PDP-9/15 Family

These newsletters contain information applicable to software available from Digital's Program Library, Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contact at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning Digital's Software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance Report form with details of the problem to:

Software Information Service  
Digital Equipment Corporation  
146 Main Street, Bldg. 3-5  
Maynard, Massachusetts 01754

These forms which are provided in the software kit should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

Orders for new and revised software and manuals, additional Software Performance Report forms, and software price lists should be directed to the nearest Digital Field office or representative. U.S.A. customers may order directly from the Program Library in Maynard. When ordering, include the code number and a brief description of the software requested.

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS  
Digital Equipment Corporation  
146 Main Street, Bldg. 3-5  
Maynard, Massachusetts 01754



READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability and readability.

---

---

---

Did you find errors in this manual? If so, specify by page.

---

---

---

---

How can this manual be improved?

---

---

---

---

Other comments?

---

---

---

---

Please state your position. \_\_\_\_\_ Date: \_\_\_\_\_

Name: \_\_\_\_\_ Organization: \_\_\_\_\_

Street: \_\_\_\_\_ Department: \_\_\_\_\_

City: \_\_\_\_\_ State: \_\_\_\_\_ Zip or Country \_\_\_\_\_

-----  
Fold Here  
-----

-----  
Do Not Tear - Fold Here and Staple  
-----

FIRST CLASS  
PERMIT NO. 33  
MAYNARD, MASS.

BUSINESS REPLY MAIL  
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

**digital**

Digital Equipment Corporation  
Software Information Services  
146 Main Street, Bldg. 3-5  
Maynard, Massachusetts 01754

