PERKIN-ELMER

OS/32
LIBRARY LOADER

Reference Manual

48-020 R00

The Perkin-Elmer Corporation, Computer Systems Division 2 Crescent Place, Oceanport, New Jersey 07757

N O T I C E

The Perkin-Elmer OS/32 Task Establisher Task (TET) is replaced by a new linkage editor called the Perkin-Elmer OS/32 Link. For OS/32 R05.2 and higher, all references made to TET in OS/32 software manuals apply to Link.

For current users of TET, all references made to Link in the revised OS/32 software manuals can be applied to TET.

PAGE REVISION STATUS SHEET

PUBLICATION NUMBER 48-020 R00

TITLE Operating System/32 (OS/32) Library Loader Reference Manual

REVISIONS R00 DATES 4/80

PAGE	REVISION
i	R00
iii	R00
v through vii	R00
1-1 through 1-5	R00
2-1 through 2-30	R00
A-1, A-2	R00
B-1 through B-3	R00
C-1	R00
D-1, D-2	R00
Ind-1	R00

PREFACE

This manual describes the functions of the Operating System/32 (OS/32) Library Loader under the OS/32 operating system. This manual is intended for operators, programmers, and system administrative personnel.

Chapter 1 discusses loading and starting the library loader along with loading pure and impure segments. Chapter 2 presents all library loader commands and their definitions.

Appendix A contains the command summary, while Appendix B contains the message summary. A sample command sequence is shown in Appendix C. Appendix D presents a sample file library creation.

TABLE OF CONTENTS

CHAPTERS (Continued)

APPENDIXES

TABLES

CHAPTER 1
THE OPERATING SYSTEM/32 (OS/32) LIBRARY LOADER

1.1 INTRODUCTION

The Operating System/32 (OS/32) Library Loader resolves external references, edits file libraries, defines common blocks, and builds object load modules. The library loader operates under OS/32 and interactively accepts commands and generates messages to the operator. Library loader biases programs on a doubleword boundary.

An object program library file can be created on a bulk storage device by using operator commands. This file can be searched for a particular program, selectively copied onto another logical unit (lu), or added to under operator control. Automatic link editing is available allowing the operator, with one command, to load all library programs required for any one particular job. A group of programs can be pseudo-loaded, producing a single absolute load module that can be originated at any address, and then loaded by the 32-Bit Relocating (REL) Loader. This module contains forward references but no entry point or externally referenced symbols.

1.2 OS/32 LIBRARY LOADER OPERATION

After loading the library loader, lu 5 must be assigned by the user for the command file. Primary and optional secondary input files along with optional output files can be assigned to any of the other lu devices.

After entering the START command, library loader displays the following message:

```
LIBLDR-Rnn
```

This message indicates that the library loader is ready to accept commands. This message is also displayed after the XOUT command is entered.

1.3 PROGRAM LOADING WITH PURE AND IMPURE SEGMENTS

Program segmentation allows certain blocks of executable code to be shared when operating under OS/32. Segmentation is acccmplished by dividing a file into pure and impure segments at assembly time. The pure segment is a sharable segment while the impure segment is not. There are certain rules used to handle file segmentation. For example, a program such as CAL, that contains both pure and impure segments, can be loaded and executed by specifying a single impure bias value. The BIAS command is used to specify the impure bias for this type of program load operation.

As another example, a FORTRAN program can be loaded with a single impure bias value, even though the main program is impure and the run time library routines are pure. All programs, pure or impure, are loaded contiguously when only the single bias value is specified. To separate the pure and impure segments of a program or number of programs, specify a pure bias, using the pure bias (PB) command, and pure segments will be originated at the pure bias, and impure segments at the impure bias. When the pure bias is zero or unspecified, all segments are loaded at the impure bias value. The BIAS command automatically zeros the pure bias value; therefore, when specifying both biases, enter the PB command after the BIAS command. Table 1-1 displays program loading and the pure and impure bias addresses.

TABLE 1-1 PROGRAM LOADING

BIAS SPECIFIED	PURE BIAS SPECIFIED	PROGRAM CONTAINS		RESULT
		PURE SEGMENT	IMPURE SEGMENT	
0 (1)	0	*(2)		Program loaded (PBOT) (3)
0	0		*	Program loaded at PBOT
0	0	*	*	Pure segment loaded at PBOT. Impure segment immediately follows pure segment.
*	0	*		Program loaded at impure bias address
*	0		*	Program loaded at impure bias address
*	0	*	*	Pure segment loaded at impure bias address. Impure segment immediately follows pure segment.

BIAS SPECIFIED	PURE BIAS SPECIFIED	PROGRAM CONTAINS		RESULT
		PURE SEGMENT	IMPURE SEGMENT	
0	*	*		Program loaded at pure bias address
0	*		*	Program loaded at PBOT
0	*	*	*	Pure segment loaded at pure bias address. Impure segment loaded at PBOT
*	*	*		Program loaded at pure bias address
*	*		*	Program loaded at impure bias address
*	*	*	*	Pure segment loaded at pure bias address. Impure segment loaded at impure bias address

1 If no bias is specified, zero is the default.

2 * indicates specified.

3 PBOT indicates pure bottom.

1.4 STATEMENT SYNTAX CONVENTIONS

These statement syntax conventions are used in all statement, command, and instruction formats:

CONVENTION	USAGE
Capital letters, parentheses, and punctuation marks	must be entered exactly as shown.
Lowercase letters n	represent parameters or information provided by the user.

Underlining PAUSE	indicates only the underlined portion of the entry is required.
Ellipsis ... param1,...,param5	represents an indefinite number of parameters or a range of parameters.
Lettering with shading n	represents a default option.
Braces { }	represent required parameters from which one must be chosen.
Brackets [ ]	represent an optional parameter that can be chosen.
Braces inside brackets [{ }]	represent optional parameters from which one can be chosen.
Commas ,	separate parameters and substitute missing positional parameters.
Comma inside brackets [,]	must be entered if the optional parameter is chosen.
Comma preceding braces inside brackets [,{ }]	must be entered if one of the optional parameters is chosen.

Comma outside brackets except last parameter [ ],[ ],[ ] [,]	must be entered in place of missing positional parameters and to separate optional parameters that are chosen. Commas are omitted for trailing parameters.
Equal sign separating keyword from parameters KEYWORD=param	must be entered to associate parameter with keyword.

NOTE

Library loader commands require that only the first two letters of each command be entered. This is true in all cases except one: the LINKFILE command uses the letters LF so it will not be confused with the LINK (LI) command.

CHAPTER 2
LIBRARY LOADER COMMANDS

2.1 INTRODUCTION

Library loader commands are presented alphabetically in this chapter. All library loader commands are listed below:

AMAP
BF (BACKSPACE FILE)
BR (BACKSPACE RECCRD)
BIAS
BC (BLANK COMMON)
COPY
DUPE
EDIT
ENL
FIND
FF (FORWARD FILE)
FR (FORWARD RECORD)
GO
LABEL
LC (LABEL COMMON)
LINK
LF (LINKFILE)
LOAD
LG (LOG)
MAP
OUT
PAUSE
PB (PURE BIAS)
RW (REWIND)
TABLE
TOP
WF (WRITE FILEMARK)
XOUT

2.2 AMAP COMMAND

The AMAP command outputs a memory map to the specified logical unit (lu) with all symbols listed in alphabetical order.

Format:

AMAP lu

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu on which the map is output.

Examples:

AM PR:

```
 PROGRAMS:
NAME      IMPURE  PURE    ABS        NAME      IMPURE  PURE    ABS
ASYNCTOP  0266E0                     CDVR.F01  00F558
CLEANUP   00EE9E          000030     CMDB.F28  00FB00
CMEX.F08  0125C0                     CMON.F01  014A20

 ENTRY POINTS:
013E88 $BUILD     013F52 $CLEAR     014188 $COPY      0140E4 $ELSE      014114 $ENDC
013F22 $EXIT      014018 $IFE       014000 $IFG       01400C $IFL       01403C $IFNE
014024 $IFNG      014030 $IFNL      014060 $IFNNULL   014048 $IFNULL    01409E $IFNX

COMMON-BLOCKS:

UNDEFINED:
```

2.3 BF (BACKSPACE FILE) COMMAND

The BF command backspaces the lu one filemark, leaving the lu positioned before the filemark. If no filemark is found, an error message is issued.

Format:

BF lu

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.

Functional Details:

If the lu is positioned at the first file and the BF command is issued, an error message will be displayed. The RW (rewind) command must be used in this case, and the lu will be positioned at load point.

Example:

BF 2	Backspaces one filemark on lu 2.

Error Messages:

I/O DEVICE ERROR	No filemark was found.

2.4 BR (BACKSPACE RECORD) COMMAND

The BR command backspaces the lu one record in the file.

Format:

BR lu

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.

Example:

BR 2	Backspaces one record on lu 2.

2.5 BIAS COMMAND

The BIAS command sets a bias address to be used as the base address for the next impure relocatable program.

Format:

BIAS[adr]

Parameters:

adr	is a 1- to 6-digit hexadecimal number specifying the address to be used as the base address. If this parameter is omitted, the default is the next available location above the highest address used.

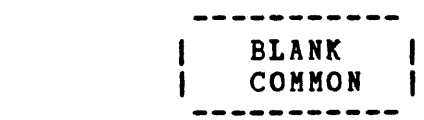
Functional Details:

The impure bias address becomes the bias of the impure relocatable segment. Immediately after loading any file, the bias value is set to the next available location above the highest address used. When the loader is initially loaded, bias is set to the next location above the loader. This value is not initialized when the loader is restarted. If no operand is specified in this command, the bias is set to the next location above the loader. The BIAS command resets the pure bias value to zero. The BIAS command cannot be used during output of an overlay module.

If the BIAS command is not specified, the bias default is zero.

Example:

BI 189C	Sets the bias at X'189C'.



2.6 BC (BLANK COMMON) COMMAND

The BC (BLANK COMMON) command allocates the maximum amount of memcry in bytes for blank ccmmon area.

Format:

BC n

Parameters:

n	is a 1- to 6-digit hexadecimal number specifying the number of bytes to allocate for a blank common.

Functional Details:

The BC command must be entered before the LO command is entered. When a blank common area is specified by the BC command, the loader places the common area at the impure bias value or at the impure bias value plus labeled common, if labeled common was previously specified. (See LC command). BLANK COMMON cannot contain pre-loaded data.

Example:

BC 2000	Allocates X'2000' bytes for blank common.

2.7 COPY COMMAND

The copy command copies one program from the first lu to the seccnd lu.

Format:

COPY lu_1,lu_2[program-label]

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.
program-label	is a 1- to 8-character alphanumeric string specifying the name of the program on the specified lu to be copied from one lu to another.

Functional Details:

If a program-label is specified, the first lu is searched for the specified program-label. If no program-label is specified, the first program encountered on the first lu is copied. After a copy operation is completed, the first lu specified is left-positioned past the end of the copied program.

Example:

CO 1,2 INITCRT	Copies the file INITCRT from lu 1 to lu 2.

2.8 DUPE COMMAND

The DUPE command copies programs from the first lu to the second lu until the specified program is found. The specified program is not copied and the operation terminates.

Format:

DUPE lu_1,lu_2[program-label]

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.
program-label	is a 1- to 8-character alphanumeric string specifying the name of the program on the specified lu to be searched to terminate the dupe operation.

Functional Details:

If no program-label is specified, programs are copied until an EOF or EOM on the first lu is detected.

Example:

DU 1,2 INITCRT	Copies all files from lu 1 to lu 2 until the file INITCRT is found.

2.9 EDIT COMMAND

The EDIT command searches the specified library tape or file and selectively loads those programs needed to satisfy external references (EXTRNs) within previously loaded programs.

Format:

EDIT lu[program-label]

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.
program-label	is a 1- to 8-character alphanumeric string specifying the program-label on the specified lu to be found.

Functional Details:

The criterion for loading during an edit operation is a match between the label of a program in the library file and the symbolic name of any EXTRN in the loader symbol table. The EDIT command:

- searches for the program on the lu before starting the edit process, if a program-label is specified. If no program-label is specified, the lu is processed from its current position.
- searches the loader symbol table for undefined EXTRNs. If there are none, the operation terminates.
- searches for an lu forward from its current position for a program-label matching an undefined EXTRN, if there are undefined EXTRNs in the symbol table. If such a label is found, that program is loaded and linked, and the second step is repeated. If an EOF or EOM is encountered in the search, an appropriate message is logged, and the operation terminates.

The EDIT command does not have the rewind capability.

If an ENDVOL is read, the edit operation terminates. This label can be placed at the end of each paper tape file library to prevent reading off the end of the tape. Following an edit, the MAP command can be used to determine if any undefined EXTRNs are in the symbol table.

Example:

```
ED 4            Resolves external references from lu 4.
```

2.10 END COMMAND

The END command terminates the library loader.

Format:

END

2.11 FIND COMMAND

THe FIND command searches the specified lu until the user-specified program is found. The lu is then backspaced to the beginning of the that program, which then can be loaded or linked.

Format:

FIND lu[program-label]

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.
program-label	is a 1- to 8-character alphanumeric string specifying the name of the program to be found.

Functional Details:

The lu is not rewound by the library loader when the FI command is given. Use the FI command to position a file containing several programs. If the file is not on a rewindable device, backspace the file one record after the FI command is completed befcre loading the found program. If the specified program is not found, the FIND command terminates when an end-of-file (EOF) or end-of-medium (EOM) is encountered. The following messages are logged to the console device:

```
EOF
(program-label) NOT FOUND
```

Example:

```
FIND 1 INITCRT          Searches for program INITCRT.
```

2.12 FF (FORWARD FILE) COMMAND

The FF command forward spaces the lu one filemark and positions the lu past the filemark. If no filemark is found, an error message is issued.

Format:

FF lu

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.

Example:

FF 2	forwards one file on lu 2.

Error Messages:

I/O DEVICE ERROR	No filemark was found.

2.13 FR (FORWARD RECORD) COMMAND

The FR command forward spaces the lu to the next record in the file.

Format:

FR lu

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.

Example:

FR 2	Forwards one record on lu 2.

2.14 GO COMMAND

The GO command starts execution of the load module by transferring control from the library loader to the transfer address of the loaded programs.

Format:

```
GO
```

Error Messages:

CMD-ERR	The transfer address of the loaded file was not specified.

2.15 LABEL COMMAND

The LABEL command is used to label a program in a library file.

Format:

LABEL lu[program-label]

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.
program-label	is a 1- to 8-character alphanumeric string indicating the name to be given to the program being labeled on the specified lu.

Functional Details:

This command causes the loader to output one record containing the specified program-label to the specified lu. If the program-label exceeds eight alphanumeric characters, only the first eight are used. This command can also be used prior to an assembly or compilation that writes the binary object program to the specified lu.

If this command is used between an CU and XO command, it is rejected and the following error message is displayed:

CMD-ERR

Example:

LA 1 CRTDVR	Labels a program on lu 1 as CRTDVR.

2.16 LC (LABELED COMMON) COMMAND

The LC command allocates the maximum amount of memory in bytes for a labeled common area.

Format:

LC n

Parameters:

n	is a 1- to 6-digit hexadecimal number specifying the number of bytes to allocate for labeled common.

Functional Details:

The LC command must be entered before the LO command is entered. When a labeled common area is specified by the LC command, the loader places the common area at the bias value or at the bias value plus BC, if BC was previously specified. Labeled common can contain pre-loaded data before being loaded into memory.

Example:

LC 2000	Allocates X'2000' bytes for labeled common.

2.17 LINK COMMAND

The LINK command is used to load and link additional files after the LOAD command loads the first file.

Format:

LINK lu[program-label]

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.
program-label	is a 1- to 8-character alphanumeric string specifying the program on the specified lu to be linked and loaded into memory.

Functional Details:

The symbol table is not cleared, so the loaded program can be linked to any previously loaded program and can reference previously defined common blocks. The current bias values are used to load the program. If a program-label is specified, the user-specified lu is first searched for the program-label. If no program-label is specified, the first program encountered on the lu is loaded and linked. Following a link operation, the specified lu is left-positioned past the end of the loaded program.

Example:

LI 4	Loads and links a program from lu 4.

2.18 LF (LINKFILE) COMMAND

The LF command links and loads all programs and subroutines in the specified file until an end-of-volume (EOV) is reached.

Format:

LF lu[program-label]

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.
program-label	is a 1- to 8-character alphanumeric string specifying the program on the specified lu to be linked and loaded into memory.

Functional Details:

Linking begins from the current position of the file. If a program-label is specified, the lu is positioned to that program befcre linking begins.

Example:

LF 4	Links and loads all programs from lu 4 until EOV is reached.

2.19 LOAD COMMAND

THe LOAD command initializes the loader and loads a program from a specified lu at the current bias values.

Format:

LOAD lu[program-label]

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.
program-label	is a 1- to 8-character alphanumeric string specifying the name of the program to be loaded.

Functional Details:

Befcre the program is loaded, the loader symbol table is cleared, and previously defined common blocks are released. This command should be used to load the first of a group of user programs not related to any previous job. If a program-label is specified, the specified lu is searched for the program-label. If no program-label is specified, the first program encountered on the lu is loaded. Following a load operation, the specified lu is left-positioned past the end of the loaded program.

Example:

LO 1	Loads a program from lu 1.

2.20 LG (LOG) COMMAND

The LOG command logs all library loader commands from the command input device to a specified lu.

Format:

$$LG\left\{\begin{matrix}0\\1\end{matrix}\right\},lu$$

Parameters:

0	specifies logging is enabled.
1	specifies logging is disabled.
lu	is a hexadecimal number from 1 through 254 specifying a log device.

Example:

LG 1,5	Disable logging to lu 5.

2.21 MAP COMMAND

The MAP command sends a memory map to the specified lu with all symbols in address order. The map shows the location of programs loaded into memory and the locations of files output in a load module. The map includes only those files processed during and after the last load operation.

Format:

MAP lu

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which the map is sent.

Functional Details:

Link and edit operations add to the map, while the load operation begins a new one. The map shows the starting address of each file, the next available address, the location of every ENTRY defined, the starting address of each common block defined, and a list of any undefined EXTRNs.

Examples:

MA PR:

```
 PROGRAMS:
NAME      IMPURE  PURE    ABS        NAME      IMPURE  PURE    ABS
EXIN.F12  016898  003010  000000     CLEANUP   00EE9E          000030
DCB02200  0062C0          000400     INITTYKP  006348          000420
DCB03800  006728          0005B0     DCB03900  006780          0005E8

 ENTRY-POINTS:
000000 JRNLBKS    000000 SCTT       000000 SCTH       000060 SPT        0000D0 ISTAB
000420 ISR1TYKP   000580 TYKPHMRK   000588 TYKPCR     00059A TYKPBARW   000D72 ICRTHASH
000D78 ICRTCR     000D86 ICRTBS     0023E8 ASYNCPB    002642 ARENDB     002648 AREOT

COMMON-BLOCKS:

UNDEFINED:
```

2.22 OUT COMMAND

The OUT command selects the output mode used to generate a load module rather than to load programs into memory.

Format:

OU lu[program-label]

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.
program-label	is a 1- to 8-character alphanumeric string specifying the filename to be generated before the output mode is selected.

Functional Details:

In cutput mode, during load, link, and edit operations, all data normally loaded in memory is output in loader format to the specified lu. All programs are output as absolute code. The OU command sets the bias value to the first location above the current operating system. The BI command must be issued after the OU command if any other bias address is required. The pure bias address is reset to zero. Any external references or common references are resolved and converted to forward references without symbolic names. Thus, a load module program can be loaded by the 32-bit REL loader.

Example:

OU 2	Generates a load module on lu 2.

```
-----------
|  PAUSE  |
-----------
```

2.23 PAUSE COMMAND

The PAUSE command causes the loader to pause.

Format:

PAUSE

Functional Details:

The loader can be continued by the operator CONTINUE command, and the status of the loader will be the same as before the pause. Do not use the PA command to assign additional disc files. Use the EN command to release memory and restart the library loader once all assignments are made.

2.24 PB (PURE BIAS) COMMAND

The PB command sets a bias address to be used as the base address for the pure relocatable segments.

Format:

PB [adr]

Parameters:

adr	is a 1- to 6-digit hexadecimal number specifying the address to be used as the base address for pure segments.

Functional Details:

After loading a program, unless a new PB command is issued, the pure bias value is set to the next available location above the highest pure address used. When the loader is initially loaded, PB is zero. This value is not initialized when the loader is restarted, but it is reset to zero by the BIAS and OU commands. The PB command cannot be used during output of an overlay module.

Example:

PB 1418	Sets the pure bias at X'1418'.

2.25 RW (REWIND) COMMAND

The RW command rewinds the specified lu.

Format:

```
RW lu
```

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.

Example:

RW 1	Rewinds lu 1.

2.26 TABLE COMMAND

The TABLE command scans the first lu and lists on the second lu all program-labels found from the current file position when the command was issued until an EOF is read.

Format:

TABLE lu_1,lu_2

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.

Functional Details:

To list all program-labels on the first lu, ensure that the lu is positioned at the beginning of the file. If the record containing the program-label was passed before the TABLE command was entered, that program is not listed. Any program without a label in the file is not shown in the list.

The TABLE command does not have the rewind capability.

Example:

TA 1,5	Lists program-labels from lu 1 to lu 5.

```
-----------
|   TOP   |
-----------
```

2.27 TOP COMMAND

The TOP command sets the highest memory address available to the loader for loading user programs. The TOP command overrides the default that all memory on the system is available.

Format:

TOP[adr]

Parameters:

adr	is a 1- to 6-digit hexadecimal number specifying the highest memory address available for user programs.

Functional Details:

The TO command can be used to build load modules for execution on systems with more or less memory than the system used to build the load module. This command deletes the symbol table.

Example:

TO FFFFF	Sets the highest memory address available.

2.28 WF (WRITE-FILEMARK) COMMAND

The WF command writes a filemark to the specified lu.

Format:

WF lu

Parameters:

lu	is a decimal number from 1 through 254 specifying the lu to which a device is assigned.

Functional Details:

If a device is write-protected and the WF command is attempted, an I/O error status will be issued.

Example:

WF 2	Writes one filemark on lu 2.

2.29 XOUT COMMAND

The XOUT command is used in conjunction with generation of a load module. XCUT generates the final record of the load module and then cancels the output mode.

Format:

XOUT

Functional Details:

This is a typical command sequence used to create a load module:

```
CUT 2
BIAS A000
LOAD 6 MAR
LINK 7
EDIT 8
XOUT
```

APPENDIX A
COMMAND SUMMARY

AMAP lu

BC n

BF lu

BIAS[adr]

BR lu

COPY lu_1,lu_2 program-label

DUPE lu_1,lu_2 program-label

EDIT lu[program-label]

END

FF lu

FIND lu[program-label]

FR lu

GO

LABEL lu[program-label]

LC n

LF lu[program-label]

LG $\left\{\begin{matrix}0\\1\end{matrix}\right\}$,lu

LINK lu[program-label]

LOAD lu[program-label]

MAP lu

OUT lu[program-label]

PAUSE

PB [adr]

RW lu

TABLE lu_1,lu_2

TOP[adr]

WF lu

XOUT

APPENDIX B
LIBRARY LOADER MESSAGE SUMMARY

MEM-FULL

A program being loaded exceeds the available memory space. If the program overlaps the table, the MEM-FULL message is logged and the load is aborted. Not enough common space was allocated or the program loaded was too large.

CMD-ERR

The loader does not recognize library loader commands if a LABEL (LA) command is found between an OUT (OU) command; if GO is issued when no transfer address was specified in any loaded program; or if an LI or ED command was entered before a program was loaded.

* * NO PROGRAMS * * BIAS (current location)

A MAP was requested, and no programs were loaded.

EOF

A filemark or ENDVOL was read during binary input.

EOM

An end-of-media was encountered during a binary input.

CKSM ERR

A record was read in which the checksum did not match the checksum computed by the loader. Following the message, the loader paused. If the input device is paper tape, it can be manually backspaced one record before continuing the load. Bulk storage devices are automatically backspaced when the CONTINUE command is entered.

SEQ-ERR

A record was read out of sequence. Reposition the input lu at the beginning of the current program and retry.

REF-LOOP

A loop was found in a forward or external reference thread, the object program was incorrectly generated, or the XO (XOUT) command was given during the building of overlay modules, when unresolved external references remain in the completed overlay.

M (entry point)

The specified entry pcint was defined more than once. The load continues, and the previous entry point encountered remains in effect. The others are ignored.

LOAD ABORTED

An unrecoverable error condition caused the abortion of a load in progress. Repeat the original LO command, which will clear any incorrect symbol locations that might have occurred in the symbol table.

ADRS-ERR

An attempt was made to load a program below the top of the loader. The program might contain absolute data at such an address or the impure bias might be set incorrectly. The load process is aborted. This error message is also displayed if a program with references or definitions of halfword-address constants is loaded above 64K, or if a subsequent declaration of a labeled common block is greater than the length of the first declaration of that common block.

DEV END

A device went off-line during input.

I/O DEV ERR

A parity error was detected on input, or an off-line condition was detected on output. The operating system status byte and a device address is displayed.

LOAD ERR

An illegal loader item was detected during a load.

(program-label) NOT FOUND

An end-of-device condition occurred before the program-label was located during a search operation.

If commands are issued in batch mode with logging suppressed, and an error is found, the last command executed is displayed on the console device.

If the loading process is aborted in batch mode, the loader returns control to the operating system and issues an end-of-task code 1. If commands are issued interactively, the loader reads the next command from the console device.

APPENDIX C
SAMPLE COMMAND SEQUENCE

The following is an example of building an object load module while running in an OS/32 environment. It is assumed that CUP processing, which gives both the pure and impure bias values, was completed.

```
LOAD .BG, LIBLDR
TASK .BG
AS 1,Voln:CUPOUT.OBJ,SRC
AS 2,SYS.LIB,SRO
AS 3,PR:
AL Voln:SYSTEM.OBJ,IN
AS 4,SYSTEM.OBJ
AS 5,CON:
START
10:19:31 .BG LIBLDR 04-00
.BG>OU 4
.BG>TO FFFFF
.BG>BI 189C
.BG>PB 1418
.BG>LO 1
.BG>ED 1
.BG>ED 2
.BG>XO
.BG>MA 3
.BG>AM 3
.BG>END
10:24:35 .BG:END OF TASK 0
```

APPENDIX D
SAMPLE LIBRARY CREATION

```
* LO.BG,LIBLDR
* AS1, PROG1.OBJ
* ALPROGLIB.OBJ,IN,126
* AS2, PROGLIB.OBJ
* AS3, CON:
* AS5, CON:
* AS4, PROG2.OBJ
* AS6, PROG4.OBJ
* AS7, PROG5.OBJ
- LIBLDR 04-00
- OUT 2
- COPY 1,2
- COPY 4,2
- COPY 6,2
- COPY 7,2
- WF 2
- RW 2
- TABLE 2,5
- RW 2
- END
- END OF TASK
```

PROG3.OBJ is inserted between PROG2.OBJ and PROG4.OBJ in the file library:

```
* LO.BG,LIBLDR
* AS1, PROGLIB.OBJ
* ALNEWLIB.OBJ
* AS2, NEWLIB.OBJ
* AS3, CON:
* AS5, CON:
* AS4, PROG3.OBJ
* ST
- LIBLDR 04-00
- DUPE 1,2 PROG4
- COPY 4,2
- DUPE 1,2
- RW 2
- TABLE 2,5
- RW 2
- END
- END OF TASK CODE 0
```

User-written subroutines can be edited from the library if the:

- subroutine contains a PROG statement, or a label is created with the LABEL command;
- subroutine contains an ENTRY statement with a name that is the same as the name in the label;
- user main file contains an EXTRN statement that refers to the name of the subroutine in the library.

INDEX

PUBLICATION COMMENT FORM

Please use this postage-paid form to make any comments, suggestions, criticisms, etc. concerning this publication.

From _____ Date _____

Title _____ Publication Title _____

Company _____ Publication Number _____

Address _____

_____

_____

FOLD FOLD

Check the appropriate item.

☐ Error Page No. _____ Drawing No. _____

☐ Addition Page No. _____ Drawing No. _____

☐ Other Page No. _____ Drawing No. _____

Explanation:

FOLD FOLD

CUT ALONG LINE

Fold and Staple

No postage necessary if mailed in U.S.A.

6434

STAPLE STAPLE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 22 OCEANPORT, N.J.

POSTAGE WILL BE PAID BY ADDRESSEE

PERKIN-ELMER

Computer Systems Division
2 Crescent Place
Oceanport, NJ 07757

TECH PUBLICATIONS DEPT. MS 322A

FOLD FOLD

STAPLE STAPLE