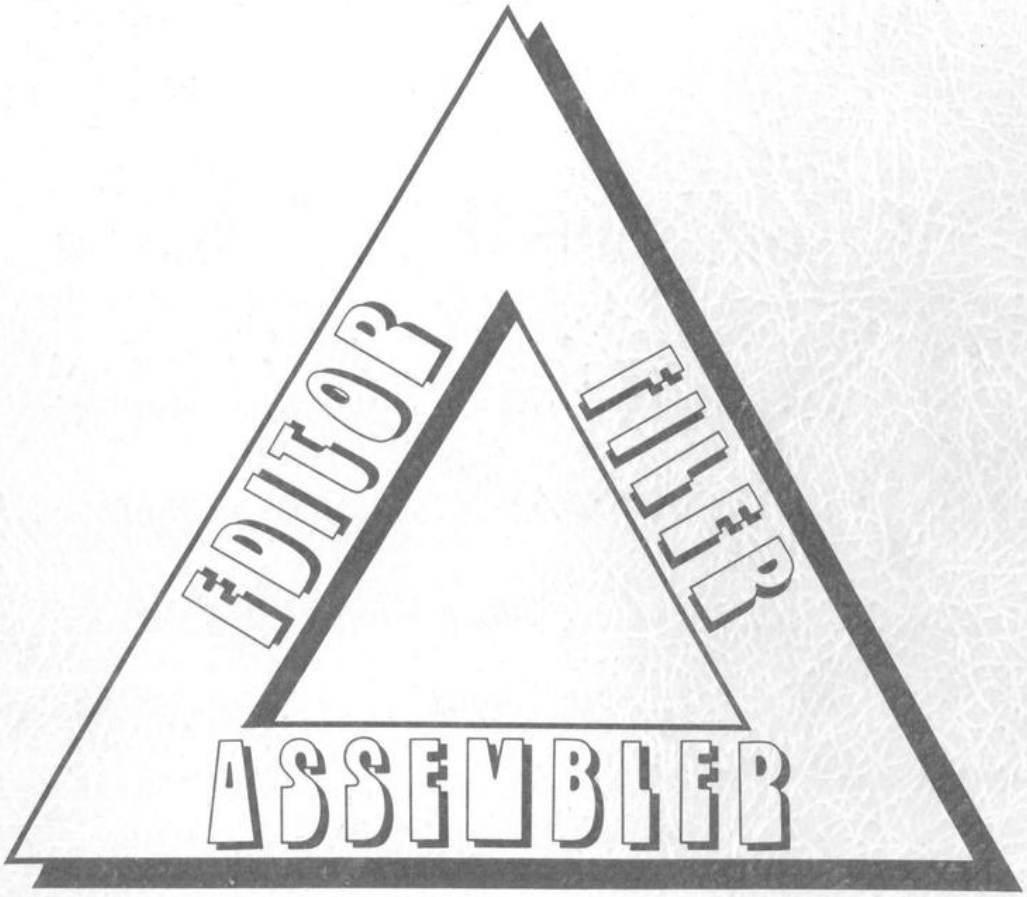
LAP6W



▲ LABORATORY COMPUTER FACILITY

▲ UNIVERSITY OF WISCONSIN

LAP6W

Conrad C. Bjerke
Jo Ann Coram
Peter A. Gutterman
Richard A. LeFaivre
Marilyn Lenahan
Alan C. Roochvarg

LCF Program UP-070-03

Laboratory Computer Facility
83 Medical Sciences Building
University of Wisconsin
Madison, Wisconsin 53706

This work was supported by the Biotechnology Resources Branch, Division of Research Resources, National Institutes of Health, under Grant RR-00249. It is based on work supported under Grant FR-218 at the Computer Research Laboratory, Washington University, St. Louis, Missouri.

FORWARD

In 1967 LAP6 was published by Mary Allen Wilkes of the Computer Research Laboratory, Washington University, St. Louis, who also had conceived and developed the original LINC assembly program. The greatly enhanced interactive assembling, editing, and filing capabilities of LAP6 represented an innovation of fundamental importance to the users of small computers. This volume contains a description of LAP6W, which incorporates modifications of LAP6 by the staff of the Laboratory Computer Facility at the University of Wisconsin Medical School. Large portions of the text of the LAP6W document have been copied verbatim from the LAP6 Manual.

This latest version of LAP6W, originally released in July of 1970, includes several additions and improvements: an 8K option which allows the system to utilize 8K of memory; a feature which permits a user to expand the meta-command capabilities of the system; and rewritten LINC and PDP-8 assemblers (along with the reference table maker) which provide considerably faster assembly and reference table compilation for long programs. It is our hope that these changes will make the system more useful to LINC, μ-LINC, LINC-8, and PDP-12 users.

Staff of the Laboratory Computer Facility

July, 1972

Table of Contents

Table of Contents (Continued)

Abstract

The LAP6W system is an on-line program for the LINC, μ-LINC 100, μ-LINC 300, LINC-8, and PDP-12 which uses the LINC or Teletype keyboard and scope for communication with the user and the magnetic tapes for storage and working area. It may be used for preparation and editing of any character string (manuscript) or especially for LINC or PDP-8 program preparation. The input-output character set is closely aligned with the standard ASCII character set.

The LAP6W editor handles the manuscript display so that any portion of the manuscript can be displayed at any time and edited directly by simply adding or deleting characters. Changes are shown integrated with the manuscript display as the user types.

Meta-commands provide automatic filing of manuscripts and programs on LINC tapes and handle the assembly and memory loading of LINC or PDP-8 binary programs. Debugging aids include displays and/or printed copies of symbol tables and errors, and repeatable access to the manuscript display for editing and reassembly.

Preface

The major features that distinguish the LAP6W system from LAP6 are:

1.	Character Editing	Characters may be deleted or inserted anywhere in a manuscript line.
2.	Context Searching	A LAP6W meta-command can search for a specified string of characters.
3.	Partial Save Manuscript	This meta-command does not use any intermediate tape storage.
4.	Extended Copy Manuscript and Copy Binary	The location of the manuscript or binary to be copied may be supplied by the user or obtained from the file Index.
5.	Two Tape Unit Numbers	A manuscript or binary may be copied from any tape unit to any tape unit; both the "from" and "to" units are specified in the meta-command. A single meta-command can be used to save up to two copies of a manuscript and/or binary if a second tape unit is specified.
6.	Implicit Meta-Command Parameters	An omitted unit number implies the current system unit; an omitted file location of a binary program implies the binary working area, etc.
7.	Copy File	The location of the Index to the file being copied may be specified by the user.
8.	Index Label and File Bounds	A LAP6W Index includes an alphanumeric Index label and a set of file bounds. Simply altering the file bounds respecifies the range of the LAP6W file.
9.	LINC Assembly	A LINC assembly package is provided which allows for the use of six [four] character symbols, maintains both execution and storage location counters, and assembles into a full 4K of core.
10.	PDP-8 Assembly	A PDP-8 assembly package is optionally provided for LINC-8 and PDP-12 users.
11.	PDP-8 Loader	On the LINC-8 and PDP-12, PDP-8 binary programs may be stored in and loaded from a LAP6W file.
12.	LINC and PDP-8 Loaders	The tape location of the LINC or PDP-8 binary may be supplied by the user or obtained from the file Index. The binary is loaded into as many banks of core as possible.
13.	Assembly	LINC or PDP-8 program manuscripts need not be added to the manuscript working area for assembly, but can be assembled directly from the file.

The following features have been incorporated in the second release of the LAP6W system:

14.	8K Option	An option is available to LINC-8, PDP-12 and μ-LINC 300 users which allows the system to utilize 8K of memory.
15.	Meta-programs	A "meta-program" feature has been added which permits the user to expand the meta-command capabilities of LAP6W to fit his particular needs.
16.	"Kill" Working Area	An additional editing capability has been added which permits the user to clear the manuscript working area from the keyboard.
17.	Tape Copying	Meta-commands which perform tape copies now utilize all of available memory as a copy buffer, considerably improving copying time on 4K and 8K machines.
18.	Assembly, Reference Table	The LINC and PDP-8 assemblers and the reference table maker have been re-written to be considerably faster for long programs.
19.	Manuscript Listing	Manuscript line numbers may optionally be included on output produced by LIST, PDP-8 LIST and PRINT MANUSCRIPT.

The following modifications have been made to PROGOFOP and TRAP, and thus affect only the LINC-8 and PDP-12 versions of LAP6W. For details, consult the documentation for PROGOFOP [2] and TRAP [3].

20.	2000_8-Block Tapes	An option is provided to allow programs to reference the upper half of 2000_8-block tapes on units 0 and 1 as units 4 and 5.
21.	Typing LINC Code	LINC code may be typed directly from the accumulator, eliminating the conversion to ASCII code.
22.	PDP-12 Interrupt Routines	LINC mode and PDP-8 mode interrupt handling procedures are now available in TRAP.
23.	LINC-8 START 400	PROGOFOP now interprets the START 400 switch as an actual start 400. The CLEAR switch is used to restart LAP6W.

Introduction

This manual describes LAP6W as it is available on the several versions of the LINC. By using one of the programs, GASTCL or GASTMB (LCF Program UP-039-03), a user may generate a LAP6W system suited to his computer that will operate on LINC tape unit 0, 1, 4, or 5. Other than the start-up procedure (Section I), the primary difference in LAP6W usage arises from different keyboard input devices. The classic LINC, μ-LINC 100 and μ-LINC 300, which have LINC keyboards, use CASE character combinations for manuscript display control; the LINC-8 and PDP-12, which have Teletype keyboards, use CONTROL shift characters for manuscript display control. In this manual, the CASE characters for the former are indicated in brackets ([]) after the corresponding CONTROL character combinations. Other differences in usage between different models of the LINC are noted where they occur.

It is also important to note the differences in structure of the various versions of LAP6W. The classic LINC and μ-LINC 100 systems are "2-bank" systems, in that they occupy two 2000_8-word banks of memory (the "lower" and "upper" banks). The LINC-8, PDP-12 and μ-LINC 300 systems are "multi-bank" systems, in that they occupy three or more banks of memory. The μ-LINC 300 system operates in, and loads user programs into, banks 0 ff. The LINC-8 and PDP-12 systems require that PROGOFOP or TRAP be resident in bank 0 for proper operation, and thus operate in and load user (LINC) programs into banks 1 ff. Programs may be loaded into bank 0 through use of the PDP-8 LOAD BINARY meta-command. It is recommended that LINC-8 and PDP-12 users thoroughly familiarize themselves with the documentation for PROGOFOP [2] and TRAP [3].

This manual is for the standard configuration of LAP6W, namely, a unit 0 system with the tape configuration shown in Chart I, page 61.

The LAP6W System

I. Loading Procedure for Unit 0 Systems

A. Classic LINC and μ-LINC 100 (2-Bank System)

1. Mount a LAP6W classic LINC system tape on unit 0.

2. Read blocks 400 through 407 into quarters 0 through 7:

   Set the left switches to 0701, the right switches to 7400 (RCG, 7/400).
   Raise the DO toggle.

3. When the tape stops, press START 20. LAP6W is now ready to accept keyboard input.

B. μ-LINC 300 (Multi-Bank System)

1. Mount a LAP6W μ-LINC 300 system tape on unit 0.

2. Read blocks 400 through 403 into quarters 0 through 3 of bank 0:

   Set LSW = 0600 (LMB 0)
   Raise the DO toggle
   Set LSW = 0701, RSW = 3400
   Raise the DO toggle

3. When the tape stops, press START 20. LAP6W is now ready to accept keyboard input.

C. LINC-8 (Multi-Bank System)

1. Mount a LAP6W LINC-8 system tape on unit 0.

2. Read in PROGOFOP [LCF Program UP-072-02] by raising the LOAD toggle. When PROGOFOP is loaded, it will proceed to load and start LAP6W. When the tape stops, LAP6W is ready to accept keyboard input.

3. After running a LINC program, LAP6W may be reloaded by pressing CLEAR.

D. PDP-12 (Multi-Bank System)

1. Mount a LAP6W PDP-12 system tape on unit 0. Place the tape drive in REMOTE with WRITE ENABLED.

2. Read TRAP [LCF Program UP-086-02] into quarters 0 and 1 of any bank:

   Set the MODE switch to LINC
   Press I/O PRESET
   Set LSW = 0701, RSW = 1000
   Press DO

3. When the tape stops, start TRAP by pressing START 400. TRAP will relocate itself to bank 0 and proceed to load and start LAP6W. When the tape stops, LAP6W is ready to accept keyboard input.

4. After running a LINC program, LAP6W may be restarted as follows (assuming TRAP is still resident in quarters 0 and 1 of bank 0):

```
Set the MODE switch to LINC
Press I/O PRESET
Set LSW = 0  0400
Press START LS
```

5. After running a PDP-8 program which was loaded by LAP6W, LAP6W may be restarted as follows (assuming locations 7600-7777 of field 0 have not been changed):

```
Set LSW = 0  7600
Press START LS
```

II. Display Format

After loading LAP6W, the system is ready to accept keyboard input. All manuscript lines and meta-command statements are displayed as they are typed. All characters shown in Chart II on page 62 are displayed except EOL, DELETE, CASE, and EOMS.

A. Manuscript lines are vertically centered around the middle of the scope automatically.

1. A cursor (^) appears in the current line. It marks the position at which LAP6W will insert the next character typed. Manuscript lines which are too long to fit on one scope line (more than 25 characters, 51 on the PDP-12) are automatically carried over to the next scope line. Breaking words between scope lines can be avoided by spacing until the cursor moves to the next line.

2. The line number (assigned automatically by LAP6W) of the current line appears below the current line in the form *<LN>*

3. The number of manuscript lines displayed may be varied by rotating knob 3.

4. Lines are positioned on the scope as follows:

   a. Sense switch 5 down: Lines beginning with (#), ([), or ($) are positioned at the left edge of the scope; other lines are indented 8 spaces (6 on the classic LINC).

   b. Sense switch 5 up: All lines are positioned at the left edge of the scope.

5. All the characters to the right of the cursor on the current line are not always displayed.

III. Keyboard Input

A. CHARACTER SET

The character set used in LAP6W appears in Chart II, page 62. On the left are the characters' names and internal representation; on the right are the corresponding Soroban, μ-LINC 300 and Teletype keyboard characters. In this manual, the characters will be referred to by their names. For example, the LAP6W characters EOL, DELETE and META may be produced by striking the Teletype keys marked RETURN, RUBOUT and LINE FEED.

B. META-COMMANDS

A meta-command is one of 27 special directives to LAP6W. It is executed by LAP6W at the time it is stated and deleted from the display at that time.

The character META, when typed as the first character of a manuscript line, produces an arrow (the META arrow: →) at the bottom of the scope. To state a meta-command, type the META arrow and the meta-command, followed by any arguments the meta-command may require. Terminate the statement by typing an "End of Line" (EOL).

1. The meta-command is executed when the terminating EOL is struck.

2. Illegal meta-command statements are deleted automatically.

3. A meta-command must be stated on a line by itself, i.e., a line beginning with the META arrow. It cannot be stated while entering a manuscript line.

4. A line starting with the META arrow will be interpreted as a meta-command only if it is terminated by striking EOL. If terminated in any other fashion, it will be treated as an ordinary line.

C. MANUSCRIPT

1. A manuscript is a string of characters formed into manuscript lines and retained by LAP6W as a permanent record of keyboard input. A manuscript can be edited, saved in a file, copied from file to file, or otherwise manipulated.

   a. One quarter of the LINC memory is used to collect manuscript. As the quarter is filled, it is saved on the LAP6W tape in the manuscript working area beginning in block 470. 512 keyboard characters fill one block.

   b. LAP6W will accept up to 72_{10} blocks of manuscript. Additional manuscript is ignored until the size of the manuscript is reduced.

   c. The manuscript in the manuscript working area, accessible to the scope and keyboard, is called the current manuscript.

2. A manuscript line is any combination of characters, excluding DELETE and EOMS, which is terminated with an EOL. An EOL by itself does not constitute a manuscript line and will be ignored.

   a. LAP6W assigns a line number to every line entered. The numbers are sequential, beginning with 1, and octal.

b. LAP6W will accept no more than 7775_8 manuscript lines. Additional lines are ignored until the size of the manuscript is reduced.

3. The line number on the scope (e.g., 63 below) is called the current line number.

a. The current line number identifies the manuscript line currently being, or about to be, added to the manuscript. At no time during the manuscript display does the manuscript not have a current line number.

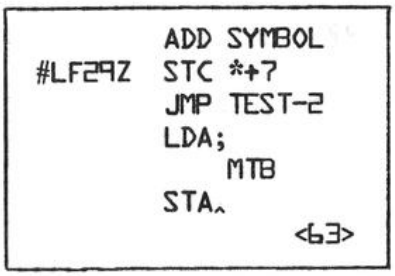


Figure 1. Manuscript Display

b. The number 1 appears as the current line number if there is no current manuscript. Subsequently, the new current line number appears every time a manuscript line is entered.

In Figure 1, when the current line, 63, is terminated with EOL, all the lines will move up one space, the top line, which is line 56, will disappear and 64 will appear as the new current line number.

D. LOCATE REQUESTS

1. The manuscript is said to be "located" at the position marked by the cursor in the current line. For reading or editing, it may be relocated at any time.

2. The manuscript may be relocated to display any given line by typing the META arrow, the line number of the line to be displayed, and EOL.

For example, to locate between lines 104 and 105, type

$\rightarrow 104_{EOL}$

which will cause LAP6W to display lines through 104 and to display 105 as the current line number (Figure 2).

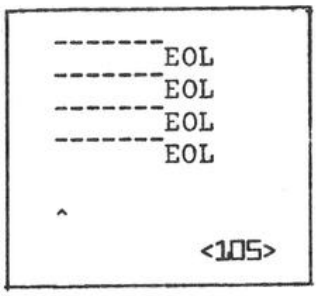


Figure 2. Locating at Line 105

3. The manuscript may also be relocated by using the following undisplayed key combinations:

   a. CONTROL Q [CASE O] Forward one page (begin display with next line).
      CONTROL A [CASE Q] Backward one page (make the line number of the top line on the scope the current line number).

   b. CONTROL W [CASE 1] Forward one line.
      CONTROL S [CASE W] Backward one line.

   These key combinations locate the manuscript in the appropriate direction so that a new manuscript line can be entered.

   For example, if the user strikes CONTROL S [CASE W] while the manuscript is located as in Figure 1, the display would consist of lines 56-62 (as currently shown); the current line number would be number 63.

   If the user strikes CONTROL S [CASE W] while the manuscript is located as in Figure 2, the current line number would become number 104 and lines 100-103 would also be displayed.

   c. CONTROL E [CASE 2] Forward one character.
      CONTROL D [CASE R] Backward one character. (CASE E on μ-LINC 300)

   These key combinations locate the manuscript one character in the appropriate direction.

   For example, if the user strikes CONTROL E [CASE 2] while the manuscript is located as in Figure 1, the manuscript would be relocated as in Figure 3. Note that the result would be the same if the user were to strike EOL and then strike CONTROL E [CASE 2] or if he were to strike CONTROL W [CASE 1] and then CONTROL E [CASE 2].

```
#LF29Z  STC *+7
        JMP TEST-2
        LDA;
            MTB
        STA
        .EXIT
                <64>
```

Figure 3. Manuscript Display after Relocation

Note that the magnitude and direction of the manuscript relocation is directly related to the physical keyboard position of the above control keys, and that the keys are grouped together for easy typing while watching the scope:

(Teletype Keyboard)

	Page	Line	Character
Forward	Q	W	E
Backward	A	S	D

(LINC Keyboard)

	Page	Line	Character
Forward	O	1	2
Backward	Q	W	R (E on μ-LINC 300)

d. EOL: End-of-Line

Striking EOL locates the manuscript after the last character on the current line. If the first character of the line is the META arrow, the current line is interpreted as a meta-command (see pages 4 and 11). If the current line is not empty, an EOL is placed at the end of the current line and the current line number increases by one. LAP6W will then be located on an empty current line.

e. Before the current line number is changed due to a key combination as described in (b) and (c) above, an EOL is placed at the end of the line if it is not empty.

4. The STRING SEARCH meta-command (see page 22) may be used to find given sequences of characters in the current manuscript.

5. Since the user will seldom know the exact line number of the line(s) he would like to see, $\rightarrow LN_{EOL}$ can be used to locate the general area, and the above key combinations can be used to "zero in".

6. To locate at the end of the manuscript, request any line number (octal) larger than the last line number. To locate at the beginning of the manuscript, type $\rightarrow 0_{EOL}$. ($\rightarrow 7777_{EOL}$ is equivalent to $\rightarrow 0_{EOL}$.)

E. EDITING

1. Characters or manuscript lines may be added or deleted wherever the manuscript is located. An entire manuscript may be deleted or deletions may be done a frame at a time, a line at a time, or a character at a time.

a. To add characters to the manuscript, locate the manuscript at the position in which you desire to add characters (see section III.D, p. 5) and then type the desired characters. Remember that when EOL is struck while characters are to the right of the cursor on the current line, LAP6W locates the manuscript after the last character of the current line before placing the EOL on the line.

b. Striking CONTROL SHIFT K [CASE K] will delete ("kill") the entire manuscript. This key combination can only be given on an otherwise empty manuscript line. (Note: executing a START 20 from the console will also delete the entire manuscript.)

c. Striking CONTROL P [CASE P] will delete all the lines on the page currently displayed on the scope. This key combination can only be given on an otherwise empty manuscript line.

For example, if the user strikes CONTROL P [CASE P] while the manuscrip is located as in Figure 2, lines 101-104 would be deleted from the manuscript, the display would show lines 75-100, and line 101 woul be the current line.

d. Striking CONTROL L [CASE L] will delete the information on the current line. If no information has been entered on the current line, the previous line will be deleted. In either case, the line number of the deleted line is retained as the current line number.

   For example, in Figure 1, striking CONTROL L [CASE L] will delete "STA" on line 63, leaving 63 as the current line number. Striking CONTROL L [CASE L] again will delete "MTB" on line 62; all the lines will move down one space, line 55 will appear as the first line displayed, and 62 will be the new current line number.

e. Striking DELETE will delete the character at the left of the cursor. If no characters of the current line are to the left of the cursor, the EOL of the previous line will be deleted, the current line number will be decremented by one, and the current line will become the concatenation of the former previous line and the former current line.

   For example, if the user strikes DELETE while the manuscript is located as in Figure 3, the EOL of line 63 would be deleted and the manuscript would be located as in Figure 4.

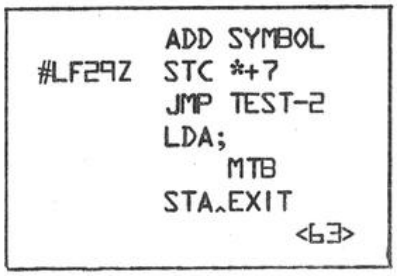


Figure 4.

2. Striking CONTROL X [CASE S] will split the current line at the cursor. If there are any characters to the left of the cursor, they will become the previous line and the current line number will be incremented by one. For example, if the user strikes CONTROL X [CASE S] while the manuscript is located as in Figure 4, the manuscript would be relocated as in Figure 3.

3. Manuscript lines following the point at which changes are made are automatically renumbered by LAP6W if necessary.

4. The LAP6W system tape will move frequently, but briefly, while locating or editing. It will, however, move at unpredictable times, perhaps even when a manuscript line or meta-command is being entered. This is indicative of normal operation and requires no user action.

IV. Files

Any tape on any unit may be used as a LAP6W file. The meta-commands described in Section V use the tapes in this way.

A tape need not have a system on it to be a file tape. In this case the entire tape may be used for filing. (See Section B below)

An entry in a LAP6W file may be designated as either a manuscript or a binary program. A file may contain only binary programs, only manuscripts, or both.

A. FILE INDEX

1. Location

When a tape is used as a file, blocks 376-377 are automatically reserved for an Index in which information about the file entries is recorded.

2. Label and Bounds

The Index of every file contains the bounds of the file and an eight-character internal label provided by the user. The label has the same format as an entry name (see below).

3. Entry Names

a. A file entry must be given a name at the time it is filed. In the Index one name describes one manuscript or one binary program, or both. A full Index, 63 names, therefore, describes a maximum of 126 entries. A binary program given the same name as a manuscript need not be the same program.

b. An entry name can be any combination of displayable characters (except comma) on the keyboard, as long as the first character is not (%)(57_8) and the name includes at least one non-octal character other than "space".

c. Names can be no more than eight characters long. Spaces in the middle or at the end are significant. Names may not have leading spaces.

B. FILE BOUNDS

1. The bounds of the reserved area within which entries will be filed, as contained in the Index (see Section IV.A.2 above), may be specified by the user via the FILE-ONLY INDEX meta-command (see page 31) or through use of the program GASTMB [GASTCL] (LCF Program UP-039-03). If the Index is a system Index (i.e., the Index was created by GASTMB [GASTCL]) the lower file is as follows:

System	Blocks (without PDP-8 metas)	Blocks (with PDP-8 metas)
Classic LINC, μ-LINC 100	0-374	not available
μ-LINC 300	0-374	not available
LINC-8	5-374	5-356
PDP-12	2-374	2-356

and the upper file is in 600-777 (inclusive).

In this case, the area between the files except blocks 376 and 377 may be used for a copy of LAP6W or in any way the user sees fit. If the FILE-ONLY bounds are selected (i.e., the Index was created via the FILE-ONLY INDEX meta-command), the lower file consists of blocks 0-375 inclusive and the upper file consists of blocks 400-777 inclusive.

C. FILE ENTRY PLACEMENT

1. Within the reserved area, a file entry is automatically saved as close as possible to the Index at block 376.

2. A file entry is always saved in contiguous blocks. Block 777 is not contiguous to block 0.

V. Meta-Commands

The following meta-commands all use the tape(s) or scope, or both, for their execution. Most return automatically to the manuscript display when the operation is finished. However, they do not always return with the manuscript located at the line which was current when the meta-command was given.

A. The meta-commands, with the exception of ADD MANUSCRIPT, do not change the current manuscript in any way. (Except perhaps to relocate it.)

B. Meta-commands may be given at the beginning of any manuscript display line. They may be given in any order.

C. If a legally stated meta-command or phase thereof cannot be executed, NO will appear on the scope. Strike EOL to continue. The current manuscript and all files are still intact. (See Chart V, p. 65 to explain the NO.)

D. Meta-commands must be two letters, followed by the necessary arguments in the order *LNA, LNB, NAME, UNITA, UNITB.*

1. *LNA* and *LNB* usually denote manuscript line numbers. *NAME* usually means the name of a file entry. *UNITA* or *UNITB* means the tape unit, either 0, 1, 4, or 5.

2. When there are two or more arguments, they must be separated by commas. *NAME* must not begin with (%), must be no more than eight characters, must contain a non-octal character other than "space", and must always be terminated with a comma.

3. Spaces are permitted almost anywhere. Leading spaces in *NAME* are not significant; embedded and trailing spaces are.

4. Parentheses in the following meta-command formats enclose optional parameter fields (i.e., the parameter and its delimiting comma). These fields need not be typed when entering a meta-command. Do not type parentheses in the actual meta-commands.

5. When specifying an optional parameter, fields enclosed together with the desired parameter field must also be typed (e.g., in the meta-command →PM (*LNA*,(*LNB*,))(*NAME*,)*UNITA* the field "*LNA*," must be present whenever "*LNB*," is desired).

6. The numeric parameters (*LNA*, *LNB*, *UNITA*, and *UNITB*) need not be specified even though the corresponding field is specified.

   If *UNITA* or *UNITB* is left out, the current system unit (i.e., the unit that LAP6W is running on) is inferred (e.g., →SM *NAME*,$_{EOL}$ or →SM *NAME*,1,$_{EOL}$)

   If the line number *LNA* is left out, line number 1 is inferred.
   If the line number *LNB* is left out, the last manuscript line number is inferred (e.g., →PM,,*NAME*,1$_{EOL}$ is equivalent to →PM *NAME*,1$_{EOL}$).

7. If the field ",*UNITB*" is required but left out, the current system unit is inferred for *UNITB* (e.g., →CM *NAME*,4$_{EOL}$ is equivalent to →CM *NAME*,4,$_{EOL}$).

E. If a tape on which a manuscript or binary is to be filed appears to have no Index, the display

NO INDEX ON UNIT *u*

will appear, where *u* indicates the tape in question. Strike any key to go to the next phase or return to the manuscript display. Use the meta-command FILE-ONLY INDEX (see page 31) or the program GASTMB [GASTCL] to create an Index.

F. The REPLACE display appears when an entry in the file is about to be replaced. It is of the form

REPLACE *xx NAME*,*u* <#>

where:

xx is the type of entry about to be replaced:
MS for manuscript, BI for binary,

NAME is the name of the entry, and

u is the file unit.

Strike (#) to replace the entry; strike EOL to go on to the next phase or return to the manuscript display.

G. META-COMMAND REPERTOIRE

1. ADD MANUSCRIPT →AM (*BN*,)*UNITA*$_{EOL}$

or: →AM (*NAME*,)*UNITA*$_{EOL}$

Any manuscript may be added to the current manuscript.

a. The manuscript to be added is identified by its first block number *BN*, on any tape, *UNITA*, or by *NAME* in the file on *UNITA*. If *NAME* or *BN* is not specified, the manuscript to be added is in the working area on *UNITA*.

b. The incoming manuscript is added to the current manuscript at the current line. Thus, the location of the current manuscript is relevant to the execution of the meta-command.

c. After the meta-command, the new current line will be at the end of the added manuscript (just as though the incoming manuscript had been added at the keyboard one character at a time). Manuscript lines which follow the added manuscript are automatically renumbered.

d. If there is no current manuscript, the added manuscript will become the current manuscript.

e. Do not attempt to add the current manuscript to itself.

f. If the manuscript in the working area was deleted (e.g., as a result of striking CONTROL SHIFT K [CASE K] inadvertently), →AM$_{EOL}$ may be used to recover the manuscript. In this case, any editing which was done immediately before killing the working area may have been lost. The user should examine the section of the manuscript in question before proceeding.

CO

2. COPY →CO$_{EOL}$

This meta-command copies the contents of any number of consecutive tape blocks on any unit to any set of consecutive tape blocks on any unit. It should not be confused with putting entries in a file and has no effect on a file Index.

When the meta-command is given, the following appears on the scope:

COPY ???? BLOCKS	← Number of blocks to be copied
FROM BLOCK ??? UNIT ?	← Present location
TO BLOCK ??? UNIT ?	← Requested location

a. Fill in the question fields as indicated. Terminate each field by striking EOL. The meta-command will be executed when the terminating EOL is struck for the last question field.

b. If the user-supplied information is illegal, the question marks will re-appear when the terminating EOL is struck for the question field. Examples of illegal responses are a non-octal number, number of blocks to be copied greater than 1000, illegal or non-existent tape unit, set of blocks which extends beyond block 777.

c. Striking DELETE will delete the answer(s) and restore the question mark(s), one question field at a time. Fill in the question field(s) again. Do any DELETEs before striking the terminating EOL for the last question field.

d. Strike CASE or (↑) at any time to interrupt the above display and return to the manuscript display.

e. To copy an entire tape, copy 1000 blocks from block 0 to block 0.

f. The blocks which are copied from are not affected.

g. CAUTION: It is never safe to copy overlapping block numbers on the same unit, regardless of the number of blocks copied. This limitation results from writing all the blocks before checking them.

EX

SU

3. EXIT →EX$_{EOL}$

4. SWITCH UNIT →SU *UNITA*$_{EOL}$

These meta-commands make it possible to leave LAP6W or leave the computer without losing the current manuscript. The next time this copy of LAP6W is used, the manuscript will still be accessible to the scope and keyboard.

a. The EXIT meta-command rewinds the tape containing the LAP6W system and then halts. Raise RESUME (push CONT on the PDP-12) to re-enter LAP6W.

b. The SWITCH UNIT meta-command causes the LAP6W system on *UNITA* to be loaded and started. If *UNITA* is not specified, the current system is re-entered.

c. LAP6W may be re-entered by executing the regular start procedure manually at the console (see page 1), or under program control as described on page 57. In either case, EXIT is good for <u>one</u> re-entry only. (Subsequent pushes of START 20 simply erase the current manuscript.)

d. There is never any reason not to EXIT from LAP6W when leaving the computer.

5. LOAD BINARY →LB (*NAME*,)*UNITA*$_{EOL}$

or: →LB (*QBBB*,*N*,)*UNITA*$_{EOL}$

This meta-command loads the binary specified into LINC memory.

a. If no parameters are specified, the "binary most recently assembled" with the AS or 8A meta-command is loaded. If only *UNITA* is specified, the "binary most recently assembled" on *UNITA* is loaded.

b. If *NAME* is specified, the binary filed as *NAME* in the file on *UNITA* is loaded.

c. If *N* does not equal zero, the *N* block binary to be loaded is assumed to start at block *BBB* on *UNITA*. The first block is loaded into quarter *Q*. If *Q* is non-zero, *BBB* must include leading zeros.

d. WARNING: No check is made for the type of binary, i.e., LINC or PDP-8. Use the PDP-8 LOAD BINARY meta-command (see page 18) to load a PDP-8 binary program.

e. All available LINC memory is loaded, starting with bank 0 (bank 1 in the LINC-8 and PDP-12). Succeeding groups of 4 blocks go into succeeding banks, i.e., blocks 0-3 into bank 0(1), 4-7 into bank 1 (2), 10-13 into bank 2 (3), etc.

f. In the multi-bank system, the lower memory bank is left set to bank 0 (bank 1 in the LINC-8 and PDP-12) and the upper memory bank to bank 1 (2).

g. Memory registers not occupied by the binary program are cleared (set to +0). Exception: if the binary program itself does not occupy any registers in quarters 3 thru 7, registers 1770-1777 will not be cleared.

h. LB uses register 0 after the binary program is read into the memory. Therefore, if the program requires an initial value in register 0, it must be reset.

i. LB starts the program in location 1 of bank 0 (bank 1 in the LINC-8 and PDP-12). The user can ignore this feature by simply assigning nothing to register 1, in which case the computer will halt at 1. The program must then be started from the console. The following hints may help:

   i. If the first instruction, or a JMP to the first instruction, is put in register 1 (i.e., in the program manuscript following an $1 statement), the program will start automatically as soon as it is read in.

ii. If nothing is put in register 1 and the first instruction is put in register 2, the computer will halt at register 1, but the program can be started by raising the RESUME lever. This is a helpful procedure when there are switches to be set or tapes to change before the program can be run.

j. LB leaves the first block number of the program's tape location in the left 9 bits of the Z register. It leaves the corresponding unit number in the right 3 bits of Z (to make it easier to overlay programs which have no fixed tape location).

k. LB EXITs from LAP6W before reading the binary program. The current manuscript may be recovered by executing the regular console start procedure or by re-entering the system under program control as described on page 57.

8B

6. PDP-8 LOAD BINARY →8B (*NAME*,)*UNITA*$_{EOL}$

(LINC-8 & PDP-12 Only) or: →8B (*QBBB*,*N*,)*UNITA*$_{EOL}$

This meta-command loads the specified binary into PDP-8 memory.

a. If no parameters are specified, the "binary most recently assembled" with the AS or 8A meta-command by this copy of LAP6W is loaded. If only *UNITA* is specified, the "binary most recently assembled" on *UNITA* is loaded.

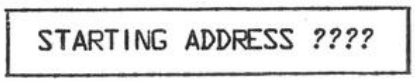
b. If *NAME* is specified, the binary filed as *NAME* in the file on *UNITA* is loaded.

c. If *NAME* is not specified and *N* is specified to be non-zero, the *N*-block binary starting at *BBB* on *UNITA* is loaded. Block *BBB* will be loaded into quarter *Q*. If *Q* is non-zero, *BBB* must include leading zeros.

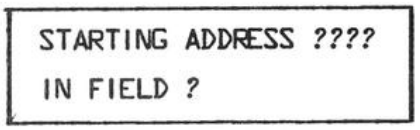
d. WARNING: No check is made for the type of binary, i.e., LINC or PDP-8. Use the LOAD BINARY meta-command (see page 16) to load a LINC binary program.

e. All available PDP-8 memory will be loaded, if possible. Loading starts at the quarter specified by *Q* (quarter 0 is locations 0-377 of field 0, quarter 1 is locations 400-777, etc.), and continues to the end of available memory or to the end of the binary program on tape, whichever occurs first. However, locations 7600 thru 7777 of field 0 are never loaded, because the loading routine is there.

f. One of the following displays will appear:



(a)



(b)

(Display (b) appears if the computer in use has more than one 4096 word field of memory and the program is more than 20_8 blocks long; otherwise display (a) appears.)

Enter the starting address and strike EOL, then if display (b), enter the field and strike EOL. If other than octal digits are entered or the field entered is not available, the erroneous questions will re-appear immediately. If nothing is entered for the starting address, it is assumed to be 200. If nothing is entered for the field, it is assumed to be 0. Strike (↑) at any time to return to the manuscript display.

The PDP-8 program will be started by a JMP to the specified address with the instruction and data field registers set to the specified field (0 for display (a)).

Location 7777 in field 0 contains a PDP-8 HLT instruction. It may be used as a starting address.

g. Memory registers not occupied by the program are not changed. The PDP-8 loading routine occupies locations 7600-7777 in field 0.

h. On entry to the PDP-8 program, the LINC subsystem has been de-selected and the PDP-8 accumulator and LINK-bit cleared.

i. PDP-8 LOAD BINARY EXITs before loading the PDP-8 program. The manuscript display may be re-entered by reloading LAP6W or under PDP-8 program control by jumping to 7600 in field 0 (which causes PROGOFOP (TRAP in the PDP-12) to be reloaded).

j. In the PDP-12, this meta-command may be used to load LINC programs which need to run in bank 0. In this case, it is necessary to have a LINC instruction (6141) at the starting address to switch the computer to LINC mode.

k. The PDP-8 loader in locations 7600-7777 of field 0 may be used to load overlays into field 0 by setting up the following locations and jumping to 7616:

7766	First block number
7776	Starting address (i.e., return address)
7770	First word address
7757	0 for tape units 0 and 1; 1 for tape units 4 and 5.
7760	$2+(4000_8)\cdot$(tape unit bit); i.e., 4002 for units 1 and 5, 0002 for units 0 and 4.
7774	Two's complement of number of words to load.
7754	$6202_8 + 10_8 \cdot$ (Field of starting address).

Caution: Do not read overlays into page 37 on the LINC-8. Do not read overlays into pages 36-37 (quarter 17) on the PDP-12.

Initially, locations 7766, 7757 and 7760 are left set (as explained above) to the values for the PDP-8 program just loaded. These values may be used to set up overlays for programs which have no fixed tape location.

PM

7. PRINT MANUSCRIPT →PM (*LNA*,(*LNB*,))(*NAME*,)*UNITA*$_{EOL}$

The PRINT MANUSCRIPT meta-command is used to list manuscripts on the Teletype.

a. If *NAME* and *UNITA* are not specified, the current manuscript is printed. If only *UNITA* is specified, the manuscript in the manuscript working area on *UNITA* is printed. If *NAME* is specified, the manuscript filed as *NAME* in the file on *UNITA* is printed.

b. If no line numbers are given, the entire manuscript is printed. When line numbers are specified, that portion of the manuscript from *LNA* thru *LNB* is printed. When only *LNA* is specified, *LNB* is assumed to be the last line number of the manuscript. When only *LNB* is specified, *LNA* is assumed to be 1.

c. The *NAME*, if any, and the page number in decimal are typed in the title at the top of each page. The line number of the first manuscript line which appears on that page is typed in the second line of the title. The title has the form

PM OF *NAME* PAGE *nn*
LN=*llll*

d. The following sense switches may be used to control the format of the printed manuscript:

i. Sense switch 5

down: Lines beginning with (#), ([) or ($) are printed starting at the left margin; other lines are indented 8 spaces (6 on the classic LINC)

up: All lines are printed starting at the left margin.

ii. Sense switch 4

down: Manuscript lines are single spaced.

up: Manuscript lines are double spaced.

iii. Sense switch 3

down: The manuscript line number of each line is printed to the left of the line.

up: No manuscript line numbers are printed.

e. Manuscript lines which are too long to fit on one Teletype line are automatically carried over to the next Teletype line without splitting words.

f. The correspondence between the characters and their displayed and typed notation is shown in Chart IV on page 64.

g. Strike EOL to interrupt printing and return to the manuscript display.

h. Printing time is approximately one minute per page for program manuscripts.

```
    PM OF TEST MS     PAGE 01
LN=0001

0001 [SAMPLE WISAL PROGRAM; GENERATES LINC OBJECT CODE
0002 $200:SAM+5 [SAM IS AN OP CODE
0003 #BEG1    LDA;
0004              VARIBL   [PROGRAM SYMBOL
0005          ADA;
0006              *+4
0007 #X6      JMP X6-10
0010          JMP BEG1
0011          VARIBL=ADD  [EQUALITY [=]
0012 #FILE    [NULL STATEMENT (FILE=DATA)
0013 #DATA    -4000   [OCTAL NUMBER
0014          192.    [DECIMAL NUMBER
0015 $300
0016          LDA;
0017              ''   [EMPTY CHARACTER CONSTANT
0020          STC FILE+1
0021 #TEXT    SAE;
0022              'A' [ZERO FILL
0023          JMP BEG1
0024          JMP X6-10
0025 $2000
0026 [LABELS
0027 #R3      "GRAPH #?" [EXAMPLES OF TEXT STATEMENTS
0030 #X4      "DATA:?
0031          RUN ?"
0032          LDA;
0033              "Y"   [77 FILL
0034          ADA;
0035              BEG1-X6
0036 $R3-1000
0037 #F2      ADD 3
0040          JMP F2+2  [NOTE FIRST VALUE FOR F2 IS USED
0041          F2=40  [F2 BECOMES DOUBLY-DEFINED HERE
0042          ADD F3  [F3 IS UNDEFINED; ZERO VALUE USED
0043 $3777
0044          ;+;-X6+*-LAM+RDC   [MEANINGLESS, BUT PERMISSABLE
0045          LDH
0046              4/R3+3
0047          STC FILE
0050          TESTER=VALUE [UNDEFINED ERROR
0051 #TABLE1 0,1,1,0,1
0052          'ABC' [SYNTAX ERROR  [END SAMPLE MS
```

Figure 5. Print Manuscript Output

(Sense switches 3, 4 and 5 down)

SS

8. STRING SEARCH →SS (*LNA*,(*LNB*,))*STRING*,$_{EOL}$

This meta-command locates the manuscript immediately after the specified character string *STRING*.

a. *STRING* is a string of no more than eight characters. Leading, trailing and embedded blanks are significant. All characters but EOL and (,) are permissible.

b. The search for *STRING* begins at manuscript line *LNA*. If *LNA* is not specified, then the current line number is used for *LNA*.

c. The search for *STRING* will not extend beyond manuscript line *LNB*. If *LNB* is not specified, or *LNB*<*LNA*, or *LNB* is greater than the last manuscript line number, then the last manuscript line number is used for *LNB*.

d. When the manuscript display reappears, the manuscript will be located immediately after the first occurrence of *STRING* encountered during the search. If *STRING* is not found, the manuscript will be located on an empty line after line *LNB*.

e. While searching, strike a key to terminate the search. The manuscript will be located between two manuscript lines and the key will be analyzed as keyboard input.

f. For example, if the manuscript is located as in Figure 3 and the user types

→SS 56,STA,$_{EOL}$

or

→SS 56,,STA,$_{EOL}$

The manuscript would be relocated as in Figure 1 (see page 5).

After using STRING SEARCH, the manuscript may be located immediately following the next occurrence of the character string *STRING* by using the CONTINUE SEARCH meta-command.

9. CONTINUE SEARCH →CS *LNA*$_{EOL}$

This meta-command locates the manuscript immediately after the character string specified as *STRING* in the last STRING SEARCH. Use CONTINUE SEARCH only if the previous meta-command was STRING SEARCH or CONTINUE SEARCH.

a. The search for *STRING* begins at manuscript line *LNA*. If *LNA* is not specified, then the current line number is used for *LNA*.

b. The search will not extend beyond the line number specified as *LNB* in the last STRING SEARCH. If *LNB* was not specified, or *LNB*<*LNA*, or *LNB* is greater than the last manuscript line number, then the last manuscript line number is used for *LNB*.

c. When the manuscript display reappears, the manuscript will be located immediately after the first occurrence of *STRING* encountered during the search. If *STRING* is not found, the manuscript will be located on an empty line after line *LNB*.

d. If CONTINUE SEARCH is used after a meta-command other than STRING SEARCH or CONTINUE SEARCH, the specified parameters *NAME* and *LNB* will be used in place of the STRING SEARCH parameters *STRING* and *LNB*.

e. While searching, strike a key to terminate the search. The manuscript will be located between two manuscript lines and the key will be analyzed as keyboard input.

10. LOAD META-PROGRAM →+*MP* ((*N1*,(*N2*,))*STRING*,*UNITA*(,*UNITB*))

The use of meta-programs allows the user to expand the meta-command capabilities of LAP6W to fit his particular needs. A meta-program is a LINC binary program which, instead of being loaded via the LB meta-command, is loaded using a "meta-program command". This allows the user to pass up to five parameters to the meta-program: two octal numbers (*N1* and *N2*), a string of up to eight LINC characters (*STRING*), and two unit numbers (*UNITA* and *UNITB*).

a. The meta-program to be loaded must be a LINC binary program assembled starting in quarter 0. It must be filed on the system unit from which it is to be loaded.

b. The meta-program's name must be of the form "+*MP*-----", where *M* and *P* are any displayable LINC characters, followed by zero or more characters. That is, the name must begin with (+) and must be at least three characters long. The remaining five characters are arbitrary, and may be used to further identify the meta-program.

c. There must be no other file entry name with the same first three characters in the Index.

d. A meta-program is loaded by typing the meta arrow, followed by the first three characters of the name, followed by the parameters.

e. LAP6W loads a meta-program by reading its first block into quarter 0 of bank 0 [bank 1 on the LINC-8 and PDP-12] and jumping to location 20. The accumulator will contain the first block number of the meta-program in the left 9 bits and the system unit number in the right 3 bits. If the meta-program is over one block long, it must read in the remainder of the routine itself.

f. The following parameters are left in core:

```
2367:  System type (see below)
2370:  N1 (-0 if not entered)
2371:  N2 (-0 if not entered)
2372 ⎫
  :  ⎬ STRING (77 fill; -0 if not entered)
2375 ⎭
2376:  UNITA (ROL'ed 3; system unit if not entered)
2377:  UNITB (ROL'ed 3; -0 if not entered)
```

The system type gives the type of LAP6W system which loaded the meta-program:

```
-3:  Classic LINC and μ-LINC 100
-2:  μ-LINC 100 only
-1:  Classic LINC only
 1:  LINC-8 with PDP-8 metas
 2:  LINC-8 without PDP-8 metas
```

3: PDP-12 with PDP-8 metas
4: PDP-12 without PDP-8 metas
5: μ-LINC 300
6: Expanded Memory Classic LINC

g. Also left in core are LAP6W's character display grid at location 2000, the Index on the system unit in quarters 6 and 7, and a routine to display "NO" on the scope and return to the system at location 400.

h. The meta-program loader EXITs from LAP6W before reading the binary program. The current manuscript may be recovered by executing the regular console start procedure or by re-entering the system under program control as described on page 57.

CF

11. COPY FILE

→CF *UNITA*(,*UNITB*)$_{EOL}$

→CF *BN*,*UNITA*,*UNITB*$_{EOL}$

The COPY FILE meta-command files a copy of all the entries in the file on *UNITA* into the file on *UNITB*. It does not, however, disturb or replace any entries which are already in the file on *UNITB*. Thus, a file may be reorganized by combinations of the commands CM, CB, or CP, to move specific entries, and CF to move all the remaining entries.

a. Entries are filed as close as possible to the Index in the file on *UNITB*. Any gaps created by former deletions are used; this "packs" the file. The file on *UNITA* is not changed.

b. If *BN* is not specified, the Index of the file on *UNITA* is considered to be in the standard location (blocks 376-377). If *BN* is specified, the Index of the file on *UNITA* is considered to be in blocks *BN* and *BN*+1. If *BN* is not between 0 and 776 inclusive, "NO" will be displayed. In all cases, the Index of the file on *UNITB* is considered to be in the standard location (blocks 376-377).

c. If COPY FILE is used to merge two files, "NO" will be displayed if the entries which are to be copied from the file on *UNITA* do not all fit into the file or Index on *UNITB*. As many entries as possible have been copied. Inspection of the two Indexes will show which entries were not copied.

d. If *UNITA* and *UNITB* have the same value, the Index and the file remain unchanged.

e. Strike EOL to interrupt the meta-command and return to the manuscript display. This has no effect once the Index has been updated.

CM

CB

FROM TO

12. COPY MANUSCRIPT →CM (*BBB*,*N*,)*NAME*,*UNITA*(,*UNITB*)$_{EOL}$

13. COPY BINARY →CB (*QBBB*,*N*,)*NAME*,*UNITA*(,*UNITB*)$_{EOL}$

These meta-commands move an entry in one file into another file or from somewhere on tape into a file. The meta-commands are identical except for the type of file entry copied.

a. If *N* equals zero or is not specified, a copy of the manuscript (binary) filed as *NAME* in the file on *UNITA* is filed as *NAME* in the file on *UNITB*.

b. If *N* does not equal zero, the manuscript (binary) is considered to occupy *N* blocks starting at block *BBB* on *UNITA*. If it is a binary, its first block loads into quarter *Q*. If *Q* is non-zero, *BBB* must include leading zeros. A copy of the manuscript (binary) is filed as *NAME* in the file on *UNITB*.

c. Strike EOL to interrupt these meta-commands and return to the manuscript display. This has no effect once the Index has been updated.

d. Appropriate caution should be exercised when assigning the same value to *UNITA* and *UNITB* since the gap selected in the file may overlap the current location of the manuscript (binary) in the file.

CP

14. COPY PROGRAM →CP *NAME*,*UNITA*(,*UNITB*)$_{EOL}$

a. The COPY PROGRAM meta-command has two phases:

i. A copy of the manuscript filed as *NAME* in the file on *UNITA* is filed as *NAME* in the file on *UNITB*.

ii. A copy of the binary filed as *NAME* in the file on *UNITA* is filed as *NAME* in the file on *UNITB*.

b. Strike EOL to interrupt the meta-command. This has no effect on a phase once the Index has been updated.

c. Appropriate caution should be exercised when assigning the same value to *UNITA* and *UNITB* since the gap selected in the file may overlap the current location of the manuscript (binary) in the file.

15. DISPLAY INDEX →DX (*NAME*,)*UNITA*$_{EOL}$

This meta-command displays the contents of the Index of the file on *UNITA*. Copies of the Index may be printed on the Teletype and unwanted entries may be deleted. An example of the display is:

NAME	*t*	*label*	
NAME		BN	#B
MSDIS	M	276	100
	B	400	20
CFCO	M	420	44
	B	464	5
12345678	M	471	65
GASTX	M	230	46
	B	2223	5
PRGFP X8	B	214	7
UNITONE	B	556	50
SEQ+LILR	M	115	77

Figure 6. Index Display

where: *NAME* is an external label specified as *NAME* at this call to DX.

t is the type of file bounds (see page 9). F = File-Only, S = System, O = Other bounds supplied by the user.

label is the internal label supplied by the user through use of the FX or NX meta-commands (see page 31).

In this example, the manuscript (M) named MSDIS starts at block number (BN) 276 and is 100 (octal) blocks long (#B). The binary program (B) named MSDIS is in blocks 400-417, etc.

a. Ten (25 in the PDP-12) entries are displayed per frame.

b. A manuscript (M) and a binary program (B) with the same name (e.g., MSDIS, CFCO, GASTX) always appear together, manuscript first, in the Index display. Thus, this file contains no manuscript named PRGFP X8 or UNITONE and no binary program named 12345678. No statement can be made about a binary program named SEQ+LILR without displaying at least one more entry.

c. If the first block of a binary is to be loaded into a quarter other than quarter zero, the quarter number will immediately precede the block number (e.g., the binary of GASTX would be loaded in quarters 2 through 6).

d. While displaying, the keys are interpreted as follows:

CONTROL Q [CASE 0]:	Forward one frame
CONTROL W [CASE 1]:	Forward one entry
CONTROL A [CASE Q]:	Backward one frame
CONTROL S [CASE W]:	Backward one entry
DELETE:	Delete the last entry displayed. (e.g., the manuscript named SEQ+LILR in Figure 6). Whatever is deleted from the Index display will be deleted from the file if (#) (see below) is struck. The file space is then available for later use.
EOL:	Return to the manuscript display, ignoring any deletions.
R:	Restore the Index display, i.e., ignore all deletions.
#:	Make the deletions permanent and return to the manuscript display.
P:	Print a copy of the Index in its currently edited form on the Teletype.
Other:	Ignore.

LINCWISH V2 needs CASE#

e. While printing, the keys are interpreted as follows:

EOL:	Return to the manuscript display, ignoring any deletions.
P:	Print a copy of the Index in its currently edited form on the Teletype.
Other:	Return to the Index display.

An example of a printed Index is:

INDEX OF NAME F LABEL

	MANUSCRIPT		BINARY	
NAME	BN	#B	BN	#B
MSDIS	276	1ØØ	4ØØ	2Ø
CFCO	420	44	464	5
12345678	471	65		
GASTX	23Ø	46	2223	5
PRGFP X8			214	7
UNITONE			556	5Ø
SEQ+LILR	115	77	626	5
ASDIS+PM	633	43	1Ø5	1Ø

Figure 7.

FX

NX

16. FILE-ONLY INDEX →FX (*NAME*,)*UNITA*$_{EOL}$

17. NAME INDEX →NX *NAME*,*UNITA*$_{EOL}$

These two meta-commands allow the user to create a file-only Index and/or specify an internal label for an Index (see page 9).

a. If there is an Index on *UNITA*, then the following display will appear:

```
REPLACE
   oldtype INDEX:  oldlabel
WITH
   newtype INDEX:  NAME
ON UNIT u ?     <#=YES>
```

where:

oldtype is SYSTEM if the file currently has system bounds, i.e., it was created via GASTMB [GASTCL] (see page 9).
FILE if the file currently has file-only bounds.
OTHER if the file currently has any other bounds.

oldlabel is the current internal label.

newtype is FILE if the file is about to be given file-only bounds(FX).
oldtype if only the file label is being changed (NX).

NAME is the *NAME* supplied to the meta-command (it will be the new internal label).

u is *UNITA*, the unit of the tape whose Index label is to be changed.

Strike (#) to replace the *oldtype* bounds with the *newtype* bounds and the old internal label *oldlabel* with *NAME* in the Index on *u*.
Strike EOL to leave the Index unchanged. In either case, control then returns to the manuscript display.

FX

NX

b. If there is no Index on *UNITA*, the following display will appear:

```
REPLACE
  NO INDEX
WITH
  FILE INDEX:  NAME
ON UNIT u ?    <#=YES>
```

where the variable fields are described above.

Strike (#) to create a file-only Index on *UNITA*. This Index will contain no entries, and will have an internal label of *NAME*. Strike EOL to leave the tape on *UNITA* unchanged. In either case, control then returns to the manuscript display.

18. SAVE BINARY →SB *NAME*,*UNITA*(,*UNITB*)$_{EOL}$

a. The SAVE BINARY meta-command has two phases:

i. The "binary most recently assembled" with either the AS or the 8A meta-command by this copy of LAP6W is saved as *NAME* in the file on *UNITA*. If the assembly was interrupted after the production of binary code started or a symbol table full condition occurred (see page 37) there will be no "binary most recently assembled" to be saved.

ii. If *UNITB* is specified, a copy of the binary filed as *NAME* in the file on *UNITA* is filed as *NAME* in the file on *UNITB*. If *UNITB* is specified, the copy will be performed even if the "binary most recently assembled" was not saved in phase (i) due to the user's reply to the REPLACE display (see page 12) or there was no "binary most recently assembled" to be saved.

b. Only relevant, but inclusive, quarters are saved. However, if the first relevant quarter is quarter 10 or greater, all quarters from quarter 7 thru the last relevant quarter are saved. For example, if the program was written to occupy quarters 1 and 4, quarters 1 thru 4 will be filed in four successive tape blocks. (The second and third blocks will contain all zeros.) However, if the program occupies quarters 11 and 14, quarters 7 thru 14 will be filed in six successive tape blocks.

c. Strike EOL to interrupt the meta-command. This has no effect on a phase once the Index has been updated.

19. SAVE MANUSCRIPT →SM (*LNA*,(*LNB*,))*NAME*,*UNITA*(,*UNITB*)$_{EOL}$

a. The SAVE MANUSCRIPT meta-command has two phases:

i. If no line numbers are specified, the current manuscript is saved as *NAME* in the file on *UNITA*.

If *LNA* is specified as being non-zero, part of the current manuscript is saved as a separate, new manuscript called *NAME* in the file on *UNITA*. The line numbers, inclusive, indicate the part to be saved. When only one line number is given, *LNB* is assumed to be the last line number of the current manuscript.

NOTE: Unlike LAP6, LAP6W does not use the working area on the opposite unit of the tape transport when saving part of the current manuscript.

ii. If *UNITB* is specified, a copy of the manuscript filed as *NAME* in the file on *UNITA* is filed as *NAME* in the file on *UNITB*. If *UNITB* is specified, the copy will be performed even if the current manuscript was not saved during phase (i) due to the user's reply to the REPLACE display (see page 12).

b. Strike EOL to interrupt the meta-command. This has no effect on a phase once the Index has been updated.

SP

20. SAVE PROGRAM →SP (*LNA*,(*LNB*,))*NAME*,*UNITA*(,*UNITB*)$_{EOL}$

a. The SAVE PROGRAM meta-command has four phases:

i. The current manuscript is saved as *NAME* in the file on *UNITA*. The line numbers are interpreted as in SAVE MANUSCRIPT.

ii. The "binary most recently assembled" is saved as *NAME* in the file on *UNITA*.

iii. If *UNITB* is specified, a copy of the manuscript filed as *NAME* in the file on *UNITA* is filed as *NAME* in the file on *UNITB*. If *UNITB* is specified, the copy will be performed even if the current manuscript was not saved during phase (i) due to the user's reply to the REPLACE display (see page 12).

iv. If *UNITB* is specified, a copy of the binary filed as *NAME* in the file on *UNITA* is filed as *NAME* in the file on *UNITB*. If *UNITB* is specified, the copy will be performed even if the "binary most recently assembled" was not saved in phase (ii) due to the user's reply to the REPLACE display (see page 12) or there was no "binary most recently assembled" to be saved.

b. Strike EOL to interrupt the meta-command. This has no effect on a phase once the Index has been updated.

AS

8A

21. ASSEMBLE →AS (*LNA*,(*LNB*,))(*NAME*,)*UNITA*$_{EOL}$

22. PDP-8 ASSEMBLE →8A (*LNA*,(*LNB*,))(*NAME*,)*UNITA*$_{EOL}$

The specified manuscript is assembled into LINC (→AS) or PDP-8 (→8A) binary object code. Information about the program is displayed. Conventions used in assembling program manuscripts are specified in section VI. The symbolic operation codes and their octal equivalents are given in Charts VI and VII.

a. If *NAME* and *UNITA* are not specified, the current manuscript is assembled. If only *UNITA* is specified, the manuscript in the manuscript working area on *UNITA* is assembled. If *NAME* is specified, the manuscript filed as *NAME* in the file on *UNITA* is assembled.

b. The line numbers, *LNA* and *LNB*, are ignored if they are specified. The entire manuscript is always assembled. If a listing or reference table is requested by striking L or R during a display (see below), the line numbers will be interpreted as arguments to the corresponding meta-command.

c. The assembled binary program is placed in the binary working area (blocks 450-467) of the LAP6W tape. The binary is positioned according to the storage locations specified by the user (see page 50). The block numbers correspond to memory quarters 0-17 respectively. A program written to be stored in memory registers in quarters 0 and 2 will be found in tape blocks 450 and 452. Locations not occupied by the binary are cleared (i.e., set to +0).

d. Strike EOL at any time to prevent the further production of binary code. The assembler will continue to the end of the manuscript and the displays (see below) will be as if EOL had not been struck. If EOL is struck after the production of binary has started, there will be no "binary most recently assembled" (see page 16) and no valid binary is left in blocks 450-467. If EOL is struck before the production of binary begins, the binary, if any, in the binary working area remains unchanged and is still the "binary most recently assembled".

e. Strike EOL again to terminate the assembly and return control directly to the manuscript display.

f. Assembler Displays

Symbol Table Full

The assembler keeps one entry in its symbol table for each error, origin statement, and user-defined symbol. The maximum number of entries in the symbol table for various system configurations is given in the following table:

System	Entries (→AS)	Entries (→8A)
Classic LINC, μ-LINC 100 (2K)	199	Not available
μ-LINC 300 (4K)	321	Not available
LINC-8, PDP-12 (4K)	321	288
μ-LINC 300 (8K)	1345	Not available
LINC-8, PDP-12 (8K)	1345	1312

If more entries are needed to complete the assembly, the following display will appear:

```
SYMBOL TABLE
FULL AT LINE
  llll
```

where: *llll* is the number of the manuscript line at which the assembler tried to make an entry in the full symbol table.

In this case:

i. There is not a valid binary in the binary working area and there is no "binary most recently assembled" (see page 16).

ii. If the errors, which will be displayed subsequently, are corrected, then it may be possible to assemble the manuscript without filling the symbol table.

iii. Since some errors are entered in the symbol table during the second pass of assembly, the line number, *llll*, may be surprisingly low.

iv. If there are no errors and the symbol table still fills, either reduce the number of symbols and/or origin statements or split the source program into two separate manuscripts.

Errors

i. A Doubly-defined Symbol is indicated by the letter D and appears with the manuscript line number of the double definition.

ii. A Syntax Error is indicated by the letter S and appears with the manuscript line number on which the error occurs.

AS

8A

iii. An Undefined Symbol is indicated by the letter U and appears with the manuscript line number on which the symbol is referenced.

iv. For example, the assembly of the manuscript in Figure 8, page 42, would yield:

```
ERRORS

41 D F2
42 U F3
50 U VALUE
52 S
```

F2 is defined previous to line 41 and on line 41.
F3 is undefined and is referenced on line 42.
VALUE is undefined and is referenced on line 50.
A Syntax Error exists on line 52.

Memory Allocation

This display shows the storage locations, inclusive, required for the binary program. Remember that the storage location and the execution location are not necessarily the same (see page 51).

For example, the assembly of the manuscript in Figure 8 would yield:

```
MEMORY ALLOCATION
 LINE  FROM-TO

    2   105- 114
   15   300- 306
   25  2000-2015
   36  1000-1002
   43  3777-4010
```

This binary program is stored in memory locations 105-114, 300-306, etc. The corresponding origin statements appeared on manuscript lines 2, 15, etc. Reading this display is the simplest way to tell whether a program exceeds a certain number of memory quarters, or whether certain portions overlap in the binary working area. This is especially crucial with "→8A" where the allocation of literal storage areas is not explicitly specified by the programmer. By perusal of this display, the programmer can detect any incursion of his object code into literal storage areas. [In classic LINC LAP6W, the "LINE" column does not appear.]

Symbols

All symbols defined via labels or equalities and all undefined symbols appear in this display in alphabetical order.

i. Each defined symbol is displayed along with its value.

ii. The first value of a multiply-defined symbol is used during assembly and is displayed.

iii. The letter U is displayed as the value of an undefined symbol (zero is used as its value during assembly).

iv. For example, the assembly of the manuscript in Figure 8 would yield:

SYMBOL	VALUE
BEG1	0200
DATA	0206
F2	1000
F3	U
FILE	0206
R3	2000
TABLE1	4003
TESTER	0000
TEXT	0303
VALUE	U

g. While displaying, the keys are interpreted as follows:

CONTROL Q [CASE 0]:	Forward to the next display, if any.
CONTROL W [CASE 1]:	Forward one frame of the current display.
CONTROL A [CASE Q]:	Back to the first display. (The errors cannot be redisplayed once the symbols have been displayed.)
CONTROL S [CASE W]:	Back to the first frame of the current display.
EOL:	Return to the manuscript display.
P:	Print a copy of the entire current display on the Teletype.
L:	List this manuscript.
R:	Print a reference table of this manuscript.
OTHER:	Ignore.

h. While printing, striking a key causes a return to the display, then the key is interpreted as in (g) above.

LI

8L

23. LIST →LI (*LNA*,(*LNB*,))(*NAME*,)*UNITA*$_{EOL}$

24. PDP-8 LIST →8L (*LNA*,(*LNB*,))(*NAME*,)*UNITA*$_{EOL}$

Any manuscript which can be assembled will be listed on the Teletype. When listing is finished, a reference table is printed using the same arguments. Examples of the two types of listings appear in Figures 8 and 9.

a. If *NAME* and *UNITA* are not specified, the current manuscript is listed. If only *UNITA* is specified, the manuscript in the manuscript working area on *UNITA* is listed. If *NAME* is specified, the manuscript filed as *NAME* in the file on *UNITA* is listed.

b. If no line numbers are given, the entire manuscript is listed. When line numbers are specified, that portion of the manuscript from *LNA* thru *LNB* is listed. When only *LNA* is specified, *LNB* is assumed to be the last line number of the manuscript. When only *LNB* is specified, *LNA* is assumed to be 1.

c. The *NAME*, if any, and the page number in decimal are typed in the title at the top of each page. The line number of the first manuscript line which appears on that page is typed in the second line of the title. The title has the form:

LI OF *NAME* PAGE *nn*
LN=*llll*

d. Three four-digit octal numbers are printed to the left of manuscript lines that generate object code: the manuscript line number, the execution location and the corresponding object code. If the execution location differs from the storage location (see page 51), an (*) will precede the printed execution location. When listing PDP-8 code (→8L) a fourth octal field, printed to the right of the above three, contains the address referenced by, or literal value appearing in, a memory reference statement.

e. If a manuscript line generates no object code, only the manuscript line number is printed with the line. The printing of the manuscript line number column may be disabled by putting sense switch 3 up.

f. If a manuscript line begins with a (#) or ($), it is printed immediately after the octal number fields; otherwise, it is indented eight [six] spaces. When possible, comments are made to start in the same column by inserting preceding blanks.

g. A single line of text may generate many words of object code. When this happens, the text line is printed adjacent to the first word of object code and succeeding words of object code are printed on succeeding lines.

h. When commas are used to terminate several statements appearing on the same manuscript line, the entire line is printed together with the execution location and object code of the first statement. Succeeding print lines contain object word locations and contents arising from the remainder of the manuscript line.

i. Before the listing starts, strike EOL to return to the manuscript display. After listing starts, strike R to stop listing and begin the reference table, or strike EOL to return to the manuscript display.

j. The symbol table used by the LIST meta-commands is obtained by running the corresponding assembler (without changing the contents of the binary working area). If the symbol table fills up during this assembly, the following message is printed at the beginning of the first page:

ST FULL AT LINE *llll*

Symbols which are defined after line *llll* will be considered as undefined (value = +0) when calculating the contents of a location.

WARNING: In this case, the binary listed does not necessarily correspond to the contents of the binary working area after assembly (see page 37).

k. The format of the listing can be controlled to some degree by including special statements in the manuscript. There are three options available: No List (%N), List (%L), and Top of Form (%T). The No list option suppresses listing until the List option is encountered. Listing continues at the beginning of a new page when the Top of Form option is encountered.

l. Manuscript lines which are too long to fit on one Teletype line are automatically carried over to the next Teletype line.

m. The correspondence between the characters and their displayed and typed notation is shown in Chart IV on page 64.

n. Listing time is approximately two minutes per page.

o. A Teletype page is exactly eleven inches long if cut at the marker which appears between pages. There are 55_{10} lines of print on each page.

p. WARNING: If the manuscript contains overlaps, the binary listed may not correspond to the binary as assembled because the LIST meta-commands assemble the manuscript and print it simultaneously.

```
    LI OF TEST MS     PAGE 01
LN=0001

0001                                    [SAMPLE WISAL PROGRAM; GENERATES LI
                                         OBJECT CODE
0002                 $200:SAM+5         [SAM IS AN OP CODE
0003 *0200 1020 #BEG1    LDA;
0004 *0201 2000              VARIBL     [PROGRAM SYMBOL
0005 *0202 1120          ADA;
0006 *0203 0207              *+4
0007 *0204 6174 #X6      JMP X6-10
0010 *0205 6200          JMP BEG1
0011                 VARIBL=ADD         [EQUALITY [=]
0012                 #FILE              [NULL STATEMENT (FILE=DATA)
0013 *0206 3777 #DATA    -4000          [OCTAL NUMBER
0014 *0207 0300          192.           [DECIMAL NUMBER
0015                 $300
0016  0300 1020          LDA;
0017  0301 0000              ''         [EMPTY CHARACTER CONSTANT
0020  0302 4207          STC FILE+1
0021  0303 1460 #TEXT    SAE;
0022  0304 0024              'A'        [ZERO FILL
0023  0305 6200          JMP BEG1
0024  0306 6174          JMP X6-10
0025                 $2000
0026                                    [LABELS
0027  2000 3245 #R3      "GRAPH #?"     [EXAMPLES OF TEXT STATEMENTS
      2001 2443
      2002 3314
      2003 2260
0030  2004 2724 #X4      "DATA:?
      2005 4724
      2006 7660
0031  2007 1245          RUN ?"
      2010 5041
      2011 1460
0032  2012 1020          LDA;
0033  2013 5477              "Y"        [77 FILL
0034  2014 1120          ADA;
0035  2015 7773              BEG1-X6
0036                 $R3-1000
0037  1000 2003 #F2      ADD 3
0040  1001 7002          JMP F2+2       [NOTE FIRST VALUE FOR F2 IS USED
0041                 F2=40              [F2 BECOMES DOUBLY-DEFINED HERE
0042  1002 2000          ADD F3         [F3 IS UNDEFINED; ZERO VALUE USED
0043                 $3777
0044  3777 3333          ;+;-X6+*-LAM+RDC   [MEANINGLESS, BUT PERMISSABLE
0045  4000 1300          LDH
0046  4001 6003              4/R3+3
0047  4002 4206          STC FILE
0050                 TESTER=VALUE       [UNDEFINED ERROR
0051  4003 0000 #TABLE1 0,1,1,0,1
      4004 0001
      4005 0001
      4006 0000
      4007 0001
0052  4010 2425          'ABC'          [SYNTAX ERROR  [END SAMPLE MS
```

Figure 8. Listing Printed by →LI

```
    8L OF TEST MS8   PAGE 01
LN=0001

0001                                        [SAMPLE WISAL-8 PROGRAM; GENERA
                                             TES  PDP-8 OBJECT CODE
0002                        $200:SAM+5      [SAM UNDEFINED - 0 VALUE USED
0003 *0200 7300             #BEG1   CLA CLL
0004 *0201 1177 7300                TAD (VARIBL   [LITERAL
0005 *0202 1206 0206                TAD *+4
0006 *0203 5173 0173 #X6            JMP X6-10
0007 *0204 5200 0200                JMP BEG1
0010                        VARIBL=CLA CLL  [EQUALITY STATEMENT
0011                        #FILE           [NULL STATEMENT (FILE=DATA)
0012 *0205 4000             #DATA   -4000   [OCTAL NUMBER
0013 *0206 0300                     192.    [DECIMAL NUMBER
0014                        $300
0015  0300 1176 0000                TAD (''     [EMPTY CHARACTER CONST. (ZERO L
                                                 ITERAL VALUE)
0016  0301 3206 0206                DCA FILE+1
0017  0302 5200 0200                JMP BEG1
0020  0303 5173 0173                JMP X6-10
0021  0304 0024             #TEXT   'A'     [CHARACTER CONST., ZERO FILL
0022                        $2000
0023                                        [LABELS
0024  2000 3245             #R3     "GRAPH #?"  [EXAMPLES OF TEXT STATEMENTS
      2001 2443
      2002 3314
      2003 2260
0025  2004 2724             #X4     "DATA:?
      2005 4724
      2006 7660
0026  2007 1245                     RUN ?"
      2010 5041
      2011 1460
0027  2012 5477                     "Y"         [77 FILL
0030  2013 1175 7775                TAD (BEG1-X6
0031                        $R3-1000
0032  1000 1003 0003 #F2            TAD 3
0033  1001 5202 1002                JMP F2+2    [NOTE FIRST VALUE FOR F2 IS USE
                                                 D
0034                        F2=40               [F2 BECOMES DOUBLY-DEFINED HERE
0035  1002 1000 0000                TAD F3      [F3 IS UNDEFINED SO ZERO IS USE
                                                 D AS ITS VALUE
0036                        $3777
0037  3777 7774                    ;+;-X6+*-JMS+CLA   [MEANINGLESS, BUT PERMIS
                                                 SABLE
0040  4000 4574 2003                JMS R3+3    [NOT PAGE 0 OR CURRENT PAGE: AD
                                                 DRESS TREATED AS LITERAL
0041  4001 3573 0205                DCA FILE    [LIKEWISE: ADDRESS TREATED AS L
                                                 ITERAL
0042                        TESTER=VALUE        [UNDEFINED ERROR
0043  4002 0000             #TABLE1 0,1,1,0,1
      4003 0001
      4004 0001
      4005 0000
      4006 0001
0044  4007 2425                     'ABC'       [SYNTAX ERROR  [END SAMPLE
```

Figure 9. Listing Printed by →8L

RT

8R

25. REFERENCE TABLE →RT (*LNA*,(*LNB*,))(*NAME*,)*UNITA*$_{EOL}$

26. PDP-8 REFERENCE TABLE →8R (*LNA*,(*LNB*,))(*NAME*,)*UNITA*$_{EOL}$

These meta-commands generate a table on the Teletype of symbols (except instruction mnemonics) referenced and defined in a LINC (PDP-8) program manuscript.

a. If *NAME* and *UNITA* are not specified, the current manuscript is used. If only *UNITA* is specified, the manuscript in the manuscript working area on *UNITA* is used. If *NAME* is specified, the manuscript filed as *NAME* in the file on *UNITA* is used.

b. If no line numbers are given, the entire manuscript is used. When line numbers are specified, that portion of the manuscript from *LNA* thru *LNB* is used. When only *LNA* is specified, *LNB* is assumed to be the last line number of the manuscript. When only *LNB* is specified, *LNA* is assumed to be 1.

c. The symbols used in the manuscript are printed in alphabetical order, each preceded by its value and followed by the sequence of line numbers at which the symbol is defined. This sequence is followed by two blanks and the sequence of line numbers at which the symbol is referenced. Four blanks will be printed in place of the definition line numbers if the symbol is undefined.

d. If line numbers are specified, only those symbols occurring within the specified line numbers are printed. Similarly, the definitions and references are printed only if they also occur within the specified line numbers.

e. The *NAME*, if any, and the page number in decimal are typed in the title at the top of each page in the form:

RT OF *NAME* PAGE *nn*

f. Strike EOL at any time to return to the manuscript display.

g. Line number sequences which are too long to be printed on one line are automatically carried over to the next line.

h. Symbols appearing in text, comments, or character constants are not included in the reference table.

i. The symbol table used by these meta-commands is obtained by running the corresponding assembler (without changing the contents of the binary working area). If the symbol table fills up during this assembly, the following message is printed at the beginning of the first page:

ST FULL AT LINE *llll*

The reference table will include only those symbols defined before the line *llll*.

j. 56_8 lines of references are printed per page. A Teletype page is exactly eleven inches long if cut at the marker which appears between pages.

```
     RT OF TEST MS     PAGE 01

0200 BEG1    0003   0010 0023 0035
0206 DATA    0013
1000 F2      0037 0041   0040
0000 F3           0042
0206 FILE    0012   0020 0047
2001 R3      0027   0036 0046
4003 TABLE1  0051
0000 TESTER  0050
0303 TEXT    0021
0000 VALUE        0050
2000 VARIBL  0011   0004
2004 X4      0030
0204 X6      0007   0007 0024 0035 0044
```

Figure 10. Reference Table of the Manuscript in Figure 8

VI. Assembly Language Conventions

. INTRODUCTION

To facilitate program writing, characters and symbols are used instead of their binary (or octal) equivalents. For example, "LDA" rather than $001\ 000\ 000\ 000_2$ (or 1000_8) is written in a program, but when the program is put into memory, $001\ 000\ 000\ 000_2$ must be substituted for the "LDA". A program which performs this substitution is called an assembler.

The LAP6W system provides two assemblers: the WISAL (WISconsin Assembly Language) assembler for LINC programs, and the WISAL-8 assembler for PDP-8 programs. The AS and 8A meta-commands initiate WISAL and WISAL-8 assembly, respectively. The source code is read from LINC tape, the LINC or PDP-8 object code is written onto LINC tape, and assembly errors, memory allocation and the symbol table are displayed on the CRT. Other meta-commands allow the programmer to list a WISAL or WISAL-8 source program and print its reference table on the Teletype.

In order to communicate effectively with the WISAL and WISAL-8 assemblers, the user must be aware of certain syntactic conventions which govern the writing of programs. The remainder of this section describes in detail the various statement elements available to the WISAL and WISAL-8 user and the conventions for combining these elements into meaningful programs.

Figures 8 and 9, pages 42 and 43, are nonsense WISAL and WISAL-8 programs which contain examples of some of the features of these assemblers.

.. ELEMENTS

1. Numbers

a) An octal number may be represented by a string of octal digits (0 through 7). This representation in base 8 positional notation is assigned its corresponding octal value. Octal numbers may have any value between 0 and 7777_8. Preceding zeros are ignored.

Examples of Octal Numbers:

Legal	Illegal
46	48
152	10000
0000143	

b) A decimal number may be represented by a string of decimal digits (0 through 9) terminated by a decimal point (.). This representation in base 10 positional notation is assigned its corresponding octal value. Decimal numbers may have any value between 0 and 4095_{10}. Preceding zeros are ignored.

Examples of Decimal Numbers:

Legal	Illegal
38.	38
4006.	4106.
0000099.	

2. Symbols

A symbol is a single letter (A through Z) or a letter followed by a string of letters and digits (0 through 9). In symbols of more than six [four] charac s, all but the first six [four] characters are ignored.

To each program symbol (i.e., a symbol appearing in the user's source code) that is properly used, WISAL assigns an octal value. The symbols tabulated in Chart VI (the standard LINC instruction mnemonics) and Chart VII (the standard PDP-8 instruction mnemonics) are inherently assigned the corresponding octal values in WISAL and WISAL-8, respectively. To assign values to all other program symbols, the programmer should use the "label" and "equality" features yet to be discussed.

Examples of Single Symbols:

Legal	Illegal
A	2A
NBC	ABC9$
JMP	A3C9QR$
A3C9Q	AB#CD
A3C9QR	A B
A3C9QR4	

Note: To WISAL and WISAL-8, A3C9QR4 and A3C9QR [and A3C9Q] are the same symbol.

3. Text (")

Text is a string of characters enclosed in a pair of double quotes ("). The string may be of any length. All characters, including EOL, but excluding (") are permitted in the string.

Unlike numbers, symbols, and character constants, the value assigned to a text string may occupy more than one 12-bit word. Values are assigned as follows:

a) if the string contains 2N ($N \geq 0$) characters, N 12-bit words are assigned, each containing the internal representation of two characters.

b) if the string contains 2N-1 ($N \geq 1$) characters, N 12-bit words are assigned, each but the last containing the internal representation of two characters, and the last containing the left justified representation of the last character with 77 fill.

Examples of Text:

Legal	Value	Illegal
"AB"	2425_8	"A"B"
"A"	2477_8	""""
"3AB"	0324_8	
	2577_8	
" "	1477_8	
""	none	

4. Character Constants (')

A character constant is one or two characters enclosed in a pair of single quotes ('). All characters, including EOL, but excluding ('), are permitted in the string.

The character constant is assigned a value as follows:

a) if the (')'s enclose a single character, the value is the internal representation of that character right justified with zero fill;

b) if the (')'s enclose two characters, the value is the internal representation of those characters in the order they appear.

In the unusual cases where:

c) the (')'s enclose no characters, the value 0000_8 is assigned;

d) the (')'s enclose more than two characters, the value is the internal representation of the first two characters as in b, and, this "syntax" error is called to the user's attention in the post-assembly error display.

As will be seen, character constants may be used in places where text may not be used.

Examples of character constants:

Legal	Value	Illegal
'A'	0024_8	'ABCXYZ'
' A'	1424_8	'A''
'AB'	2425_8	'
'A '	2414_8	

5. Comments ([)

A comment is a string of characters preceded by a left bracket ([). The string may be of any length and may include any characters, including ([), but excluding EOL.

A comment is not assigned a value. Comments have no effect on the object code; they allow the programmer to describe adjacent source code without destroying it.

Examples of Comments:

Legal	Illegal
[ALPHA=27,396	A[COMMENT
[$1970	ABC
[[A]+B[	

6. Expressions

An expression is a string of symbols, numbers, and character constants separated by blanks and/or the special characters (+), (-), (/), (!), (;), and (*).

Legal Examples	Illegal Examples
27	27+"AB"
ALPHA	A+$
'AB'	
ALPHA BETA	
QN/BN+27	
WRC;!40	
35// 'Z'-'4A'	
A/-B	

The special characters (!), (;), and (*) serve not only as separators, but also have values assigned to them as follows:

(!) = 0010_8 in WISAL (the LINC "unit-bit")
= 0000_8 in WISAL-8
(;) = 0020_8 in WISAL (the LINC "i-bit")
= 0400_8 in WISAL-8 (the PDP-8 "indirect-bit")
(*) = the present execution location$_8$ in WISAL (the LINC "p") and WISAL-8.

The last character (*) will be discussed later (p. 51).

An expression is evaluated from left to right by forming a sequence of subtotals. The operators (+) and (-) determine whether the value assigned to the following symbol, number or character constant is added to or subtracted from the previous subtotal to form the new subtotal. Arithmetic is 12-bit one's complement in WISAL and 12-bit two's complement in WISAL-8.

A (/) operates on the subtotal arising from the evaluation of the expression to the left of the (/); it causes this 12-bit subtotal to be rotated circularly three bits to the right (e.g., 2001_8 rotated circularly three bits to the right yields 1200_8).

The following rules pertain to the evaluation of peculiar cases:

a) Blanks adjacent to special characters are ignored.

b) If blanks are used as a separator, a (+) is inferred in WISAL, but a 12-bit inclusive 'or' (designated by "∨" below) is performed in WISAL-8.

c) In a string of (+)'s, (-)'s and blanks, all but the rightmost (+) or (-) are ignored.

d) If a string of (+)'s, (-)'s and blanks precede a (/), they are all ignored.

e) For completeness, the special characters (?), (<), (>), (]), ([), and (\) have the value 0000_8 and may be used as separators.

Examples of Expressions (parentheses are used to indicate the order of operations

Expression	WISAL Equivalent	WISAL-8 Equivalent
-0	7777	0000
-1	7776	7777
JMP;	6000+20	5000+400
7/1	7001	7001
'AB'//	'BA'	'BA'
A- B	A-B	A-B
A-B C	(A-B)+C	(A-B)∨C
STC*C	(STC+*)+C	(STC∨*)∨C
A--B	A-B	A-B
A-+B	A+B	A+B

C. STATEMENTS

An element, or combination of elements, followed by a "terminator" may form a statement. A statement, or string of statements, in turn forms a WISAL or WISAL-8 source code manuscript.

There are seven kinds of statements:

Equality Statements	- for assigning values to symbols.
Origin Statements	- for specifying execution and storage positions, and for assigning values to (*) and symbols appearing in labels.
Expression Statements	- for putting 12-bit binary numbers into storage locations.
Memory Reference Statements	- for putting PDP-8 memory reference instructions into storage locations.
Text Statements	- for putting internal character codes into storage locations.
Null Statements	- for isolating comments and labels from other statements.
Control Statements	- to control printing when listing WISAL source code using the LI meta-command, or to control literal assignment in WISAL-8. (Control statements are not to be confused with meta-commands.)

As in expressions, blanks adjacent to special characters in statements are ignored.

A statement must always be terminated by a terminator:

i. a (,), except for control statements (allows the placement of several statements on one manuscript line);

ii. an EOL; or

iii. a comment followed by an EOL.

Recall that the purpose of an assembler is to generate binary object code that may be conveniently loaded into core memory (e.g., by the LAP6W LB meta-command). The string of 12-bit numbers generated by WISAL or WISAL-8 goes into the binary working area (LINC tape blocks 450_8 through 467_8), and not directly into core storage locations. The user may control where in this 20_8-block area, and thus ultimately where in core memory, object code is to be placed by inserting origin statements (see page 51) at appropriate points in the source code manuscript. The relationship between the "storage location counter", the location in the binary working area, and the corresponding location in core storage is as follows:

Storage Location	Tape Blocks	LB (Classic LINC, μ-LINC 100)	LB (μ-LINC 300) 8B (LINC-8, PDP-12)	LB (PDP-12) (LINC-8)
0000-1777	450-453	lower bank	bank 0	bank 1
2000-3777	454-457	upper bank	bank 1	bank 2
4000-5777	460-463		bank 2	bank 3
6000-7777	464-467		bank 3	(bank 4)

Thus, the whole assembly procedure amounts to generating the right numbers and putting them in the right spots. Roughly speaking, expression, text, and memory reference statements, with the help of equalities and labels (to be discussed below), generate the right numbers, and origin statements take care of positioning.

Before examining the different statements, let us review the assignment of values to, or synonomously the "definition" of, a program symbol.

i. After WISAL or WISAL-8 assigns a value to a symbol, the symbol is said to be defined.

ii. Instruction mnemonics are inherently defined.

iii. A program symbol not inherently defined is "undefined" until it is defined in an equality statement or by a label.

iv. If labeling or an equality statement is used to assign a value to a previously defined symbol, that symbol is said to be "doubly defined". A doubly defined symbol has only one value, the first value assigned to it.

Misused symbols are "marked" undefined or doubly defined, as appropriate. Unrecognizable statements are marked as "syntax errors". WISAL uses these marks to form the post-assembly error display discussed on page 37.

1. Equality Statements: *symbol = expression*

An equality statement is a symbol followed by an (=) followed by an expression and a terminator.

Equality statements serve to define a symbol to the assembler; they do not of themselves generate a location in the binary. If the symbol is previously ur. defined, it will be assigned the value of the expression; if it is defined (or doubly defined), it will be marked doubly defined and no other action taken. All symbols in the expression must have been previously defined.

Examples of Equality Statements:

Legal	Illegal
ALPHA=10.	A+B=C
A = B/'Q'	10 = ALPHA
C = ALPHA	=6
STOPCH='.'	A=B=C
	STOPCH="."

Sample Program Segment
(Assign the value of X, 0010_8, to A):

Legal	Illegal
X=10	A=X or X=10
A=X	X=10 A=0
	A=X

2. Origin Statements: *$expression* or *$expression:expression*

The single origin is a ($) followed by an expression and a terminator. The double origin is a ($) followed by an expression followed by a (:) followed by another expression and a terminator.

WISAL and WISAL-8 maintain two counters, the execution location counter and the storage location counter. This use of two counters facilitates the preparation of program overlays by allowing several overlays to be assembled together, so that common locations may be referenced. These counters, initially 20_8 in WISAL and 200_8 in WISAL-8, are incremented by 1 after each expression or memory reference statement and by N after a text statement with 2N or 2N-1 characters in the text string (the arithmetic is 12-bit two's complement). The present value of the execution location counter is assigned to (*) which may

appear in any expression. The present value of the storage location counter may not be accessed like the execution location counter, but serves only to determine the location in the binary working area where the next 12-bit binary word generated by an expression or text statement will be stored. Note that in WISAL the execution location counter should be less than 2000_8 for executable code and less than 4000_8 for data. In WISAL-8, the execution location counter may be set to any value from 0000 to 7777_8.

The origin statement allows the programmer to reset these two counters. If only one expression appears (e.g., $A+B), then both counters are set to the value of that expression. If both expressions appear (e.g., $400+A:2000+A), the execution location counter is set to the value of the first expression and then the storage location counter is set to the value of the second expression. All symbols appearing in origin statements must have been previously defined.

Note: In WISAL-8, if literals (see page 54) are being assigned locations in the current page, they are written out on tape whenever the storage location counter is changed to a new page, either by an origin statement or the normal sequential assignment of locations. Thus, it is unwise to leave a page and then return to it, because the literals assigned earlier will be lost. Furthermore, since literals are stored on the current storage page, the storage and execution locations should differ only by multiples of 200_8 (the page size). This restriction does not apply to LINC (i.e., WISAL) programs.

A legal segment of source code:

Origin	Execution Location Counter	Storage Location Counter
$100	100	100
$*+50	150	150
$*:400	150	400
$*:*+100	150	250
A=*	150	250
$*+10:A	160	150
$*+10:*	170	170

An illegal segment of source code (for A not previously defined):

$30	30	30
$*+10:A+30	40	30 (A undefined)

3. Labels: *#symbol*

A label is a (#) followed by a symbol. Any type of statement may be labeled, but no statement need be labeled. A label must precede the statement and be separated from it by another label, labels, a blank, or blanks.

The symbol, if not previously defined, is assigned the current value in the execution location counter. A label thus serves to define a particular location for other program references. If the symbol was previously defined it will be marked as doubly defined and its previous value will be unchanged. A label by itself on a manuscript line does not generate a word of binary.

Examples of Labels:

```
#K4 4/0
#SAVZREG  ZTA  [SAVE THE Z REGISTER SUBROUTINE
          ZZZ
          ADD K4
          ROL 1
          STC ZREG
          JMP 0  [RETURN TO CALLING ROUTINE
#ZREG     ?
```

4. Expression Statements: *expression*

An expression statement is an expression followed by a terminator; in WISAL-8, the first element appearing in the expression must not be a PDP-8 memory reference instruction (i.e., DCA, TAD, AND, ISZ, JMP, JMS) lest the statement be interpreted as a memory reference statement.

The value of the expression is stored in the memory location specified by the storage location counter. Symbols appearing in the expression may be defined anywhere in the program, but must be defined somewhere.

Legal WISAL Program:	Storage Location	Value
XR=3		
$XR,4/3777 [INITIALIZE XR	0003	7777
$20		
#KBDIN #WAIT KST, JMP WAIT, KBD [GET CHARACTER	0020	0415
	0021	6020
	0022	0515
AZE; [TEST FOR 0	0023	0470
JMP *+3	0024	6027
STH;XR	0025	1363
JMP KBDIN	0026	6020
#DONE HLT	0027	0000

5. Memory Reference Statements: *mri ; address expression*

Memory reference statements exist in WISAL-8, but not in WISAL. A memory reference statement is a PDP-8 memory reference instruction (*mri:* i.e., DCA, TAD, AND, ISZ, JMP, JMS) followed by the optional (;) denoting indirect addressing, followed by the "address expression" and the terminator. As usual, blanks and/or special characters are used to separate elements.

The address expression may be an expression or a "literal". If it is an expression, the value of the address expression is related to the value of the expression as follows:

Value of Expression (e)	Value of Address Expression (a)
$e<200_8$ (page zero address)	a=e
$e \geq 200_8$ and in current execution page	$a=200_8 + e \bmod 200_8$ (the lower seven bits of e with the current-page bit set.)

Value of Expression (e)	Value of Address Expression (a)
$e \geq 200_8$ and not in current execution page	If the indirect bit is set, a=0 and a syntax error is marked; otherwise, the indirect bit is set and the address expression is treated as a "literal expression".

A literal (in WISAL-8 only) is an expression or memory reference statement preceded by a left parenthesis ((). In normal operation, WISAL-8 assigns a page zero address (an octal number) to the literal, and the value of the "literal expression" (the expression or memory reference statement following the left parenthesis) is stored in that address. The addresses are assigned backwards starting at the bottom (location 177) of page zero. The value of the memory reference instruction, the (;), if present, and the value of the address expression are combined by a 12-bit inclusive 'or' and stored in the memory location specified by the current storage location counter.

To conserve memory, all equal literal expressions stored in a page are stored in the same location. Different literal expression values stored in the same page are of course stored in different locations.

Control statements (see below) can be used to switch literal assignment between page zero and the current page.

Legal WISAL-8 Program (literals assigned to page zero)

```
[ZEROS INTO PAGES 2 THRU 37
$200, INDEX=377
      CLA
#ZERO TAD (INDEX  [TAD 177
      TAD (ONE    [TAD 176
      SNA, HLT    [HLT IF ACC=0
      DCA (377    [DCA 177
      DCA;(INDEX  [DCA;177
      JMP;*+1     [NEXT
      ZERO
      ONE=1
```

6. Control Statements: *%control character*

A control statement is a (%) followed by a control character and a terminator other than (,). Characters between the control character and the EOL are ignored. As usual, blanks between the (%) and the control character (i.e., blanks adjacent to a special character) are ignored.

a. WISAL control statements only affect the listing of manuscripts (the "→LI" meta-command); they are completely ignored during assembly; they generate no object code. Control characters have the following functions:

Character	Action (WISAL only)
N	Listing will be suppressed beginning with the next line*.
L	Listing will be resumed at the next line.
T	The next line will be listed at the top of a page.

*A manuscript line is a string of characters not including EOL, but preceded by EOL and followed by EOL.

Legal WISAL Program Segment

```
        % TOP OF FORM
        [ MAIN PROGRAM
#B      SET;X
        1777
        JMP A
        %NO LISTING
        ADD X+2
        % LIST
#A      JMP B
```

b. WISAL-8 control statements affect the assignment of literals:

Character	Action (WISAL-8 only)
Z	Assign page zero addresses to subsequent literals until (%C) is encountered. This is the initial mode of assembly.
C	Assign current page addresses to subsequent literals until (%Z) is encountered.

Note: Whenever a WISAL-8 control statement is executed, previously assigned literals are forced out onto tape and forgotten. Thus, it is unwise to switch from page zero to current page literals, and back again, because the earlier page zero literals will be overwritten with any new ones.

7. Text Statements: *text*

A text statement is text followed by a terminator.
The 12-bit words of internal character code generated by the text are stored consecutively starting at the location specified in the storage location counter.

Legal WISAL Program Segment		Storage Location	Value
$400			
	QANDA=1000		
	JMP QANDA	0400	7000
	Q1	0401	0403
	HLT	0402	0000
#Q1	"HELLO"	0403	3330
		0404	3737
		0405	4277

8. Null Statements: (*blanks*)

A null statement is a possibly empty string of blanks followed by a terminator.

A null statement generates no object code and has no effect on the execution location counter or the storage location counter. It is primarily used to place a label or a comment on a line by itself.

Legal Program Segment		Storage Location	Value	
$400	[SUBROUTINE ENTRY			Note that the previously undefined symbols ENTRY and WAIT are assigned the value 400_8.
# ENTRY	[LABEL FOR ENTRY			
#WAIT				
	KST	0400	0415	

Appendix: Notes

Efficient Use of LAP6W

Although editing may be done in any order, it is occasionally more efficient (less tape shuffling) if low numbered lines are edited before high numbered lines. Reading the manuscript (i.e., not changing it) may be done in any order without affecting tape efficiency.

The abilities to SAVE part of a manuscript and to ADD one manuscript in the middle of another manuscript can be used to reconfigure large manuscripts and to substitute for a REMOVE command.

Adding or assembling a long manuscript from the file is much faster if the manuscript is not on the LAP6W system tape. Move it with CM to another tape before executing an AM, AS or 8A command.

Assembly is faster if origin statements which refer to storage addresses in one memory quarter are not interspersed with origin statements which refer to storage addresses in different quarters. This technique will save some tape shuffling during assembly.

Exiting

If either EX, SU, LB, 8B, or *+MP* is executed when 1 is the current line number, upon re-entry the manuscript display will suggest that there is no current manuscript, which may or may not be the case. Therefore, if LAP6W, when restarted, displays only the line number 1, do not assume that there is no current manuscript without trying to locate forward.

Re-entering LAP6W Under Program Control

After executing EX, SU, LB or *+MP*, LAP6W can be re-entered under user program control (assuming PROGOFOP or TRAP is still resident if on the LINC-8 or PDP-12). The program should put

```
RCG  701
3/400  [7/400 in classic LINC]
```

in registers 15 and 16 respectively, then execute a JMP 15. LAP6W and the current manuscript will be as they were before the meta-command was executed. The program causing the re-entry must insure that the lower and upper memory banks are different. Do not change the instruction, and do not move it to some other register.

On the PDP-12, LAP6W may also be re-entered by executing the following instructions:

```
LMB 0
JMP 400
```

Working Area Length

LAP6W checks for various boundary crossings. It will not, for example, permit a manuscript to go beyond the working area as it is entered.

If the manuscript working area boundary is encountered, LAP6W ignores additional manuscript input. It will, however, continue to honor all the meta-commands. Thus, one can SAVE the MANUSCRIPT, switch to a configuration of LAP6W which has a larger working area, and continue. To continue editing, save some part of the manuscript and then delete this part from the current manuscript. After deleting this part, it is necessary to perform a meta-command such as EX so that the system can realize that there is now room for more manuscript input.

LAP6W Manuscript Structure

Coding Rules:

A manuscript is a single string of 6-bit character codes. On a LINC tape the codes are stored in sequential half-words in sequential, contiguous blocks in the order in which they appear on the scope.

1. The first word of the first block contains the number 2065. The second word contains 5712. The last half-word of the manuscript contains 77. There is no other control information associated with a manuscript.

2. The characters are coded as in Chart II. The codes for DELETE (13) and EOMS (77) (except last) do not appear in a LAP6W manuscript. One EOL (12) does not appear next to another EOL.

Generating Manuscript

A manuscript may be generated other than with LAP6W and still be used with LAP6W. However, it must:

1. Conform to the above coding rules.
2. Be put in contiguous blocks on a LINC tape.
3. Be added to the LAP6W manuscript working area with ADD MANUSCRIPT,

$$\rightarrow \text{AM } (BN,)UNITA_{\text{EOL}}$$

If the above procedure is followed, the generated manuscript may then be treated as any other manuscript.

Manipulating Manuscript:

LAP6W may be used to generate a manuscript (a program segment, a bibliography, etc.) which is to be the data source for some other program. It may be assumed that the current manuscript on the LAP6W tape (blocks 470 ff.) conforms to the above coding rules and is therefore accessible following the commands EX, AS, 8A, LB, 8B, SM, SP, or +*MP*. One may read the manuscript directly from the working area.

If a program changes the contents of the manuscript working area on a LAP6W tape, it is advisable that this program cause LAP6W to think that there is no current

manuscript. This is because the pointers and information in LAP6W pertaining to the current manuscript may no longer be valid. This can be accomplished with the following code:

```
RDC unit
  Q/SYS
LDA
  base address of Q + 21
STA
  base address of Q + 20
WRC unit
  Q/SYS
SYS = 400 [first block of LAP6W
```

The program should then return to LAP6W in the usual manner. There will be no current manuscript and the current line number will be 1. The manuscript may be recovered by →AM$_{EOL}$.

Copying LAP6W Tapes

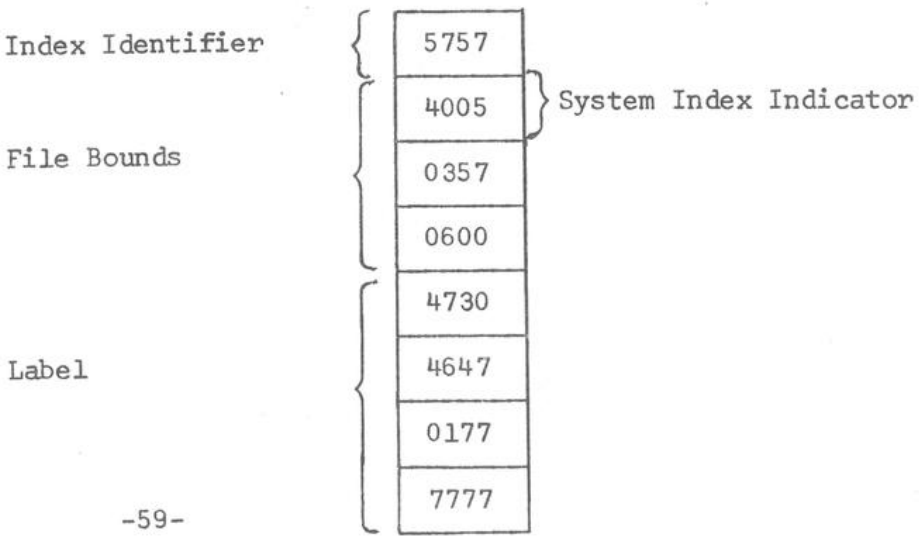
To copy an entire tape, use CO. To copy a LAP6W file use CF. To copy individual file entries use CM, CB or CP. To copy the current manuscript use AM (see "Manipulating Manuscript"). To generate a new LAP6W system use the program GASTMB [GASTCL].

Index Structure

Before attempting to make up a file Index as a manuscript, or to write programs which scan an Index for specific entries, the Index structure must be well understood.

A LAP6W Index is always two blocks long and is always in tape blocks whose block numbers end in 6 and 7 (regardless of configuration). The two blocks are divided into 100_8 segments of 10_8 words each. The first segment is the Index identifier; the other 77_8 segments are for Index entries.

Identifier Segment: If the two blocks are an Index, the first word of this segment contains 5757_8. The next three words contain the file bounds: first block of lower file (with bit 0 set if a system index), last block +1 of lower file, and first block of upper file (the upper file always ends at block 777_8). The last four words of this segment contain the Index label. Thus, for a LINC-8 system Index labeled TEST1, the first segment would appear as follows:



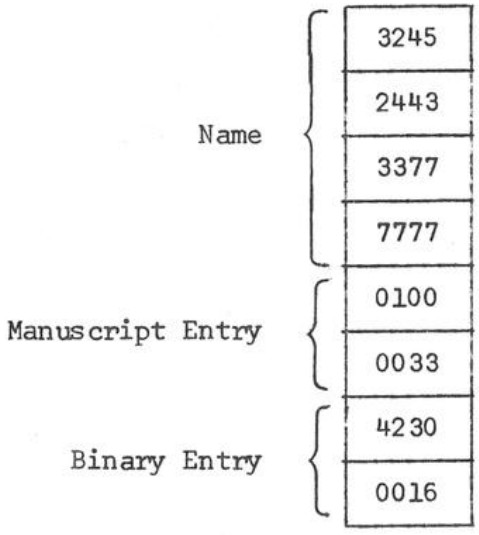
Entry Segments: An entry segment always contains the entry name in the first four words. If the segment contains a name, it must describe either a manuscript in the fifth and sixth words, or a binary program in the seventh and eighth words, or both. A name does not appear without at least one of these entries. A manuscript and a binary program of the same name must be described in the same entry segment.

Name: The entry name is stored in eight half-words beginning in the left half of the first word of the segment. Unused half-words in the name contain 77. The name conforms to the rules for entry names described in Section IV.A.3 and does not begin with a space or percent sign (%). The characters are coded as in Chart II.

Manuscript Entry: The first block number of the manuscript's location in the file is stored in the fifth word. The length of the manuscript (number of blocks) is in the sixth word. If there is no manuscript entry, these two words contain 5757.

Binary Entry: The first block number of the binary program's location in the file is stored in the seventh word, with the number of the first quarter into which it is to be loaded in the left three bits. The length of the binary program (number of blocks) is in the eighth word. If there is no binary entry, these two words contain 5757.

Example:



Coded Entry Segment

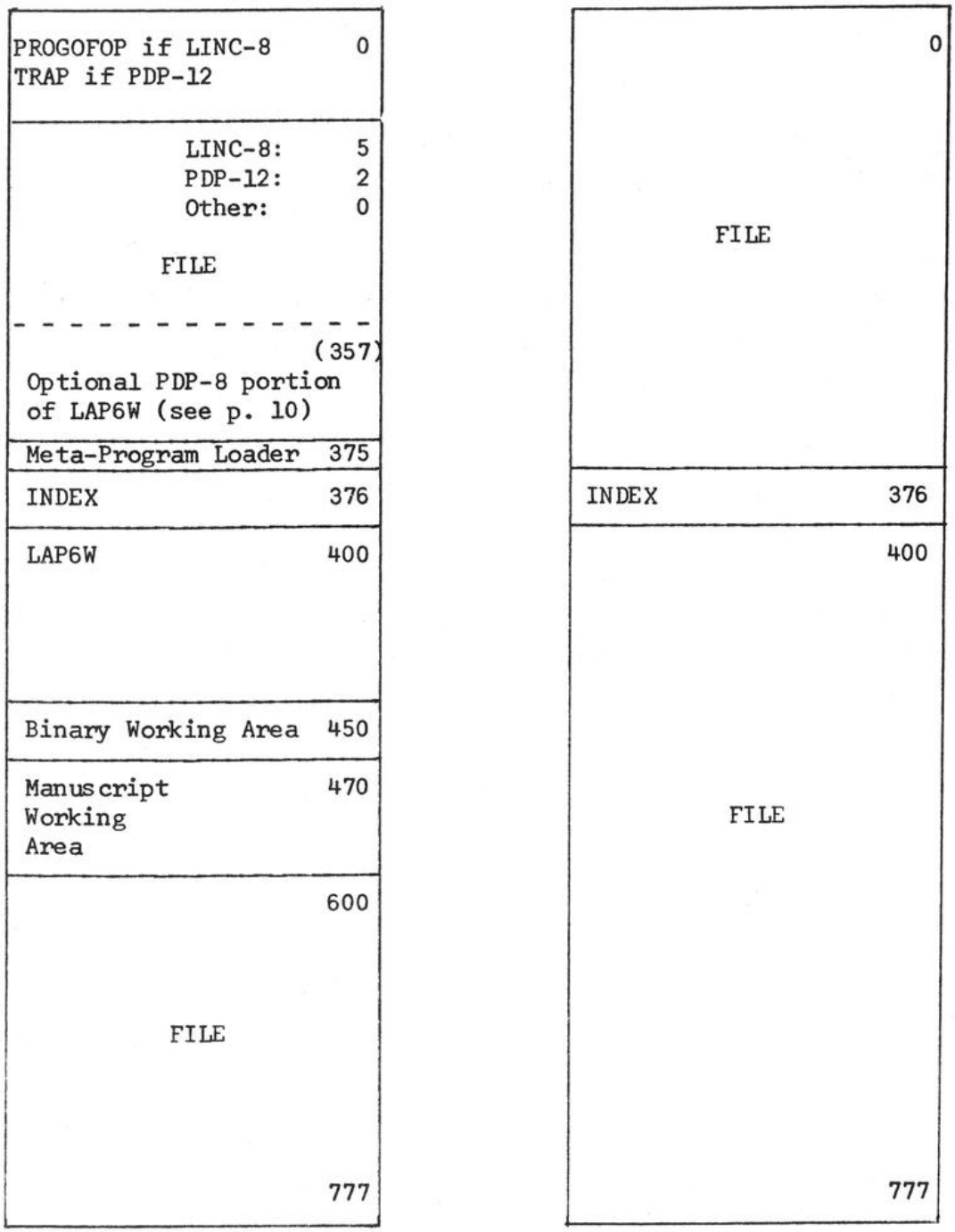
```
TAPE123 S TEST1

  NAME     BN   #B

GRAPH M   100   33
      B  4230   16
```

Corresponding Index Display

Used segments are not necessarily "packed" in the Index. They may be interspersed with unused segments.

If a segment is not used, it will contain 5757 in the first word of the name. If it is used, and describes a manuscript, the left three bits of the fifth and sixth words will contain 0. If the segment does not describe a manuscript, these bits will contain a 5. If the segment describes a binary entry, the left three bits of the eighth word will contain 0; otherwise, 5. (The seventh word, which contains the quarter number, should not be used to test whether a binary entry is present.)

CHART I. Standard LAP6W Tape Allocation



System Tape

File-Only Tape

CHART II. LAP6W Character Set - Input

Char. Name	Internal Code	LINC KBD	μ-LINC 300 KBD	TTY KBD
0	00	0	0	0
1	01	1	1	1
2	02	2	2	2
3	03	3	3	3
4	04	4	4	4
5	05	5	5	5
6	06	6	6	6
7	07	7	7	7
8	10	8	8	8
9	11	9	9	9
EOL	12	EOL	EOL	RETURN
DELETE	13	del	del	RUBOUT,←
space	14	space	space	space
;	15	i	i	;
*	16	p	p	*
-	17	-	-	-
+	20	+	+	+
/	21	|	|	/
#	22	#	#	#
CASE	23	CASE	CASE	unavailable
↑	23	unavailable	unavailable	↑,ALT MODE
A	24	A	A	A
B	25	B	B	B
C	26	C	C	C
D	27	D	D	D
E	30	E	E	E
F	31	F	F	F
G	32	G	G	G
H	33	H	H	H
I	34	I	I	I
J	35	J	J	J
K	36	K	K	K
L	37	L	L	L
M	40	M	M	M
N	41	N	N	N
O	42	O	O	O
P	43	P	P	P
Q	44	Q	Q	Q
R	45	R	R	R
S	46	S	S	S
T	47	T	T	T
U	50	U	U	U
V	51	V	V	V
W	52	W	W	W
X	53	X	X	X
Y	54	Y	Y	Y
Z	55	Z	Z	Z
META (→)	56	META	META	LINE FEED
\	56	META	META	\(SHIFT L)
%	57	CASE del	CASE del	%
?	60	CASE SPACE	?	?
=	61	=	=	=
!	62	μ	μ	!
,	63	,	,	,
.	64	.	.	.
$	65	⊟	⊟	$
[	66	[	[	[(SHIFT K)
]	67	CASE CASE	SHIFT T	](SHIFT M)
"	70	CASE A	"	"
'	71	CASE B	SHIFT V	'
<	72	CASE C	<	<
>	73	CASE D	>	>
(	74	CASE E	SHIFT Y	(
)	75	CASE F	SHIFT Z	)
:	76	CASE G	:	:
EOMS*	77			

* End-of-Manuscript

CHART III. LAP6W Character Set - Display

Char.	Display Char.	Grid
0	0	4136 3641
1	1	2101 0177
2	2	4523 2151
3	3	4122 2651
4	4	2414 0477
5	5	5172 0651
6	6	1506 4225
7	7	4443 6050
8	8	5126 2651
9	9	5122 3651
EOL	&	5126 0625
DELETE	←	1604 0404
space	blank	0000 0000
;	;	0100 0022
*	*	1024 0024
-	-	0404 0404
+	+	0404 0437
/	/	0300 6014
#	#	3614 1436
↑	↑	7720 0020
A	A	4477 7744
B	B	5177 2651
C	C	4136 2241
D	D	4177 3641
E	E	4577 4145
F	F	4477 4044
G	G	4136 2645
H	H	1077 7710
I	I	7741 0041
J	J	4142 4076
K	K	1077 4324
L	L	0177 0301

Char.	Display Char.	Grid
M	M	3077 7730
N	N	3077 7706
O	O	4177 7741
P	P	4477 3044
Q	Q	4276 0376
R	R	4477 3146
S	S	5121 4651
T	T	4040 4077
U	U	0177 7701
V	V	0176 7402
W	W	0677 7701
X	X	1463 6314
Y	Y	0770 7007
Z	Z	4543 6151
\	\	1460 0003
%	%	6462 2313
?	?	4020 2055
=	=	1212 1212
!	!	7500 0000
,	,	0500 0006
.	.	0100 0000
$	$	2735 6735
[	[	7700 0041
]	]	4100 0077
"	"	0070 0070
'	'	7000 0000
<	<	0400 2112
>	>	1221 0004
(	(	3600 0041
)	)	4100 0036
:	:	2200 0000
EOMS		0000 0000

CHART IV. LAP6W Character Set - Printed Output

Char.	Teletype Char.	Code
0	Ø	260
1	1	261
2	2	262
3	3	263
4	4	264
5	5	265
6	6	266
7	7	267
8	8	270
9	9	271
EOL	CR,LF	215, 212
DELETE	←	337
space	space	240
;	;	273
*	*	252
-	-	255
+	+	253
/	/	257
#	#	243
↑	↑	336
A	A	301
B	B	302
C	C	303
D	D	304
E	E	305
F	F	306
G	G	307
H	H	310
I	I	311
J	J	312
K	K	313
L	L	314

Char.	Teletype Char.	Code
M	M	315
N	N	316
O	O	317
P	P	320
Q	Q	321
R	R	322
S	S	323
T	T	324
U	U	325
V	V	326
W	W	327
X	X	330
Y	Y	331
Z	Z	332
\	\	334
%	%	245
?	?	277
=	=	275
!	!	241
,	,	254
.	.	256
$	$	244
[	[	333
]	]	335
"	"	242
'	'	247
<	<	274
>	>	276
(	(	250
)	)	251
:	:	272
EOMS	@	300

CHART V. Reasons for NO Display

Command	Arguments	NO Appears on Scope
AM	*(NAME,)UNITA* or *(BN,)UNITA*	If *BN*>0777. If no manuscript is found as specified.
AS,8A LI,8L PM RT,8B	*(LNA,(LNB,))(NAME,)UNITA*	If no manuscript is found as specified.
CO	None	Never
CB CM CP	*(QBBB,N,)NAME,UNITA(,UNITB)* *(BBB,N,) NAME,UNITA(,UNITB)* *NAME,UNITA(,UNITB)*	If no entry is found as specified. If Index or file on *UNITB* is full. If *UNITB* has no Index.
CF	*UNITA(,UNITB)* or *BN,UNITA,UNITB*	If no Index on *UNITA*. If Index or file on *UNITB* is full. If *BN* is illegal.
DX	*(NAME,)UNITA*	If no Index is found on *UNITA*.
EX SU	None *UNITA*	Never
FX NX	*(NAME,)UNITA* *NAME,UNITA*	Never
LB,8B	*(NAME,)UNITA* or *(QBBB,N,)UNITA*	If no binary is found as specified. If *BBB+N*>1000_8.
SB	*NAME,UNITA(,UNITB)*	If no binary on LAP6W tape. If Index or file on *UNITA* or *UNITB* is full.
SM	*(LNA,(LNB,))NAME,UNITA(,UNITB)*	If no current manuscript. If Index or file on *UNITA* or *UNITB* is full.
SP	*(LNA,(LNB,))NAME,UNITA(,UNITB)*	If no current manuscript. If no binary on LAP6W tape. If Index or File on *UNITA* or *UNITB* is full.
SS CS	*(LNA,(LNB,))STRING,* *LNA*	Never
+MP	*((N1,(N2,))STRING,UNITA(,UNITB))*	If no binary program "*+MP*-----" is found in file. If more than one binary program "*+MP*-----" is found in file. If the binary program "*+MP*-----" was not assembled for quarter 0. Others determined by the particular meta-program.

CHART VI. LINC Instruction Set

0000	MSC	Miscellaneous
0000	HLT	Halt
0001	AXO*	Accumulator to Extended Operations Buffer
0002	PDP*	Transfer to PDP-8 Mode
0003	TAC*	Tape Accumulator to Accumulator
0004	ESF*	Accumulator to Special Function Register
0005	ZTA	Z Register (MQ Register on PDP-12) to Accumulator
0006	DJR*	Disable Jump Return
0010	ENI**	Enable Interrupt
0011	CLR	Clear the Accumulator and the Link Bit
0014	ATR	Accumulator to Relay
0015	RTA	Relay to Accumulator
0016	NOP	No Operation
0017	COM	Complement
0040	SET	Set
0100	SAM	Sample
0140	DIS	Display
0200	XSK	Index and Skip
0240	ROL	Rotate Left
0300	ROR	Rotate Right
0340	SCR	Scale Right
0400	SXL	Skip on External Level Negative
0415	KST	Key Struck
0440	SNS	Sense Switch
0446	PIN**	Skip on Interrupt from Pause
0450	AZE	Accumulator Zero
0451	APO	Accumulator Positive
0452	LZE	Link Zero
0453	IBZ	Interblock Zone
0454	OVF	Skip on Overflow
0455	ZZZ	Skip on $Z_0 = 0$
0467	USK	Unconditional Skip
0500	OPR	Operate
0514	TYP***	Type Character
0515	KBD	Read Keyboard
0516	RSW	Right Switches
0517	LSW	Left Switches
0600	LMB***	Lower Memory Bank
0640	UMB***	Upper Memory Bank
0700	RDC	Read and Check
0701	RCG	Read and Check Group
0702	RDE	Read Tape

* Available for assembly in Multi-Bank versions of WISAL, but executable only on PDP-12.

** Available for assembly only in 2-Bank versions of WISAL.

*** Available for assembly only in Multi-Bank versions of WISAL.

0703	MTB	Move Toward Block
0704	WRC	Write and Check
0705	WCG	Write and Check Group
0706	WRI	Write Tape
0707	CHK	Check Tape
1000	LDA	Load Accumulator
1040	STA	Store Accumulator
1100	ADA	Add to Accumulator
1140	ADM	Add to Memory
1200	LAM	Link Add to Memory
1240	MUL	Multiply
1300	LDH	Load Half
1340	STH	Store Half
1400	SHD	Skip if Half Differs
1440	SAE	Skip if Accumulator Equals
1500	SRO	Skip and Rotate
1540	BCL	Bit Clear
1600	BSE	Bit Set
1640	BCO	Bit Complement
1740	DSC	Display Character
2000	ADD	Add
4000	STC	Store and Clear
6000	JMP	Jump

Note the use of the following PDP-12 special register instructions:

0021	AXO;	Extended Operations Buffer to Accumulator
0023	TAC;	Accumulator to Tape Accumulator
0024	ESF;	Special Function Register to Accumulator

CHART VII. PDP-8 Instruction Set

0000	AND	Logical And
1000	TAD	Two's Complement Add
2000	ISZ	Increment and Skip on Zero
3000	DCA	Deposit and Clear Accumulator
4000	JMS	Jump to Subroutine
5000	JMP	Jump
6000	IOT	I/O Transfer
6001	ION	Interrupt On
6002	IOF	Interrupt Off
6031	KSF	Skip on Keyboard Flag
6032	KCC	Clear Accumulator and Keyboard Flag
6034	KRS	"Or" Keyboard to Accumulator
6036	KRB	Read Keyboard and Clear Flag
6041	TSF	Skip on Teleprinter Flag
6042	TCF	Clear Teleprinter Flag
6044	TPC	Type Character
6046	TLS	Clear Teleprinter Flag and Type Character
6141	LINC*	PDP-12 Change to LINC Mode
6141	ICON**	LINC Control
6143	IBAC**	Read LINC B Register
6145	ILES**	Read Left Switches
6147	INTS**	Read LINC Interrupt Status
6151	ICS1**	Read Control Switches I
6153	ICS2**	Read Control Switches II
6155	IMBS**	Read LINC Memory Banks
6157	ITAC**	Read LINC Tape Window
6161	IACB**	Set LINC B Register
6163	IACS**	Set LINC S Register
6165	ISSP**	Set LINC P Register
6167	IACA**	Set LINC Accumulator
6171	IAAC**	Read LINC Accumulator
6173	IZSA**	Transfer LINC Z to LINC Accumulator
6175	IACF**	Set LINC Flip-Flops
6201	CDF	Change Data Field
6202	CIF	Change Instruction Field
6214	RDF	Read Data Field
6224	RIF	Read Instruction Field
6234	RIB	Read Interrupt Buffer
6244	RMF	Restore Memory Fields
7000	OPR	Operate
7000	NOP	No Operation
7001	IAC	Increment Accumulator
7004	RAL	Rotate Accumulator and Link Left

*Available for assembly on LINC-8 and PDP-12 versions of WISAL-8, but executable only on the PDP-12.

**Available for assembly on LINC-8 and PDP-12 versions of WISAL-8 but executable only on the LINC-8.

CHART VII (Continued)
PDP-8 Instruction Set

7006	RTL	Rotate Accumulator and Link Two Left
7010	RAR	Rotate Accumulator and Link Right
7012	RTR	Rotate Accumulator and Link Two Right
7020	CML	Complement Link Bit
7040	CMA	Complement Accumulator
7041	CIA	Complement and Increment Accumulator
7100	CLL	Clear Link Bit
7120	STL	Set Link Bit
7200	CLA	Clear Accumulator
7204	GLK	Get Link Bit
7240	STA	Set Accumulator
7402	HLT	Halt
7404	OSR	"Or" Right Switches to Accumulator
7410	SKP	Unconditional Skip
7420	SNL	Skip on non-zero Link Bit
7430	SZL	Skip on Zero Link Bit
7440	SZA	Skip on Zero Accumulator
7450	SNA	Skip on Non-zero Accumulator
7500	SMA	Skip on Minus Accumulator
7510	SPA	Skip on Plus Accumulator
7604	LAS	Read Right Switches

References

1. GASTMB/GASTCL, LCF Program UP-039-03

2. PROGOFOP, LCF Program UP-072-02

3. TRAP, LCF Program UP-086-02

Bibliography

Clark, W.A., and C.E. Molnar, "A Description of the LINC," Computers in Biomedical Research, Vol. 2, 1965, Academic Press, Inc., New York, pp. 35-66.

Convocation on the Mississippi, Proc. of the Final LINC Evaluation Program Meeting, March 18-19, 1965, Washington University, St. Louis, Missouri.

Marlowe, L., LINC Command System, Nov. 3, 1965, Brown University (informal report).

McDonald, M.D., S.R. Davisson, and J.R. Cox, Jr., A LINC Utility System, Technical Report No. 1, March 19, 1965, Biomedical Computer Laboratory, Washington University, St. Louis, Missouri.

Moore, R.K., An Operating System for the LINC Computer, Technical Report No. IRL-1038, Nov. 1, 1965, Stanford University School of Medicine, Department of Genetics, Palo Alto, California.

Wilkes, M.A., LAP3 User Manual, Aug. 1963, Center Development Office, decd., Massachusetts Institute of Technology (informal report).

Wilkes, M.A., "LAP5: LINC Assembly Program," Proc. of the DECUS Spring Symposium, May 1966, Boston, Mass., pp. 43-50.

Wilkes, M.A., LAP6 Handbook, Technical Report No. 2, May 1, 1967, Computer Research Laboratory, Washington University, St. Louis, Missouri.

Wilkes, M.A., LAP6 Use of the Stucki-Ornstein Text Editing Algorithm, Technical Report No. 18, February, 1970, Computer Systems Laboratory, Washington University, St. Louis, Missouri.

Wilkes, M.A., An Algorithm for Fast Tape File Copying, LINC Document No. 76, February, 1970, Computer Systems Laboratory, Washington University, St. Louis, Missouri.

Wilkes, M.A., "Conversational Access to a 2048-Word Machine," Communications of the ACM, Vol. 13, pp. 407-414, July, 1970.

Wilkes, M.A., "Scroll Editing: An On-Line Algorithm for Manipulating Long Character Strings," IEEE Transactions on Computers, Vol. C-19, pp. 1009-1015, November, 1970.

Wilkes, M.A., and W.A. Clark, "Programming the LINC," LINC Vol. 16, Programming and Use-I, Section 2, June, 1965, Computer Research Laboratory, Washington University, St. Louis, Missouri.

SYSTEM SUMMARY SHEET
LINC-8, PDP-12

Manuscript Displays

Key	Meaning
CTRL Q	FWD page
CTRL W	FWD line
CTRL E	FWD character
CTRL A	BWD page
CTRL S	BWD line
CTRL D	BWD character
CTRL X	Split line
RETURN	End line
CTRL SHFT K	Delete MS
CTRL P	Delete page
CTRL L	Delete line
RUBOUT	Delete character

Assembler Displays

Key	Meaning
CTRL Q	Next display
CTRL W	FWD frame
CTRL A	To first display
CTRL S	Begin display
RETURN	Return
P	Print display
L	List MS
R	Reference Table
Other	Ignore

Display Index

Key	Meaning
CTRL Q	FWD frame
CTRL W	FWD entry
CTRL A	BWD frame
CTRL S	BWD entry
RUBOUT	Delete entry
RETURN	Return
R	Reread Index
#	Write Index
P	Print Index
Other	Ignore

Meta-Commands

→*LN*	Locate at line, page 5
→AM (*BN*,)*UNITA*	Add manuscript by block number, 13
→AM (*NAME*,)*UNITA*	Add manuscript by name, 13
→AS (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	Assemble, 36
→CB (*QBBB*,*N*,)*NAME*,*UNITA*(,*UNITB*)	Copy binary, 27
→CF *BN*,*UNITA*,*UNITB*	Copy file, 26
→CF *UNITA*(,*UNITB*)	Copy file, 26
→CM (*BBB*,*N*,)*NAME*,*UNITA*(,*UNITB*)	Copy manuscript, 27
→CO	Copy, 14
→CP *NAME*,*UNITA*(,*UNITB*)	Copy program, 28
→CS *LNA*	Continue search, 23
→DX (*NAME*,)*UNITA*	Display Index, 29
→EX	Exit, 15
→FX (*NAME*,)*UNITA*	File-only Index, 31
→LB (*NAME*,)*UNITA*	Load binary by name, 16
→LB (*QBBB*,*N*,)*UNITA*	Load binary by block number, 16
→LI (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	List, 40
→NX *NAME*,*UNITA*	Name Index, 31
→PM (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	Print manuscript, 20
→RT (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	Reference Table, 44
→SB *NAME*,*UNITA*(,*UNITB*)	Save binary, 33
→SM (*LNA*,(*LNB*,))*NAME*,*UNITA*(,*UNITB*)	Save manuscript, 34
→SP (*LNA*,(*LNB*,))*NAME*,*UNITA*(,*UNITB*)	Save program, 35
→SS (*LNA*,(*LNB*,))*STRING*,	String search, 22
→SU *UNITA*	Switch unit, 15
→8A (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	PDP-8 Assemble, 36
→8B (*NAME*,)*UNITA*	PDP-8 load binary by name, 18
→8B (*QBBB*,*N*,)*UNITA*	PDP-8 load binary by block number, 18
→8L (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	PDP-8 list, 40
→8R (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	PDP-8 Reference Table, 44
→+*MP*((*N1*,(*N2*,))*STRING*,*UNITA*(,*UNITB*))	Load meta-program, 24

Other Displays

	Response Key
NO	RETURN = Return to Manuscript Display
NO INDEX ON UNIT *u*	RETURN = Return to Manuscript Display
REPLACE label	# = Yes; RETURN = No
REPLACE *xx NAME*,*u* <#>	# = Yes; RETURN = No
COpy - PDP-8 Load Binary	ALT MODE = Return to Manuscript Display RUBOUT = Delete last answer RETURN = On to next question

SYSTEM SUMMARY SHEET
Classic LINC, μ-LINC 100, μ-LINC 300

Manuscript Displays

Key	Meaning
CASE 0	FWD page
CASE 1	FWD line
CASE 2	FWD char.
CASE Q	BWD page
CASE W	BWD line
CASE R	BWD char.
CASE E	BWD char. (μ-LINC 300)
CASE S	Split line
EOL	End line
CASE K	Delete Manuscript
CASE P	Delete page
CASE L	Delete line
DEL	Delete char.

Assembler Displays

Key	Meaning
CASE 0	Next display
CASE 1	FWD frame
CASE Q	To first display
CASE W	Begin display
EOL	Return
P	Print display
L	List MS
R	Reference Table
Other	Ignore

Display Index

Key	Meaning
CASE 0	FWD frame
CASE 1	FWD entry
CASE Q	BWD frame
CASE W	BWD entry
DEL	Delete entry
EOL	Return
R	Reread Index
#	Write Index
P	Print Index
Other	Ignore

Meta-Commands

→*LN*	Locate at line, page 5
→AM (*BN*,)*UNITA*	Add manuscript by block number, 13
→AM (*NAME*,)*UNITA*	Add manuscript by name, 13
→AS (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	Assemble, 36
→CB (*QBBB*,*N*,)*NAME*,*UNITA*(,*UNITB*)	Copy binary, 27
→CF *UNITA*(,*UNITB*)	Copy file, 26
→CF *BN*,*UNITA*,*UNITB*	Copy file, 26
→CM (*BBB*,*N*,)*NAME*,*UNITA*(,*UNITB*)	Copy manuscript, 27
→CO	Copy, 14
→CP *NAME*,*UNITA*(,*UNITB*)	Copy program, 28
→CS *LNA*	Continue search, 23
→DX (*NAME*,)*UNITA*	Display Index, 29
→EX	Exit, 15
→FX (*NAME*,)*UNITA*	File-only Index, 31
→LB (*NAME*,)*UNITA*	Load binary by name, 16
→LB (*QBBB*,*N*,)*UNITA*	Load binary by block number, 16
→LI (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	List, 40
→NX *NAME*,*UNITA*	Name Index, 31
→PM (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	Print manuscript, 20
→RT (*LNA*,(*LNB*,))(*NAME*,)*UNITA*	Reference Table, 44
→SB *NAME*,*UNITA*(,*UNITB*)	Save binary, 33
→SM (*LNA*,(*LNB*,))*NAME*,*UNITA*(,*UNITB*)	Save manuscript, 34
→SP (*LNA*,(*LNB*,))*NAME*,*UNITA*(,*UNITB*)	Save program, 35
→SS (*LNA*,(*LNB*,))*STRING*,	String search, 22
→SU *UNITA*	Switch unit, 15
→+*MP* ((*N1*,(*N2*,))*STRING*,*UNITA*(,*UNITB*))	Load meta-program, 24

Other Displays

	Response Key
NO	EOL = Return to Manuscript Display
NO INDEX ON UNIT *u*	EOL = Return to Manuscript Display
REPLACE label	# = Yes, EOL = No
REPLACE *xx NAME*,*u* <#>	# = Yes, EOL = No
COpy	CASE = Return to Manuscript Display DELETE = Delete last answer EOL = On to next question