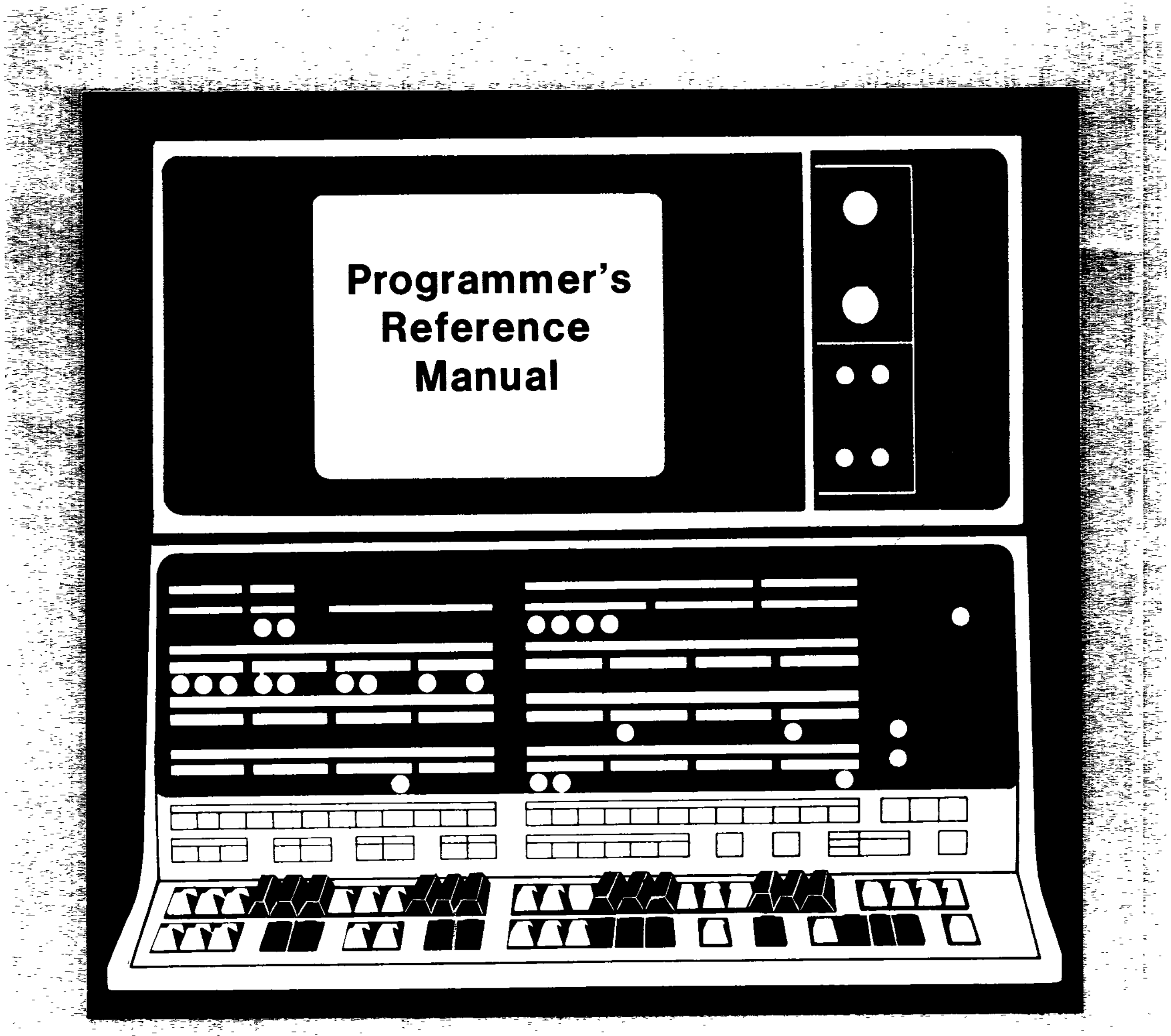
Digital Equipment Corporation
Maynard, Massachusetts

digital

PDP-12

LAP6-DIAL



Programmer's Reference Manual

PDP-12
LAP 6-DIAL
PROGRAMMER'S REFERENCE MANUAL

For additional copies of this document, order No. DEC-12-SE2B-D from Program Library,
Digital Equipment Corp., Maynard, Mass. 01754 Price $1.00

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

LAP6-DIAL is an editor, filing system and assembler for use with the PDP-12 computer. The Editor and filing portion are derived from the basic LINC program LAP6[1] by Mary Allen Wilkes of Washington University. The assembly portion is derived from several programs used for the PDP-8 computer including PAL-D[2].

The Digital Equipment Corporation wishes to express to the author, Mary Allen Wilkes (Clark), and the Computer Research Laboratory of Washington University, St. Louis, Missouri, its appreciation for the development set forth in LAP6 as well as its thanks for permission to use parts of the LAP6 program.

[1]M. A. Wilkes, LAP6 Handbook, Computer Research Laboratory Tech. Rep. No. 2, Washington University, St. Louis, May 1, 1967.

[2]PAL-D Assembler Programmer's Reference Manual DEC-D8-ASAA-D.

CONTENTS

Appendix C SUMMARIES <u>Page</u>

CHAPTER 1

USING DIAL

1.1 INTRODUCTION

LAP6-DIAL (hereafter referred to as DIAL for brevity) provides the PDP-12 user with a keyboard operating system that includes editing, assembling, and file handling capabilities. An interactive CRT display permits quick user response; a file Index and peripheral device interchange program facilitate file manipulation. The minimum hardware configuration for using DIAL is a PDP-12B system, composed of:

a. 4096-word 12-bit, 1.6 μs Core Memory

b. TC12 LINCtape Automatic Control

c. Two TU55 Magnetic Tape Transports

d. VC12 LINCscope Control and Character Buffer

e. VR12 7" x 9" CRT Display (Scope)

f. ASR33 Teletypewriter

It should be noted, however, that the DIAL character Editor is designed to be operated with an AD12 Analog-to-Digital Converter and Multiplexer.

1.2 SYSTEM DESCRIPTION

The DIAL system is provided to the user on LINCtape.[1] Each tape contains:

1. A reserved area occupied by DIAL

[1] A LINCtape contains 512_{10} blocks of 256 12-bit words each.

2. A Working Area for temporary storage of user files

3. A file area for permanent storage of user files

The DIAL area of the tape contains the DIAL Editor, Assembly and utility programs, and a file Index. The Index stores the name, starting block number, and length in blocks of each stored file. User programs are saved as named files in the file area of the system tape. DIAL tape allocation is detailed in Appendix C. The PDP-12 includes a display for viewing source programs in the Working Area. Up to 17 lines with 40 characters per line can be displayed at a time on the scope. In the edit mode, any portion of a program in the Working Area can be displayed by an appropriate line call.

1.3 SYSTEM OPERATION

A LINCtape containing DIAL must be designated as the system tape and assigned to tape transport Ø. Some DIAL operations may be performed with only one LINCtape containing DIAL, but many procedures, such as assembling programs, require two tapes. Most efficient operation is achieved when both tapes contain DIAL. When the system is started (see Appendix A), it automatically enters the edit mode. A source program may be typed in via the Teletype keyboard. The program will reside in the Working Area and will be displayed on the scope character-by-character as it is entered. The DIAL Editor may be used to add, modify, or delete characters, lines, or large sections of the program. A command may be issued also via the Teletype keyboard to a system program. When called, the Monitor writes out its buffer pointers and is replaced by the called program. When the system program operation is completed, the Monitor is automatically called back into core and retrieves its buffer pointers.

The rest of this chapter details basic concepts and terminology of the DIAL system.

1.4 SOURCE PROGRAMS

A source program is a group of lines of program input via the keyboard. Using DIAL, the program can be altered with the Editor, stored in a named file, or listed on the scope or Teletype keyboard. A line number (1 - 7777_8) appears to the left of each line on the scope and indicates the sequential location of that line in the source.

The programs are collected in blocks of memory; each block contains 512_{10} keyboard characters. Up to 64 blocks of source program can be input to DIAL. As blocks are filled, they are saved on the DIAL tape in the source program Working Area which starts at block 37Ø (see tape Ø in Appendix C). Only the program in the Working Area can be edited. If source program input exceeds 100_8 blocks or 7776_8 lines, the size of the source program must be reduced or the program can be divided into two or more parts by using the Assembler pseudo-ops SAVSYM and LODSYM, explained in Appendix C.2.

1.5 CURRENT LINE

Every source display has a current line number. By definition, it is the last line number on the display. The current line is noted by an indicator (2 dashes) on the right-hand side of the scope on the same display line as the current line. The number 1 appears as the current line number whenever the Working Area has been cleared.[1] Each time RETURN is typed to terminate a source line, the next sequential current line number appears on the scope. From 1 to 17_8 lines can be displayed at a time on the scope, as determined by the setting of A/D knob 7. After knob 7 has been set and that number of lines is present on the scope, an additional line is added at the bottom, causing the lowest numbered line to be removed from the top of the scope, thereby maintaining a constant number of lines on the display.

1.6 LINE CALLS

When using DIAL, any line in the source program can be designated as the current line. By issuing a line call, another section of the source program will be displayed on the scope with a new line as the current line. To obtain a new current line, use either of the following methods:

1. Type →n↵ where n is the number of the line to be the new current line. The right arrow indicates pressing the LINE FEED key. The display will now be positioned with line n as the last line displayed.
2. Type ALTMODE and one of the following keys:

[1]When only line 1 is seen on the scope at startup, it may merely indicate that the source program has been positioned to line 1 and the Working Area is not clear.

key	action
1	Reposition the display forward one frame
2	Reposition the display forward one line
Q	Reposition the display backward one frame
W	Reposition the display backward one line

These ALTMODE key combinations must be the first characters on a new line.

The last line of the source program can be designated as the current line by requesting a line number greater than the last line number. Similarly, the first line of the source program can be the current line by calling line Ø or line 7777.

1.7 FILES

The DIAL system tapes are file oriented. A program, either source or binary, is saved as a file in contiguous blocks of tape. Binary files require a header block for pointers. Every tape contains a file Index for the binary and/or source programs on that tape. DIAL file entries occupy blocks Ø through 267 and blocks 47Ø through 777. Blocks 346 and 347 contain the file Index. The Index records file names, starting block, and length in blocks. When a file is entered, it is given a name which must be 1 to 8 displayable keyboard characters in length, of which at least one is non-numeric. The characters slash, question mark, and comma should not be used. Only spaces in the middle or at the end are considered to be part of the name; leading spaces are ignored. Any name in the Index can describe a source and a binary program, thereby doubling the number of possible entries. Because a full Index can accommodate 63 different names, up to 126 separate file entries can be stored.

When a file is being saved, the unused file space nearest the Index within the reserved area that is large enough to contain the file being saved is the next area used. Thus, the location of entries on the tape can be controlled by their order of filing. To minimize tape movement, the most frequently used files should be placed nearest to blocks 346 - 347 during file assignment.

1.8 COMMANDS

The DIAL programs are requested through DIAL Monitor Commands. To issue a command:

1. Press LINE FEED and observe right arrow on the scope
2. Type the command
3. Press RETURN

Illegal commands are ignored and erased from the scope.

The Monitor Commands are summarized in the following table and treated in greater detail in later chapters. In this manual, a reference to unit Ø means the eighth channel of the TU55 LINC tape transport. Items in parentheses in the following table are optional and, if omitted, the user's program that was most recently manipulated is used.

Command	Function
→AS (N,U)	Assemble (U=Ø,1)
→LO (N,U)	Load Binary
→LI (L,L,)(N,U)	Assemble and List (U=Ø,1)
→QL (L,L,)(N,U)	Assemble and Quick List (U=Ø,1)
→PS (L,)(L,) (N,U)	Print Source
→SB N,U (,M)(A)	Save Binary
→SP N,U	Save Program (Source)
→AP (L,L,) N,U or B,U	Add Program (Source)
→DX (,U)	Display Index
→PX (,U)	Print Index
→CL	Clear Working Area
→PI	Peripheral Interchange
→EX	Exit
→MC (X)X,U	User's Monitor Command

Legend:

N	= File name
U	= Tape unit
L	= Line number
M	= Start Mode (L for LINC or P for PDP-8)
A	= Address (5 digits - used only if mode is specified)
B	= Tape block number
X	= Character in Accumulator
( )	= Optional parameter

The meanings of special symbols used in this manual are:

→	Press LINE FEED key	⤸	Press RETURN key

If a legal Monitor Command is issued, but can not be honored (e.g., program not on the tape) NO is displayed on the scope. Press RETURN to return to DIAL. If no unit is specified, Ø is assumed. The commands AS, LI, QL, AND SB require two LINC tapes. A binary PIP operation also requires two LINC tapes.

CHAPTER 2

THE EDITOR

2.1 USING THE EDITING CURSOR

DIAL provides a powerful and flexible character Editor which is controlled by a cursor that appears on the scope with the text. Deletion of a single character, the current line, or an entire portion of the display, and addition of text at any location in the program are features of the Editor.

The editing cursor appears on the scope as an inverted T (⊥) which moves in its own alley below a line of text and can scan up to 256 characters back from the last scope character in a display. The exact location of the cursor is determined by the setting of A/D knob 3. That setting controls how many spaces back from the last character on the scope the cursor is to be placed. After the cursor has been set, it will move along with the text at that number of spaces from the end of the text. At any time the location of the cursor may be changed by simply rotating knob 3. Rotating the knob clockwise moves the cursor to the right along the text. When the end of a line is reached, the cursor advances to the left of the next line. Similarly, rotating the knob counterclockwise moves the cursor to the left along the text.

For normal input of data, knob 3 is rotated all the way to the right. As each character is entered, the cursor follows along. When a mistake has been noted in the previous text, the cursor is used to indicate where the correction is to be made. After the correction has been completed, knob 3 is again rotated all the way to the right and input continued.

Monitor Commands and line calls can be issued only when the cursor is located at the start of a new line. Turn knob 3 all the wav to the right and press the RETURN key before issuing a Monitor Command or a line call to locate the cursor correctly.

A/D knob 7 controls the maximum number of lines displayed at a time. By rotating knob 7, from 1 to 17_8 lines can be displayed on the scope, up to a maximum of 256 characters.

For a PDP-12B system, the Right and Left Switches are used instead of knobs 7 and 3 respectively. The setting of Right Switches 8-11 determines the maximum number of lines displayed. The Left Switches can be set to values from 1 to 2047 to determine the position of the cursor. The value of the switches locates the cursor that many characters from the end of the text.

2.2 CHARACTER EDITING

A single text character[1] may be deleted or added at any point in the displayed text. To delete a character, turn knob 3 to locate the cursor under that character and type RUBOUT. The character will be removed and the rest of the line and the cursor will move one character to the left. In the following example, the letter D is deleted from line 4.

[1]To insert a line after line n (1) issue line call for line n and insert new line as the current line; or (2) position the cursor to the last character position of line n (i.e., the blank representing RETURN) and type the new line, including its RETURN.

```
before      4      JMP DTAG          (press RUBOUT)

after       4      JMP TAG
```

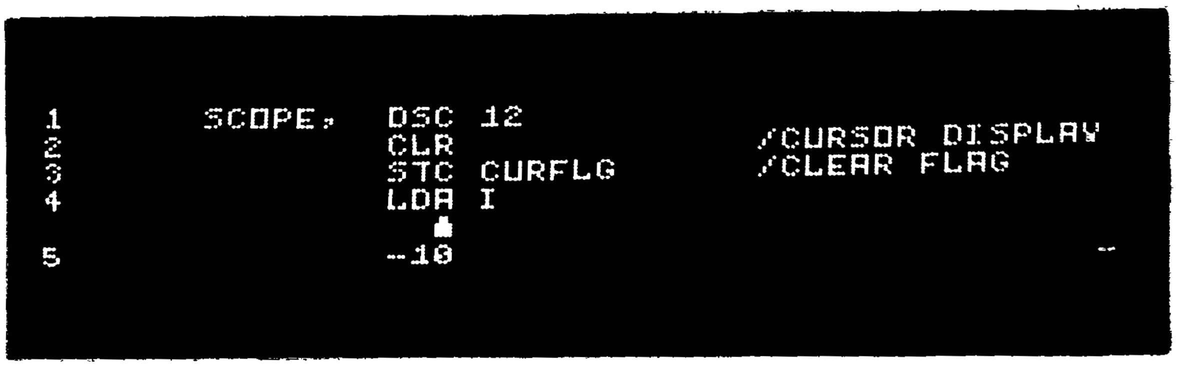
In the same manner, characters may be inserted in a line of text to the right of the character above the cursor. After typing in the additions, the cursor moves one character to the right for each character that is inserted. The characters AD are added to line 2 below.

```
before      2      T I 20            (type AD)

after       2      TAD I 20
```

2.3 CURRENT LINE DELETION

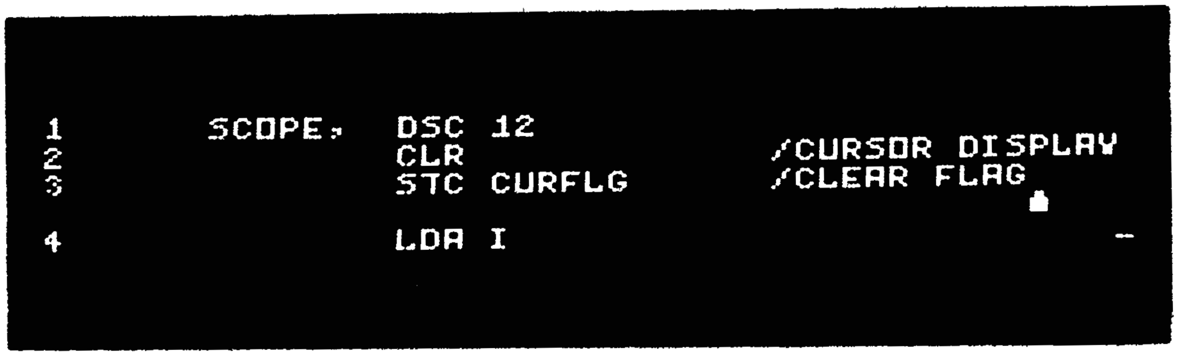
The current line of a display is the last line number visible on the scope. A source display always has a current line number and is so denoted by the horizontal indicator at the far right of the scope. The current line of a program may be deleted at any time by striking the keys ALTMODE and D. Only the current line is removed by this operation. Line 5 of the source display is omitted in the following example.

before



```
1      SCOPE,  DSC  12
2              CLR             /CURSOR DISPLAY
3              STC  CURFLG     /CLEAR FLAG
4              LDA  I

5              -10                                 -
```

after



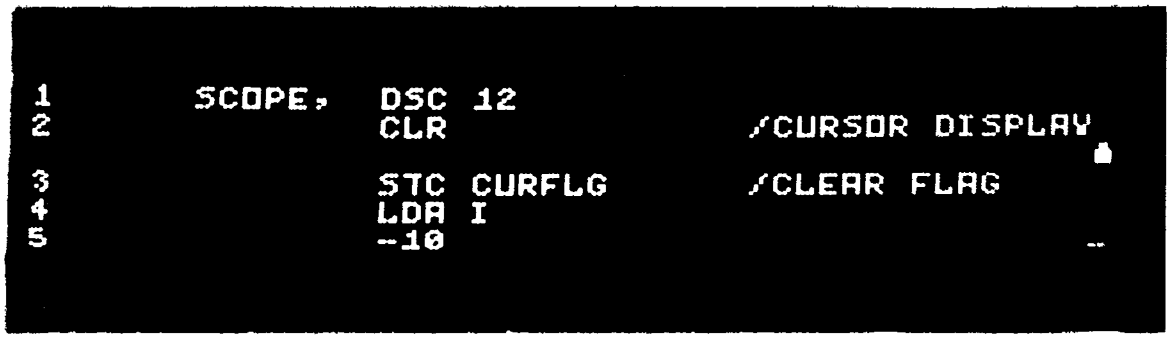
```
1        SCOPE,  DSC 12
2                CLR                 /CURSOR DISPLAY
3                STC CURFLG          /CLEAR FLAG

4                LDA I
```

Any line of the source program may be displayed as the current line. The two ways to call the current line described in section 1.6 are used with the Editor. Both require the cursor to be the first character on a new current line.

2.4 LARGE SECTION DELETION

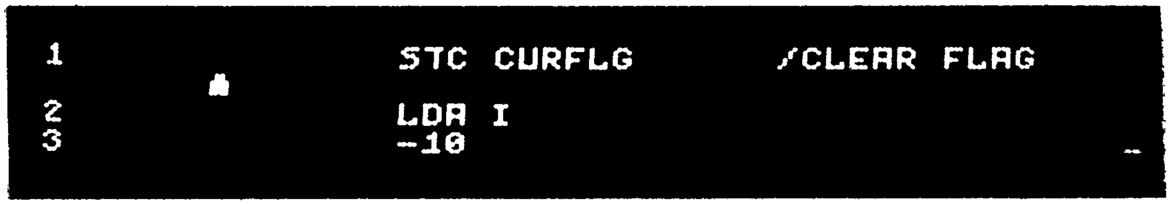
To delete a large section of the displayed text, the Editor has a command that permits deletion of all the displayed text to the right or left of the cursor. By setting the cursor at the last letter to be deleted from a string of text to its left and typing the keys ALTMODE and L, the character above the cursor and all the characters to its left on preceding lines of the display are deleted. When the delete left operation is performed, the characters on the line to the right of the cursor and sufficient lines after it are displayed to fill the number of lines defined by knob 7. The cursor is located the same number of spaces from the end of the text as it had been prior to the delete left command. In the following example, all characters on lines 1 and 2 are deleted.

before



```
1        SCOPE,  DSC 12
2                CLR                 /CURSOR DISPLAY
3                STC CURFLG          /CLEAR FLAG
4                LDA I
5                -10
```

after



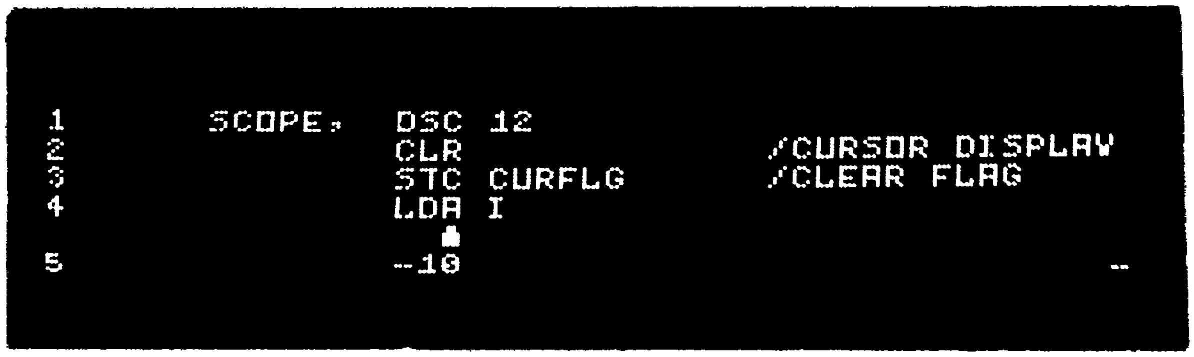
```
1                STC CURFLG          /CLEAR FLAG
2                LDA I
3                -10
```

To delete a large section of code:

1. Turn knob 3 all the way to the right.
2. Turn knob 7 all the way to the right.
3. Type a line call so the last line to be deleted is the current line.
4. Type ALTMODE and L as needed until the first line of the section of code to be deleted appears on the scope.
5. Type ALTMODE and D to delete the program through that line.

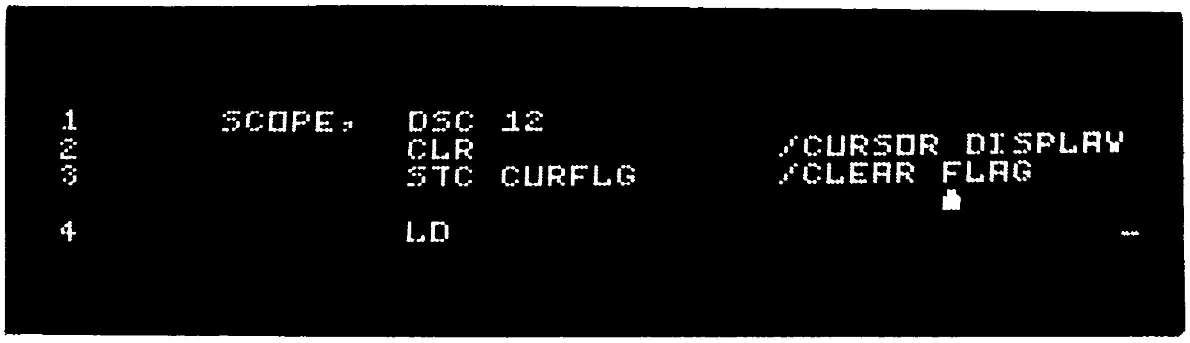
Typing the keys ALTMODE and R performs a delete-right operation, removing the character above the cursor and all the characters to its right on succeeding lines of the display. The remaining text is redisplayed with enough preceding lines to satisfy the knob 7 displayed line number requirement. The cursor is positioned the same number of spaces from the new end of the text. In the following example, the cursor is located eight characters from the end of the text. (Remember that tabs and carriage return count as one character each.)

before



```
1        SCOPE,  DSC 12
2                CLR                     /CURSOR DISPLAY
3                STC CURFLG              /CLEAR FLAG
4                LDA I

5                -10                                   -
```

after



```
1        SCOPE,  DSC 12
2                CLR                     /CURSOR DISPLAY
3                STC CURFLG              /CLEAR FLAG

4                LD                                    -
```

2.5 CURRENT LINE FORMATTING

When a new line is typed in as the current line of the displayed text, it is automatically formatted by the Editor. The text line is considered to be composed of 3 fields, each with an allotted number of displayable line spaces. The first 8 spaces are provided for the tag field, the next 16 for the instruction field, and the last 16 for the comment field. A horizontal tab is taken as 8 scope spaces, thus permitting 5 tabs per line. When a new line is encountered, the first characters are displayed in the instruction field, unless the first character was a slash. If a comma is then typed before the seventh character is input, the preceding characters are moved to the tag field and subsequent input is displayed starting from the instruction field. This operation is demonstrated with the input line TAG,ISZ 300.

```
          during                TAG


          after      TAG,       ISZ 300
```

If a slash is encountered as any character but the first on a line, it is positioned in the comment field along with the characters typed after it before a carriage return. If a slash is the first character of a line, it is displayed in the tag field. Consider the following user input:

Keys typed	Displayed as		
/L	/L		
P/L		P	/L
P,/L	P,		/L
P tab /L		P	/L

Note that text displayed on the scope and text printed on the Teletype by the commands PS, LI, or QL will have the same format.

2.6 HANDLING LARGE PROGRAMS

DIAL can edit or assemble programs that are up to 64 blocks long. (Assuming an average of 16 characters per line, this will accommodate about 2048 lines of code.) Only Monitor Commands or deletions can be accepted by the Editor when the program is 64 blocks long. To facilitate the processing of programs whose source will be greater than 64_{10} blocks, the program can be edited in two or more sections and then combined during Assembly by using the assembly pseudo-ops SAVSYM and LODSYM (refer to Appendix C.2).

If a line is entered when the Working Area is full, the line is deleted when RETURN is typed. If any corrections are required to a source of that length before assembly, one whole block must be deleted from the file first. Use the ALTMODE and L deletion procedure to remove one block from the program. (The LINC tape will move when this has been accomplished.) Use the EXIT command (see section 5.7), then press the CONT key on the computer console to perform the required editing while maintaining the current source program.

The Editor will not accept more than 120 characters on a line. A carriage return is automatically generated as the 121st character. This 120 character limit holds both before and after any editing operation, such as deleting several carriage returns to make one long line.

2.7 ASSEMBLY BUFFERS

As the core buffers are being filled, the Editor writes them out on tape. While a tape operation is in process, up to 20 additional characters may be typed. The display will not be updated with the additional characters until the tape operation has been completed. However, the characters are echoed on the Teletype to assure the user that his input is being accepted. If the auxiliary buffer space is filled during a tape operation, the Editor will indicate this to the user by not echoing the characters on the Teletype.

2.8 USE OF EXIT

At the conclusion of an editing session, an EXIT command (→ EX↲) should be given to be sure the Editor has saved its pointers on tape, thus enabling resumption of the program at a later time.

2.9 LEFTMOST CURSOR POSITION

If the setting value of the cursor is ever greater than the number of characters on the scope, the cursor moves along at the value determined by the last character on the scope at the time the cursor was set. When enough characters have been added to equal the cursor setting value, the cursor jumps back from its previous value to the location determined by its actual knob 3 setting. For example, if the cursor was set to 10 characters from the end of the text, but only 6 characters are displayed at first, the cursor will jump back when four more characters AG,T have been added.

```
          before          TAD 20
                          ▲

          during          TAGTAD 20
                             ▲

          after           TAG,TAD 20
                          ▲
```

CHAPTER 3

ASSEMBLY LANGUAGE

The DIAL Assembler processes a source program by translating mnemonic operation codes into binary codes for machine instructions, relating symbols to their numeric values, assigning absolute core addresses for data and instructions and listing the program with error messages.

This section discusses the DIAL syntax and semantics.

3.1 STATEMENT SYNTAX

DIAL source programs are usually prepared on a Teletype with the aid of the Editor, as a sequence of statements. Each statement is written on a single line and is terminated by a carriage return. DIAL statements can be typed in without having to adhere to a strict column format.

There are four fields in a DIAL statement; they are identified by the order of appearance in the statement, and by the separating, or delimiting, character which follows or precedes the field. Statements are written in the general form:

```
label, operator operand /comment
```

The Assembler interprets and processes these statements, generating one or more binary instructions or data words, or performing an assembly process. A statement must contain at least one of these fields and may contain all four.

1. Labels

   A label is the symbolic name used in the source program to identify the position of the statement in the program. If present, the label is written first in a statement and terminated by a comma. A mnemonic machine instruction (see Appendix C.4) or pseudo-op (see Appendix C.2) may not be used as a label.

2. Operators

   An operator may be one of the mnemonic machine instruction codes (see Appendix C.4), or pseudo-op codes which directs assembly processing (see Section 3.5). Operators are terminated with a space if an operand follows or with a semicolon, slash, or carriage return. Note that a semicolon located by the Assembler anywhere in a statement, except in a comment, will terminate the line at that location and increment the program's line numbers by 1 line for each semicolon; thus the line numbers on an Assembly listing will not agree with those seen on the scope.

3. Operands

   Operands are usually the symbolic address of the data to be accessed when an instruction is executed, or the input data or arguments of a pseudo-op. In each case, interpretation of operands in a statement depends on the statement operator. Operands are terminated by a carriage return, semicolon or slash.

4. Comments

   The programmer may add notes to a statement following a slash character. Such comments do not affect assembly processing or program execution, but are useful in the program listing for later analysis or debugging.

3.2 <u>SYMBOLS</u>

<u>Types of Symbols</u>

There are two main groups of symbols.

1. Permanent Symbols

   The assembler has in its permanent symbol table definitions of its operation codes, operate commands, and many input-output transfer (IOT) microinstructions (see Appendix C.4). Any symbol in the Assembler's permanent symbols may be used without prior definition by the user.

   Initially, the Assembler's permanent symbol table in memory contains the mnemonic op codes of the machine instructions of LINC mode programming and the Assembler pseudo-op codes. The symbols for PDP-8 mode programming remain on tape. As the source program is processed, symbols defined in the source program are added to the user's symbol table.

   If the programming mode is changed to PDP-8 mode, the pseudo-op PMODE must precede the new program input. This instructs the Assembler to retrieve the PDP-8 mode permanent symbols in memory. (The same LINC memory block is used for the permanent symbols of both modes.) Similarly, LMODE must precede a change to LINC mode programming.

2. User-defined Symbols

User-defined symbols, to be used as statement labels, operators, or operands, are composed according to the following rules:

a. The characters must be alphabetic (A-Z) or numeric (0-9).

b. The first character must be alphabetic. Leading numeric characters are ignored.

c. Only the first six legal characters of any symbol are meaningful to the Assembler; the remainder, if any, are ignored.

d. The Assembler assigns values according to the following rules:

(see section 3.6 for definition of LMODE and PMODE).

		Used after	
		LMODE	PMODE
Defined	LMODE	10 bits[1]	12 bits
after	PMODE	12 bits	12 bits

e. The maximum number of symbols is 876.

The programmer has 2 methods of specifying the value to be assigned to a symbol.

1. When the first symbol of a statement is terminated by a

[1]Note that no check is made on expression arithmetic. For example, if TAG = TAG1+TAG2 where TAG1 = 1777 = TAG2, then TAG = 3776, which is more than 10 bits.

comma, it is assigned a value equal to the Current Location Counter (CLC). Any instruction or data word in the program may be so "labeled".

For example:

```
        *100
TAG,    CLR
        JMP A
  B,    0
  A,    STC B
```

The symbol TAG is assigned a value of 0100, the symbol B, a value of 0102, and the symbol A, a value of 0103.

If a programmer attempts to define the same symbol as a label again, it is redefined as the user requested, but the error message ID is given.

2. The programmer may insert new symbols with their assigned values directly into the symbol table by using a direct assignment statement of the form

```
symbol=value
```

where the value may be a number or expression.

For example:

```
ALPHA=5

BETA=17
```

There must be no spaces between the symbol and the equal sign.

A direct assignment statement may also be used to give a new symbol the same value as a previously defined symbol.

```
BETA=17
GAMMA=BETA
```

The new symbol, GAMMA, is entered into the user's symbol table with the value 17.

The value assigned to a symbol may be changed.

```
ALPHA=7
```

changes the value assigned in the first example from 5 to 7.

Direct assignment statements do not generate instructions or data in the object program. These statements are used to assign values so that symbols can be conveniently used in other statements.

Use of Symbols

Symbols are used in 3 ways.

1. A symbol may be used to label an instruction or data word at any point in the program. The label must appear first in the statement and must be immediately followed by a comma. If the symbol is redefined later, the illegal definition error message is printed.
2. Symbols used as operators must be predefined by the Assembler or by the programmer. If a statement has no label, the operator may appear first in the statement, and must be terminated by a space, tab, semicolon, or carriage return. The following are examples of operators:

TAD	a permanent symbol
OCTAL	an Assembler pseudo-op
ZIP	legal only if defined by the user

3. Symbols used as operands should have a value defined by the user. These may be symbolic references to previously defined labels where the arguments to be used by this instruction are to be found, or the values of symbolic operands may be constants or character strings.

3.3 NUMBERS

Any sequence of numbers delimited by a slash, semicolon, tab or carriage return is interpreted numerically by the Assembler.

```
1/ COMMENT
12;
4372
```

The radix control pseudo-ops indicate to the Assembler the radix to be used in number interpretation. The pseudo-op DECIMAL indicates that all numbers are to be interpreted as decimal until the next occurrence of the pseudo-op OCTAL. The pseudo-op OCTAL indicates that all numbers are to be interpreted as octal until the next occurrence of the pseudo-op DECIMAL. The radix is initially set to octal and remains octal unless otherwise specified.

3.4 EXPRESSIONS

The arithmetic and logical operators used in numerical operations are:

+	Plus	2s complement addition (modulo 4096) after a PMODE pseudo-op. 1s complement addition after an LMODE pseudo-op.

-	Minus	2s complement subtraction (modulo 4096) after a PMODE pseudo-op.
		1s complement subtraction after an LMODE pseudo-op.
!	Exclamation Mark	Boolean inclusive OR (union).
&	Ampersand	Boolean AND (intersection)
␣	Space	Interpreted as inclusive OR when used to separate two symboloc operators. For example:

```
TAG,   CLA CLL
```

Note that there should be no spaces between operands and the above operators.

Symbols and numbers (exclusive of pseudo-op symbols) may be combined by using the arithmetic and logical operators to form expressions. Expressions are evaluated from left to right. These operations are shown in the following example.

MODE	A	B	A+B	A-B	A!B	A&B
PDP-8	0002	0003	0005	7777	0003	0002
	0007	0005	0014	0002	0007	0005
	0700	0007	0707	0671	0707	0000
LINC	0002	0003	0005	7776	0003	0002

3.5 ADDRESS ASSIGNMENTS

The Assembler sets the origin, or starting address, of the source program to absolute address 4020, which may then be changed by the programmer. As source statements are processed, the Assembler assigns consecutive memory addresses to the instructions and data words of the object program. This is done by incrementing a location counter each time a memory location is assigned. A statement which generates a single object program storage word increments the location counter by one. Another statement may generate six storage words, thus incrementing the location counter by six. Direct assignment statements and most Assembler pseudo-ops do not generate storage words and therefore do not affect the location counter.

The special character . (period)[1] always has a value equal to the value of the Current Location Counter. It may be used as any integer or symbol (except to the left of an equal sign). The following is equivalent to JMP 0202.

```
*200
JMP .+2
```

The next example will produce in location 0300 the quantity 700.

```
*300
.+400
```

Consider the next example.

```
*20
LMODE
CALL=JMP .
0027
```

The second line, CALL=JMP . , does not increment the Current

[1]Period must be preceded by a delimiter or operator, or erroneous code may result.

Location Counter, therefore 0027 is placed in location 20 and CALL is placed in the user's symbol table with an associated value of 6020 (the octal equivalent of JMP .).

Legal Characters

Programs processed under the DIAL Assembler are prepared by the system in the Assembler's internal code. See Appendix C.3 for a complete list of the characters with their 6-bit octal equivalents.

The following characters are acceptable by the Assembler:

a. The alphabetic characters

ABCD...XYZ

b. The numeric characters

0123456789

c. The special characters

	Space	Separates symbols and numbers (see expressions in section 3.4)
+	Plus	Combines symbols or numbers (add)
-	Minus	Combines symbols or numbers (subtract)
!	Exclamation Mark	Combines symbols or numbers (inclusive OR)
↵	Carriage Return	Terminates a statement or a line
→	Tabulation	Formats symbols or numbers for source tape output
,	Comma	Assigns symbolic address

=	Equal Sign	Direct assignment of symbol values
;	Semicolon	Terminates coding statement (will not terminate comment)
*	Asterisk	Sets current location counter; redefines origin
.	Period	Has value equal to current location counter
\	Backslash	$X\backslash Y = 1000_8X+Y$
/	Slash	Indicates start of comment
&	Ampersand	Combines symbols or numbers (AND)

All characters other than those listed above are illegal when not in a comment or TEXT field and, being illegal, their occurrence causes the error message IC (Illegal Character) to be printed by the Assembler.

3.6 PSEUDO-OPERATORS

The programmer may use pseudo-operators (pseudo-ops) to direct the Assembler to perform certain tasks or to interpret subsequent coding in a certain manner. Some pseudo-ops generate storage words in the object program, other pseudo-ops direct the Assembler on how to proceed with the assembly. Pseudo-ops are maintained in the Assembler's permanent symbol table. Do not use pseudo-ops as variable names; erroneous logic and code may result without generating an error message.

The function of each Assembler pseudo-op is described next.

PMODE

The Assembler can assemble either LINC instructions (coding) or PDP-8 instructions. Each has a pseudo-op to designate its assembly mode. PMODE indicates that PDP-8 coding follows. The Assembler remains in PDP-8 mode until explicitly changed to LINC mode by an LMODE pseudo-op.

LMODE

To designate LINC mode coding, the pseudo-op LMODE is used. The initial mode of the Assembler is LMODE and will remain in LMODE until the PMODE pseudo-op is given.

SEGMNT n ($0 \leq n \leq 7$)

SEGMNT resets the CLC to the first location of segment n, where n is an integer, a previously defined symbol, or a symbolic expression. Each memory bank is divided into four 1024-word segments. For example:

SEGMNT 2	sets CLC to location 4000
SEGMNT 1	sets CLC to location 2000

Without an argument, the CLC is reset to the first location of the succeeding segment.

FIELD n ($0 \leq n \leq 1$)

The FIELD pseudo-op indicates the field 0 or 1. If the field is not specified, 0 is assumed as the initial condition.

DECIMAL

Integers are usually taken to be octal numbers. However, following the pseudo-op DECIMAL (and prior to a succeeding OCTAL pseudo-op) all numbers are interpreted as decimal.

OCTAL

OCTAL is used to reset the radix to its original octal base. It should be noted that if a decimal number is specified when the radix is octal, the Assembler tries, often unsuccessfully, to interpret the number.

NOLIST

NOLIST is used to prevent printout during an LI assembly. The NOLIST appears in the listing, but then the printout is suppressed until a LIST pseudo-op is encountered.

LIST

LIST is used to negate the NOLIST state. The LIST statement does not appear in the listing.

TEXT

The pseudo-op TEXT enables the user to represent a character or string of characters in USASCII code trimmed to six bits and packed two characters to a word. The numerical values generated by TEXT are left-justified in the storage words they occupy, with the unused bits of the last word filled with 00.

A string of text may be entered by giving the pseudo-op TEXT followed by a space, a delimiting character, a string of text, and the same delimiting character.

For example:

```
              TEXT      ZSTRING OF TEXTZ
```

If this example is at location 0200, the listing is as follows:

```
              200       2324
              201       2211
              202       1607
              203       4017
              204       0640
              205       2405
              206       3024
                               TEXT ZSTRING OF TEXTZ
```

The first printing character following TEXT is taken as the delimiting character, and the text string is the characters which follow until the delimiting character is again encountered. Any legal character may be used as a delimiting character.[1] The Assembler will print out all the code, then print the entire line after it when a carriage return is encountered. If the text goes past the end of the manuscript the error message IE (illegal expansion) is printed and the assembly is immediately terminated.

<u>PAGE n</u>   ($0 \leq n \leq 40_8$)

PAGE is used to reset the Current Location Counter to the first location of the specified PDP-8 page. If n is not specified, the Current Location Counter is reset to the first location of the

[1]When using the TEXT pseudo-op, remember that the formatting feature of the Editor may insert unwanted tabs when a comma, slash, or carriage return is encountered.

succeeding page.

EJECT

When a program is to be listed on a line printer, the command EJECT causes the next line to be printed at the top of the next line printer paper page, thus permitting a logical splitting along page boundaries.

ASMIFx n

The conditional assembly pseudo-ops, of the general form ASMIFx, can have 3 possible forms:

Pseudo-op	Meaning
ASMIFZ n	Assemble if zero
ASMIFN n	Assemble if non-zero
ASMIFM n	Assemble if minus

In each case the expression is evaluated and if the value of the expression matches the condition specified (zero, non-zero, minus), the next line is assembled. If it does not match, then the next line is not processed by the Assembler. The next line may be any statement, but will most frequently be an ASMSKP pseudo-op (discussed below). Sections of code may thus be altered or entirely deleted from a program just be setting some values in the beginning of the assembly program.

ASMSKP n

When the pseudo-op ASMSKP and the expression n following it are encountered, the expression on the riqht is evaluated. The number of lines equal to that value of that expression, starting on the next line, are not assembled. Thus, the ASMIFx class of pseudo-ops control the assembly of one statement; the ASKSKP controls any number of lines and can be used to eliminate entire blocks of statements.

For both the ASMIFx family and ASMSKP pseudo-ops, if no expression is present, the next statement will be assembled. If an attempt is made to go past the end of a manuscript with one of these pseudo-ops no error results; the end of the manuscript is simple treated as the end of the skip block. With both the ASMIFx and the ASMSKP pseudo-ops a line may be a normal assembly instruction or it may be just a comment.

Note that with the ASMIFx family and ASMSKP pseudo-ops, the condition "zero" must be specified by ØØØØ.

Consider the following program listing.

```
ØØ42                           ASMIFN TAPE
ØØ43     Ø2Ø2      44Ø1        JMS I WRITE
ØØ44                           ASMIFN TAPE-DISK
ØØ45                           ASMSKP 2
ØØ46                           JMS I READ
ØØ47                           JMS I WRITE
ØØ50     Ø2Ø3      1ØØ5        TAD M 1Ø
```

The expression after ASMIFN pseudo-op at line 42, TAPE, had a non-zero value so the next line, 43, was assembled. The value of the expression TAPE-DISK also had a non-zero value, causing line 45 to be executed, thus skipping lines 46 and 47. Assembly was continued at line 50.

The last 2 pseudo-ops, SAVSYM n and LODSYM, are sophisticated tools which should be used only in large system programs. These pseudo-ops are discussed briefly here and in detail in Appendix C.2.

<u>SAVSYM n</u> (n=1 or 2)

The pseudo-op SAVSYM n allows the programmer to save part or all of his user symbol table for use in later assemblies. It is followed by an expression, n, which is evaluated to a value of 1 or 2. There are 2 cases when a user will want to save his symbols. Case 1 is when the user has defined some common definitions which are to be used with all his programs; case 2 is when the program is too large and must be split into 2 or more smaller programs that can communicate with each other.

<u>LODSYM</u>

The pseudo-op LODSYM loads a symbol table previously saved by a SAVSYM pseudo-op. When a LODSYM command is given all previously defined user symbols in core are erased. For this reason, a LODSYM should be one of the first statements in a program. SAVSYM and LODSYM permit the user to do an assembly, save the symbol table, do another assembly and call back in the original symbol table.

CHAPTER 4

ASSEMBLY PROGRAMS

4.1 ASSEMBLE PROGRAM

```
→AS (NAME, UNIT) ↵
```

NAME = name of filed program to be assembled

UNIT = tape unit on which named file is to be found (Ø or 1)

The Monitor Command AS performs an assembly of the NAMEd source file on the specified UNIT. If no NAME is given, the source program in the Working Area on unit Ø is assembled. With the command AS, an assembly listing is not produced, but error messages with line numbers and a tag table are printed. (See Appendix B for the assembly error messages.) Note that the assembly and tag table printout can be stopped and the Editor called back by typing RETURN.

The Assembler requires that the program to be assembled be on tape unit Ø or 1. The Working Area of tape unit 1 is always used for binary output and tag table storage.

For all DIAL programs, NAME can be 1 to 8 characters long and must have at least 1 non-numeric character and no ? or / characters.

4.2 LIST PROGRAM

```
→LI (LINE NUMBER1, LINE NUMBER2,) (NAME, UNIT) ↵
```

LINE NUMBER 1	= starting line number
LINE NUMBER 2	= terminating line number
NAME	= name of filed program to be listed
UNIT	= tape unit on which NAMEd file is to be found (Ø or 1)

The list program performs the same functions as the assemble program, but in addition produces an octal-symbolic listing on the Assembler output device. It will assemble and list from the workspace if no program NAME and UNIT are specified. The value of UNIT can only be Ø or 1. As with the assemble command, AS, the Working Area on the tape unit 1 is used for the binary output and tag table.

If two line numbers are supplied, it will cause the entire program in the workspace or a NAMEd file to be assembled and will list only the portions of the program between the two line numbers. To assemble and list only that part between lines 140 and 160 type

```
→LI 140,160 ↵
```

To assemble and list lines 300 to 310 of file MART on unit 1, the correct command is

```
→LI 300,310 MART,1 ↵
```

If a line printer is available and in the ready state, the listing will be output to that device. By using the pseudo-operators NOLIST and LIST, the output listing can be controlled.
RETURN may be typed at any time to return to the Editor.

4.3 QUICK LIST

→QL (LINE NUMBER 1, LINE NUMBER 2,) (NAME, UNIT)↵

LINE NUMBER 1	= starting line number
LINE NUMBER 2	= terminating line number
NAME	= name of filed program to be listed
UNIT	= tape unit on which NAMEd file is to be found (Ø or 1)

This command performs the same functions as the LI command with the following exceptions:

a. Line numbers and all comments are deleted.

b. All tabs are printed out as spaces.

The QL command enables the user to examine his code without having to wait the extra time to receive a full listing. The QUICK LIST feature normally saves 1 to 2 seconds per line, more if the program contains many tabs and comments. Only unit Ø or 1 may be specified. No line numbers are printed out with the QL command. To help the user determine statement line numbers from the listing, QL takes advantage of the fact that the pseudo-ops (FIELD, *, PAGE, PMODE, LMODE, etc.) generate no code and the line number is printed instead of the location counter. The assembly may be interrupted by typing RETURN to go back to the Editor.

Two listings of the same program follow to illustrate the differences between a listing produced by a list command and one generated by a quick list command.

This listing was produced by a List command.

```
ØØØØ                    *2Ø
ØØØ1                            PMODE               /START IN PDP-8 MODE
ØØØ2                            LINEP=Ø             /NO EJECTS
ØØØ3                            ASMIFN  LINEP       /EJECT?
ØØØ4                            EJECT               /YES.
ØØØ5      4Ø2Ø  6Ø32    START,  KCC                 /CLEAR KEYBOARD FLAG
ØØØ6      4Ø21  6Ø46            TLS                 /SET TELETYPE FLAG
ØØØ7                            ASMIFN  LINEP       /EJECT?
ØØ1Ø                            EJECT               /GO TO TOP OF PAGE.
ØØ11      4Ø22  6Ø31    LOOP,   KSF                 /CHAR. TYPED?
ØØ12      4Ø23  5222            JMP     .-1         /NO. WAIT
ØØ13      4Ø24  6Ø36            KRB                 /YES. READ IT IN.
ØØ14      4Ø25  6Ø41            TSF                 /TELEPRINTER READY?
ØØ15      4Ø26  5225            JMP     .-1         /NO. WAIT
ØØ16      4Ø27  6Ø46            TLS                 /YES. OUTPUT THE CHAR
ØØ17      4Ø3Ø  5222            JMP     LOOP        /GO GET ANOTHER CHAR.

ØØØØ   ERRORS

LINEP  ØØØØ
LOOP   4Ø22
START  4Ø2Ø
```

This listing was produced by a quick list command.

```
0000         *20
0001          PMODE  /
0002          LINEP=0  /
0003          ASMIFN LINEP /
0004          EJECT  /
4020   6032  START, KCC  /
4021   6046   TLS  /
0007          ASMIFN LINEP /
0010          EJECT  /
4022   6031  LOOP, KSF  /
4023   5222   JMP .-1 /
4024   6036   KRB  /
4025   6041   TSF  /
4026   5225   JMP .-1 /
4027   6046   TLS  /
4030   5222   JMP LOOP /

0000   ERRORS

LINEP   0000
LOOP    4022
START   4020
```

4.4 SAVE BINARY

→SB NAME, UNIT (,MODE) (ADDRESS)↵

NAME	=	name to be assigned to saved binary file
UNIT	=	unit on which binary file is to be saved
MODE	=	L if program should start in LINC mode
		P if program should start in PDP-8 mode
ADDRESS	=	starting address

The binary program most recently assembled with the AS, LI, or QL command can be saved with the SB command as file NAME on the specified UNIT.

The SB command has a load and go option so that when a program is loaded into memory with the LO command, it will also automatically start to be executed. To use this option, the program mode must be specified.

If the program is to be started in LINC mode whenever loaded, the unit number is followed by an L. If the program is to start at Ø4Ø2Ø, the L is followed by typing RETURN. If the program is to be started elsewhere, a full five digit address must be specified. The Data Field is always set to three when the program is started.

If the program is to be started in PDP-8 mode, the unit number is followed by a P which will cause the program to be started at location ØØ2ØØ; otherwise the full five digit starting address should be specified.

If the SB command is terminated after UNIT with a carriage return, the loader will halt to location 7774 after having loaded the program.

For example:

To save the program PGMNAM on unit 1 so it will load and start in LINC mode at location Ø4Ø2Ø

```
                →SB PGMNAM,1,L )
```

To save the program PGMNAM on unit Ø so it will load and start in LINC mode at location 4Ø2Øof the second band (IF = 6)

```
                →SB PGMNAM,Ø, L14Ø2Ø )
```

To save the program PGMNAM or unit Ø so it will load and halt

```
                →SB PGMNAM,Ø )
```

With each assembly, a binary header block is generated by the Assembler which maps the memory blocks used. The Save and Load Binary commands use the data in the header to save or load the appropriate blocks. (Block 4000 is always saved). Thus, if a program occupied blocks 4000 and 4400, three blocks would be saved: the two program blocks and the header block. If a previous version of the same program is already present on the specified tape, DIAL will display REPLACE?. Strike R to replace the existing file entry with this new entry; strike RETURN if the existing file is not to be replaced.

4.5 LOAD BINARY

→LO (NAME, UNIT)↵

NAME = name of binary file to be loaded

UNIT = unit from which file is to be loaded

If the NAMEd program was saved by a Save Binary command and no mode was specified, the program is loaded into core memory and the computer halts at location 7774. When a UNIT is also specified, the NAMEd binary program is read into memory from the selected file. If no NAME is given, the last binary program assembled by DIAL is read into memory from unit 1 and the computer again halts at 7774.

If the file had been saved with a mode and address specified in the Save Binary command, then the Load Binary program will load and go. Program execution will start automatically from the NAMEd file on the specified UNIT. This command will exit from DIAL before reading the binary program. Note that the loader overlays locations 7761-7777 with a bootstrap routine and these locations, therefore, cannot be used by the program.

4.6 ASSEMBLY OPERATION NOTES

The Assembler looks at the tape on unit 1 to determine if there is a copy of the Assembler on tape 1. If there is a copy, then symbols can be swapped in and out on tape unit 1 rather than on unit Ø, resulting in a considerable saving of tape motion and time. Note that it is not necessary to have an Assembler on unit 1, but it is faster.

For users with 8K of core, the Assembler uses the extra 4K rather than swap user symbols in and out from the tape. In addition, the pseudo-op processor will then always be resident and double buffering from the input file can be used. Such a system greatly increases the speed of assembling a program over a 4K machine.

For an 8K machine with a line-printer, the Assembler will automatically give its listing and symbol table on the line-printer, if it is in the START or READY status. Otherwise, the output will go to the Teletype.

CHAPTER 5

UTILITY PROGRAMS AND COMMANDS

5.1 SAVE PROGRAM

```
→SP  NAME, UNIT ↵
```

NAME = name to be assigned to saved program in file

UNIT = unit to contain the named program

DIAL saves the source program by NAME in one file on the UNIT specified. When saving a program, RETURN may be typed at any time. This will interrupt the command and return to the source display, with no effect once DIAL has updated the Index. To prevent saving two programs with the same NAME, DIAL displays REPLACE?. The user may either type R to replace the file entry with this source or press RETURN to keep the old file entry.

5.2 ADD PROGRAM

```
→AP BN, UNIT ↵
→AP (LN1, LN2,) NAME, UNIT ↵
```

BN = First block number of source program

NAME = name of filed program

UNIT = unit on which program is located

To add DIAL source to the current source at the current line, the AP command requires specifying only its starting block number, BN, or its NAME. Two line numbers may be specified to add that portion of the NAMEd program to the current source.

The first line of the program is added after the current line on the scope. Source lines which follow the added source are then renumbered; if there is no current source, e.g., a CLEAR command was issued, the added source will be the entire source. If the arguments are omitted, the command is ignored.

5.3 CLEAR

```
→CL↲
```

The Working Area on the tape unit Ø can be cleared by using the command CL. DIAL remains in core and is restarted with a clean buffer area. →CL may be typed at any time to clear the Working Area without having to manipulate any console switches.

5.4 DISPLAY INDEX

```
→DX(,UNIT)↲
```

UNIT = unit whose Index is to be displayed

The file Index of the specified UNIT is displayed on the scope by the command →DX. For each program, its name, source or binary, starting block number, and length in blocks is indicated.

To view the entire Index, use the following keys:

Key	Action
1	Forward one frame
2	Forward one entry
Q	Backward one frame
W	Backward one entry

Press the RETURN key to return to the source display at any time. Entries may be deleted from the display also. Pressing the RUBOUT key will delete the last line on the display. If the wrong entry is deleted, type R to restore the Index. The deletions are made permanent by pressing the colon key (:). The source display is returned to the scope. Pressing the RETURN key will cause a return to the source display after one of the above operations. Note that if there is no Index or an empty Index, NO is displayed on the scope.

5.5 PRINT INDEX

```
→PX(,UNIT)↵
```

UNIT = unit of Index to be printed

The command →PX prints out the contents of the specified Index on the Teletype. Press RETURN at any time to stop the printout and to return to the source display.

5.6 PRINT SOURCE

```
→PS (LN1,)(LN2,)(NAME,UNIT)↵
```

NAME = name of file to be printed

UNIT = unit on which named file is to be located

LN1 = starting line number

LN2 = terminating line number

The NAMEd source file is printed on the Teletype from the specified UNIT. The current source is printed when no NAME and/or UNIT are designated. If two line numbers are specified, that portion of the NAMEd file will be printed.

Any DIAL source can be printed with the PS command. No words will be split between Teletype lines. Printing time is approximately 1 minute per page for PDP-12 program sources.

Line numbers, if specified, provide inclusive bounds for the print-out. When only one line number is specified, it is assumed to be the initial bound and the rest of the source is printed.

5.7 EXIT

→EX ↵

The Exit command completes the updating of the working area from the memory buffers, thus assuring the user of leaving DIAL without losing the current source program. After →EX, DIAL halts. Press the CONT console switch to return to DIAL.

5.8 USER'S MONITOR COMMAND

→MC X(X),U ↵

X = value in left of Accumulator

(X) = value in right of Accumulator (optional)

U = unit to be read

The User's Monitor command allows the DIAL user to define an individualized function to the Editor. The MC command is similar to the Load Binary command, but with twc additional features:

1. Block 270 of the free area is read into core.
2. Arguments are passed by the Editor using the Accumulator.

When the MC command is issued, block 270 of unit U is read into core locations 4000 to 4377 and the program is started in LINC mode at location 4020. Block 270 is the first block of an eight block free area on the DIAL tape and is accessed by the MC command. Since DIAL will never use these blocks, the user can retrieve any or all by using absolute references; there is no need to access the DIAL directory. The DIAL code of the alphabetic character X is placed in the left half of the Accumulator. The value of the second character, which can be alphabetic or numeric, is placed in the right half of the Accumulator. If the second character is omitted, the value 77 is placed in the right half of the Accumulator automatically.

5.9 PERIPHERAL INTERCHANGE PROGRAM (PIP)

→PI ↵

PIP for the PDP-12 permits the user to transfer source or binary files between devices such as a LINC tape, high-speed paper tape reader/punch, card reader, or line printer.

PIP is a user's program, residing in the file area, and is not necessarily resident on unit Ø with the DIAL source programs. Calling PIP by the command →PI i equivalent to →LO PIP,Ø. Thus, if PIP is moved to a unit other than Ø the →PI command is inoperable. The appropriate tape unit may be substituted for unit Ø in the →LO PIP command above to call the program.

Mode Options

When PIP is started, tne following message is displayed. Note that lower case letters are used here to indicate half size characters on the scope.

```
                    PIP OPTIONS

                  a---auxiliary mode
                  b---binary mode
                  s---source mode

     reply:
```

The file to be manipulated by PIP must be described at this time. Only one of the 3 single letter abbreviations needs to be typed after reply. A carriage return terminates all PIP command strings.

This PIP display and all the subsequent ones are followed by reply: and a square-shaped cursor in the lower left hand corner. When a response is typed, it is seen at the location indicated by the cursor. The cursor moves one character to the right for each character typed in. When responding to any PIP display, RUBOUT can be typed to delete the last character typed or LINE FEED can be typed to delete the entire line. If an illegal character is typed, it is ignored and the PIP display is returned to the scope. Typing CTRL/P will also return the PIP OPTIONS display to the scope at any time. CTRL/D may be typed to return to DIAL.

Binary or Source Input

After the mode has been accepted, the input device must be specified. If the reply to the first PIP display was B or S, the second PIP display is shown, as follows:

INPUT DEVICE

h---high speed reader

l---linc tape

t---teletype

c---card reader

reply:

The auxiliary mode options are described later in this section.

If LINC tape is to be the input device, the user's response to the second PIP display must be in the format Ln;NAME where L indicates the LINC tape option, n is the LINC tape input unit number (Ø - 7) followed by a semicolon, and NAME is a 1 to 8 character file name. Thus, to input a file named ABC3 from LINC tape unit 4 the correct command is L4;ABC3.

If source mode was chosen as the first option, then data may be input from the card reader by typing a reply of C↵. Columns 1-110_8 are read unless a response is given in the form Caa;THRU,bb where aa is the first column to be read and bb is the last column to be read. To read columns 50 to 110 (octal) the correct string is C50;THRU,110.

The character codes used are not the card reader codes in the PDP-12 User Handbook. Instead, they are the standard IBM-029 Keypunch codes (see Appendix B of "Introduction to Programming" DEC-C-18).

There are some minor changes in that set to be compatible with standard ASCII.

CARD CODE	029 CHARACTER	DIAL CHARACTER
Ø-8-2	NONE	]
Ø-8-5	_ (Underscore)	←
11-8-7	¬ (Logical not)	\
12-8-2	¢ (Cent Sign)	L
12-8-7	| (Vertical Bar)	↑

If the user desires he may substitute an entirely different character set into PIP (with the exception of BLANK).

If, for binary input, the device is to be the high-speed reader or the Teletype, neither of which is file or unit oriented, the user's response is in the form DF;Mode,Address where D is the device abbreviation (H or T), and F is the memory field and is specified only for field 1 or larger. Mode is indicated by L for LINC Mode and by P for PDP-8 Mode and Address is the starting address. If Mode and Address are omitted, the program just loaded will not start. The mode must be specified for the program to start after it has been read.

The punctuation marks are always required if the items after them are specified. If no Address is given, a LINC Mode operation will start at location 4020 and a PDP-8 mode operation will start at location 200. A carriage return terminates the command string.

Consider the following examples.

H;P	Input is from the high-speed reader and will start in PDP-8 mode at location 200.
H1;P,1000	Input is from the high-speed reader and will start in PDP-8 mode at location 1000 in memory field 1.
T;L,6000	Input is from the Teletype and will start in Linc Mode at location 6000.
H	Input is from the high-speed reader and will not be started.

If the specified file is not at the indicated location, NO will be displayed on the scope. Return to PIP by typing CTRL/P or return to DIAL by typing CTRL/D at this time.

When the high-speed reader is the input device, the tape must be in the reader before the carriage return is typed to terminate the output device command string. For ASCII paper tape input, the character CTRL/Z must terminate the input. It must be present as the last character on the tape or typed on the Teletype after the tape has been read in. If the tape had originally been punched by PIP, a CTRL/Z will already be present at the end of it.

Binary or Source Output

When a response to the input device display has been accepted, the output device must be specified. The following display appears on the scope.

OUTPUT DEVICE

h---high speed punch

l---linc tape

t---teletype

p---line printer

reply:

If the LINC tape is to be the output device, a response in the same format as was used to specify LINC tape as the input device is required. Never ask to copy a file onto itself by using the same name and the same unit for both the input and output commands. This will destroy the named file. To locate the file PAUL on LINC tape unit Ø the correct command string is LØ;PAUL.

When using LINC tape as the input or output device for any PIP operation, be sure to specify the name of the file. If the file name is omitted when specifying LINC tape output, the file is assigned the name question mark (?) by PIP. The file will be accessible only by using PIP and the name ? or no name. If the file name is omitted when specifying LINC tape input, PIP can locate only a file named ? If there is no file with that name, then NO will be displayed on the scope. Only a return to PIP or DIAL can be generated at this time. If no unit is specified when using PIP to perform a LINC tape operation, unit Ø is assumed.

If there is not sufficient room on the indicated tape, NO is displayed. Type CTRL/P to return to the first PIP display or CTRL/D to return to DIAL. If a file with the same name is already located on that tape, the message REPLACE? is displayed. Type R↲ if the present file is to replace the existing file or, if it is not to replace it, type CTRL/P to return to the first PIP display or CTRL/D to return to DIAL.

For high-speed punch, Teletype, or line printer output, only the letter abbreviation is required. The punch will type leader tape automatically. When the output device command string is terminated, typing a carriage return initiates the specified operation. Note that tape punched using PIP can be read directly into DIAL by the Teletype reader.

If the input device was the paper tape reader, then after the first tape has been read in, the following display appears on the scope.

```
                MORE TAPES?
              a---read another tape
              n---no more tapes
   reply:
```

Type the appropriate letter answer. A reply of A will store the next file immediately after the first one. Each tape must be terminated by a CTRL/Z. Any operation can be interrupted by typing CTRL/P to return to the first PIP display or by typing CTRL/D to return to DIAL.

<u>Auxiliary Mode</u>

If the letter A was typed in response to the first PIP display, the following is seen on the scope.

```
                 OPTIONS
              c---copy specified blocks
              d---duplicate tape Ø onto unit 1
      reply:
```

In addition to specifying the option, a number can follow a reply of C or D to indicate the number of consecutive units onto which the specified blocks or whole file is to be transferred. A reply of D will perform that operation immediately. A response of C will produce the second PIP display requesting the input device. The only acceptable input device is LINC tape. The reply must be in the format Ln;fb,nb where fb is the first block and nb is the number of blocks to be read. Thus, L2;63,24 will start input from LINC tape unit 2, block 63, and continue for 24 blocks. Note that block numbers are octal.

When the input response has been accepted, the PIP output display is seen on the scope. The only acceptable output device is LINC tape. The response must be in the form Ln;fb where n is the number of the first consecutive tape unit and fb is the block where output is to start.

During all PIP operation, the program checks for tape errors. When one is encountered, the following message is displayed.

```
                             TAPE ERROR
                             AT BLOCK nnnn
                          a---accept as is
                          r---try again (repeat tape operation)
                          s---try to skip past error
          reply:
```

It is up to the user's discretion to choose one of the three above options. If the problem is a minor hardware error, such as the WRITE ENABLE switch was not set, then choice R can be used. If some of the block numbers have been modified, then choice S may still yield a working system. The user can always type CTRL/P to return to PIP or CTRL/D to return to DIAL. For binary tape input, the error message CHECKSUM ERROR can only be followed by a return to PIP or DIAL.

If the return commands CTRL/P and CTRL/D ever fail to operate, PIP may be restarted by stopping the processor, setting the Left Switches = 0200 and pressing the START LS console key.

The following command sequences are examples of PIP operations.

1. Copy the binary file BINFILE from unit Ø to unit 6.

```
B
LØ;BINFILE
L6;BINFILE
```

2. Copy the source file PSL from unit 3 to unit 7.

```
S
L3;PSL
L7;PSL
```

3. Copy the binary file OLDNAME from unit 1 to unit 7 and call it NEWNAME.

```
B
L1;OLDNAME
L7;NEWNAME
```

4. Duplicate the source file COPY1 on unit 2 and call it COPY2 on the same unit.

```
S
L2;COPY1
L2;COPY2
```

Warning: Never copy a file onto itself on the same unit. For example:

```
B
L1;COPY1
L1;COPY1
```

This will possibly result in the destruction of the file COPY1.

5. Punch the source file HERE on unit 3 by the high-speed punch.

```
S
L3;HERE
H
```

6. Read in a tape on the Teletype that was not originally punched by PIP and store it as file MARTY on tape unit Ø.

```
S
T
LØ;MARTY
```

When the tape is finished type CTRL/Z on the Teletype (if PIP had punched the tape originally there would be a CTRL/Z at the end of it).

Then type

N	if only this tape is to be read
	or
A	if another tape is to be read and added after the first tape.

7. Duplicate tape Ø onto tape 1.

```
A
D
```

8. Duplicate tape Ø onto tapes 1, 2, 3, 4, and 5.

```
A
D5
```

9. Copy blocks 300-317, unit 4, to blocks 200-217, unit 6.

```
A
C
L4;300,20
L6;200
```

10. Copy blocks 600-677, unit 2 to blocks 700-777 of units 5, 6, and 7.

```
A
C3
L2;600,100
L5;700
```

11. Read in a binary tape via the high-speed reader and store it as file J10 on unit Ø, where it will start at location 200 in PMODE.

```
B
H;P
LØ;J10
```

APPENDIX A

INITIAL STARTING PROCEDURE

1. Mount a DIAL tape on tape unit Ø. (Unit Ø is indicated by setting the tape channel indicator to 8 on TU55 transports.)
2. Mount another tape on unit 1, if required. (Refer to commands, chapter 1.)
3. Set the switches of both tape units to REMOTE and set unit Ø to WRITE ENABLE.
4. Set the mode switch to LINC mode and press I/O PRESET to initialize LINC Instruction Field to 2 and LINC Data Field to 3.
5. Set the Left Switches to Ø7Ø1 and the Right Switches to 73ØØ by raising the switches indicated by ↑ and pushing down those indicated by ↓ in the following diagram.

↓ ↓ ↓ ↑ ↑ ↑ ↓ ↓ ↓ ↓ ↓ ↑	↑ ↑ ↑ ↓ ↑ ↑ ↓ ↓ ↓ ↓ ↓ ↓
Left Switches	Right Switches

6. Press the DO console switch.
7. When the tape has stopped moving, press the START 20 key.

APPENDIX B

ASSEMBLY ERROR MESSAGES

Durinq source program assembly, error messages in the form of a 2 letter code are included in the program listing. These messages define illegal syntax or insufficient space errors and are explained below.

Error Code	Explanation
IC	Illegal Character - An illegal character was processed in the instruction field; the character is ignored and the assembly is continued.
ID	Illegal redefinition of a symbol - An attempt was made to give a previously defined symbol a new value by other means than the equal sign; the symbol was not redefined.
IE	Illegal expansion - Delimiter missing in text.
	Illegal equals - An equal sign was used in the wrong context.
IR	Illegal reference - An off page reference was made.
SE	Symbol table exceeded - Assembly is terminated and control is returned to DIAL; the symbol table may be expanded to contain up to 896 user symbols.
US	Undefined symbol - A symbol has been processed during pass 2 that was not defined before the end of pass 1.
WA	Working Area exceeded - Assembly is terminated and control is returned to DIAL; more than 100_8 blocks of source program have been input for assembly.
PS	Push-down stack exceeded - Too many symbols to be evaluated on one line.

APPENDIX C

SUMMARIES

C.1 COMMAND SUMMARY

All commands are issued in the form

→Command ⤸

Command	Function
AS (N,U)	Assemble (U=Ø,1)
LO (N,U)	Load Binary
LI (L,L,) (N,U)	Assemble and List (U=Ø,1)
QL (L,L,) (N,U)	Assemble and Quick List (U=Ø,1)
PS (L,)(L,) (N,U)	Print Source
SB N,U (,M)(A)	Save Binary
SP N,U	Save Program (Source)
AP (L,L,) N,U or B,U	Add Program (Source)
DX (,U)	Display Index
PX (,U)	Print Index
CL	Clear Working Area
PI	Peripheral Interchange
EX	Exit
MC X(X), U	User's Monitor Command

Legend:

( ) indicates an optional parameter

N = File Name

U = Tape Unit

Legend: (Cont.)

L	= Line Number
MA	= Mode (L for LINC or P for PDP-8) and Address (5 digits)
B	= Tape Block Number
X	= Accumulator
→	= Line Feed
⟩	= Carriage Return

C.2 PSEUDO-OPERATORS

Pseudo-Op	Mode	Operation
ASMIFM n	8/L	Assemble if n is negative
ASMIFN n	8/L	Assemble if $n \neq 0$
ASMIFZ n	8/L	Assemble if $n = 0$
ASMSKP n	8/L	Continue assembly after n lines
DECIMAL	8/L	Sets decimal radix for integer input
EJECT	8/L	Print next line at top of next page of line printer
FIELD n	8/L	Defines each 4K of memory; n = 0 or 1
I	8	Indirect addressing
LIST	8/L	Negate NOLIST condition
LMODE	8	Causes LINC-8 instructions and pseudo-ops to be defined subsequently
LODSYM	8/L	Load saved symbol table (see below)
NOLIST	8/L	Inhibit octal-symbolic listing
OCTAL	8/L	Sets octal radix for integer input
PAGE n	8/L	Start new page at n·200. If no parameter, start at next page ($0 \leq n \leq 40_8$)

Pseudo-Operators (Cont.)

Pseudo-Op	Mode	Operation
PMODE	L	Causes PDP-8 instructions to be defined subsequently
SAVSYM n	8/L	Saves symbol table for later assembly (n = 1 or 2) (see below)
SEGMNT n	8/L	Starts new segment at N×2000. If no parameter start at next segment. ($0 \leq n \leq 7$)
TEXT	8/L	Packs two 6-bit words per cell
Z	8	Page zero reference

The 2 cases requiring use of the SAVSYM pseudo-op and the procedure for assembling large programs with the SAVSYM and LODSYM pseudo-ops are discussed in detail in this section.

Case 1. Assume that the user has defined the following symbols for his program.

```
        INDEX=346       /set the index pointer to 346
        DISK=3          /3 disks
        TTYIN=7423      /address of Teletype input routine
        TTYOUT=7520     /address of the Teletype output routines
        SAVSYM 1        /now save just these symbols (see below)
        *XXXX
        YYYY            /the rest of the program
```

The command SAVSYM 1 will cause the user defined symbols INDEX, DISK, TTYIN, and TTYOUT to be saved when the program is assembled. Any user symbols occurring later in the program will not be saved. At a later time in another program a LODSYM command will load these symbols into the symbol table without having to retype them.

Case 2. If all symbols are to be saved in a common table, the SAVSYM 2 command is used. Thus, if SAVSYM 1 is replaced by SAVSYM 2 in the previous example, then the 4 user-defined symbols and all symbols to follow will be saved in the symbol table. This is useful in breaking up large assemblies, when all symbols are to be saved. If a symbol was redefined, the last definition assigned will be the one saved. Note that the more symbols defined and saved, the slower the Assembler will run.

The symbol table is always stored in the Working Area on unit 1 near the binary output and the present symbol table. With SAVSYM 1 the symbol table is saved at this point in pass 1 of the assembly. With SAVSYM 2 the symbol table is saved at this point during pass 2. The user who wants to save a set of common definitions or a common page zero with pointers should use the command SAVSYM 1 after the definitions on page zero. No other symbols will be saved. The user who wishes to save all his symbols because his program has to be split into sections should put the command SAVSYM 2 at the end of his program, assuring that all symbols will be properly defined.

In either case, the symbols will be permanently saved unless one of the following occurs:

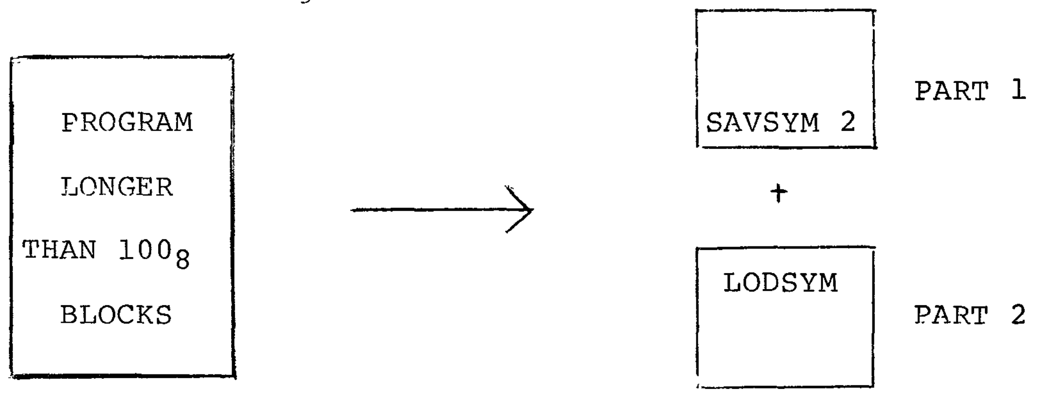
1. The tape is erased.
2. The tape is used for unit Ø workspace (source) and is overwritten.

3. Another SAVSYM command is given at some later time and replaces the old symbol table with a new one.

Assembling Large Programs with SAVSYM and LODSYM

SAVSYM and LODSYM may be used to assemble a program that is longer than 100_8 blocks, by breaking the program into several smaller files.

To avoid symbol communication difficulties, SAVSYM and LODSYM are employed in the following manner.

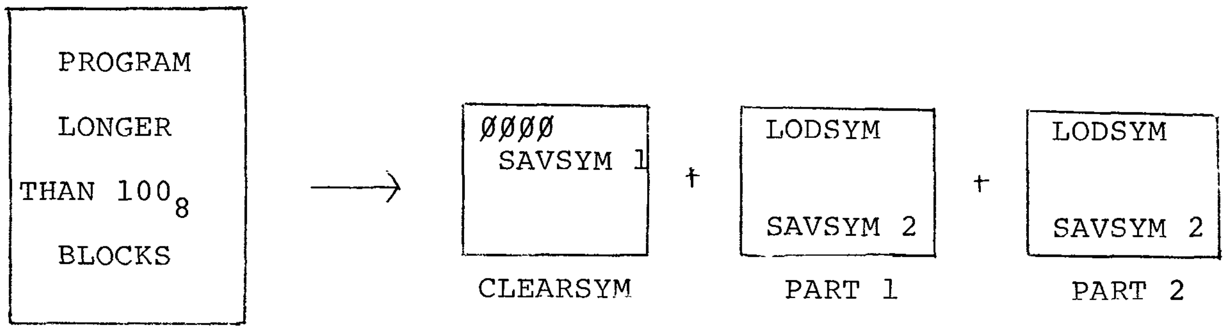


A SAVSYM 2 during the assembly of PART 1 will save all the symbols from PART 1. The LODSYM at the start of PART 2 loads in the saved symbol table, thus putting all the symbols from PART 1 and PART 2 into a common symbol table.

The program is then assembled by the commands

```
→ AS PART 1
→ AS PART 2
```

This method will only work if all the symbols referenced in PART 2 are defined in PART 1. Because most programs cannot easily be split so all references occur after definitions, the following technique is employed.



The program is assembled by the following command string.

```
→ AS CLEARSYM
→ AS PART 2
→ AS PART 1
→ AS PART 2
```

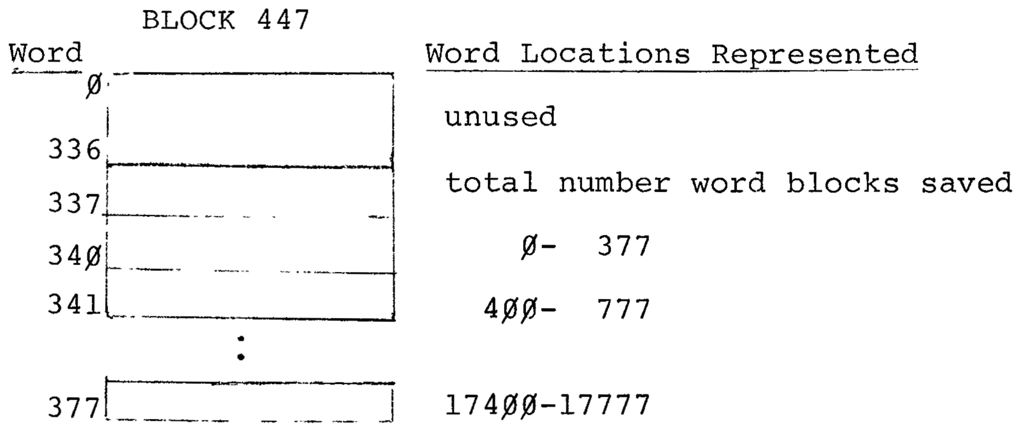
The program named CLEARSYM produces a clean symbol table. The LODSYM in PART 1 then loads in this clean symbol table. PART 2 is assembled first so that all symbols in that part will be defined when PART 1 is assembled. Ignore any error messages generated at this time. PART 1 and PART 2 are then assembled correctly by using a SAVSYM 2 pseudo-op at the end of each part and a LODSYM at the start of each. Any error messages generated now indicate real errors in the program.

When starting an assembly, the Assembler destroys the binary coding in locations 4000-4377. Therefore, the user must assemble the part containing this section of code last. If the program was split so that locations 4000-4377 are in PART 2, then the sequence above will assemble the program correctly. If the program was split so that locations 4000-4377 are in PART 1, then the correct sequence of commands is as follows.

```
→AS CLEARSYM
→AS PART 2
→AS PART 1
→AS PART 2
→AS PART 1
```

Locations 4000-4377 must be in the last part assembled, no matter into how many parts the program has been split.

The program may then be loaded directly, but cannot be saved by a Save Binary command directly because the binary header block for the file is incorrect. Only the last part of the program to be assembled is included in the header block. Block 447, the header block, must be filled with correct information. Each word of block 447 from word 340 to 377 represents 400_8 words of the file to be saved, as follows.



If any of the word locations represented contain data, then the corresponding word of block 447 must contain the value 7777. If those word locations are empty, then the appropriate word of block 447 contains 0000.

To correct the header block, load 7777 into the correct word of block 447. When completed, change word 337 of block 447 to the total number of words between 340 and 377 in block 447 that contain the value 7777. A Save Binary command may then be performed.

C.3 CHARACTER SET

Keyboard	External (ASCII)	Internal
A	301	1
B	302	2
C	303	3
D	304	4
E	305	5
F	306	6
G	307	7
H	310	10
I	311	11
J	312	12
K	313	13
L	314	14
M	315	15
N	316	16
O	317	17
P	320	20
Q	321	21
R	322	22
S	232	23
T	324	24
U	325	25
V	326	26
W	327	27
X	330	30
Y	331	31
Z	332	32
[(SHIFT/K)	333	33
\ (SHIFT/L)	334	34
] (SHIFT/M)	335	35
↑	336	36
←	337	Illegal (not displayed)
SPACE	240	40
!	241	41
"	242	42
#	243	Illegal (not displayed)
$	244	44
%	245	45
&	246	46
'	247	Illegal (not displayed)
(	250	50
)	251	51
*	252	52

C.3 Character Set (Cont)

Keyboard	External (ASCII)	Internal
+	253	53
,	254	54
-	255	55
.	256	56
/	257	57
Ø	260	60
1	261	61
2	262	62
3	263	63
4	264	64
5	265	65
6	266	66
7	267	67
8	270	70
9	271	71
:	272	72
;	273	73
<	274	74
=	275	75
>	276	76
?	277	77
@	300	Illegal (not displayed)
LINE FEED	212	37
RETURN	215	43 (not displayed)
ALTMODE	375	None " "
RUBOUT	377	None " "
CONTROL/I (TAB)	211	47 " "

C.4 INSTRUCTIONS

PDP-8 SYMBOLS

Mnemonic	Octal	Operation
	MEMORY REFERENCE INSTRUCTIONS	
AND	0000	logical AND
TAD	1000	2s complement add
ISZ	2000	increment & skip if zero
DCA	3000	deposit & clear AC
JMS	4000	jump to subroutine
JMP	5000	jump
	GROUP 1 OPERATE MICROINSTRUCTIONS	
NOP	7000	no operation
IAC	7001	increment AC
RAL	7004	rotate AC & link left one
RTL	7006	rotate AC & link left two
RAR	7010	rotate AC & link right one
RTR	7012	rotate AC & link right two
CML	7020	complement link
CMA	7040	complement AC
CLL	7100	clear link
CLA	7200	clear AC
	GROUP 2 OPERATE MICROINSTRUCTIONS	
HLT	7402	halts the computer
OSR	7404	inclusive OR switch register with AC
SKP	7410	skip unconditionally
SNL	7420	skip on nonzero link
SZL	7430	skip on zero link
SZA	7440	skip on zero AC
SNA	7450	skip on nonzero AC
SMA	7500	skip on minus AC
SPA	7510	skip on plus AC (zero is positive)
	COMBINED OPERATE MICROINSTRUCTIONS	
CIA	7041	complement & increment AC
STL	7120	set link to 1
GLK	7204	get link (put link in AC, bit 11)
STA	7240	set AC = -1
LAS	7604	load AC with switch register

Mnemonic	Octal	Operation
	IOT MICROINSTRUCTIONS	
Program Interrupt		
ION	6001	turn interrupt on
IOF	6002	turn interrupt off
Keyboard/Reader		
KSF	6031	skip if keyboard/reader flag = 1
KCC	6032	clear AC & keyboard/reader flag
KRS	6034	read keyboard/reader buffer
KRB	6036	clear AC & read keyboard buffer, & clear keyboard flag
Teleprinter/Punch		
TSF	6041	skip if teleprinter/punch flag = 1
TCF	6042	clear teleprinter/punch flag
TPC	6044	load teleprinter/punch buffer, select & print
TLS	6046	load teleprinter/punch buffer, select & print, and clear teleprinter/punch flag
Clock		
CLSK	6131	skip on clock interrupt
CLLR	6132	load clock control register 1
CLAB	6133	AC to buffer preset register
CLEN	6134	load clock control register
CLSA	6135	clock status to AC
CLBA	6136	buffer preset register to AC
CLCA	6137	counter to AC
Extended Memory (Type MC8/I)		
CDF	62n1	change to data field n
CIF	62n2	change to instruction field n
RDF	6214	read data field into AC
RIF	6224	read instruction field into AC
RMF	6244	restore memory field
RIB	6234	restore instruction field
Processor Mode Change		
LINC	6141	change to LINC mode processing

LINC SYMBOLS

Mnemonic	Octal	Operation
	ADD	
ADD	2000	add memory to A (full address)
ADA	1100	add memory to A (index class)
ADM	1140	add A to memory (sum also in A)
LAM	1200	add link and A to memory (sum also in A)
	MULTIPLY	
MUL	1240	signed multiply
	LOAD	
LDA	1000	load A, full register
LDH	1300	load A, half register
	STORE	
STC	4000	store and clear A (full address)
STA	1040	store A (index class)
STH	1340	store half A
	SHIFT/ROTATE	
ROL N	240	rotate left N places
ROR N	300	rotate right N places
SCR N	340	scale right N places
	OPERATE	
HLT	0	Halt
NOP	16	no operation
CLR	11	clear A and LINC
SET	40	set register N to contents of register Y
JMP	6000	jump to register Y
QAC	5	MQ transfer to A
	LOGICAL OPERATIONS	
BCL	1540	bit clear (any combination of 12-bits)
BSE	1600	bit set (any combination of 12-bits)
BCO	1640	bit complement (any combination of 12 bits)
COM	17	complement A

LINC Symbols (Cont)

Mnemonic	Octal	Operation
	SKIP	
		Skip next instruction if:
SAE	1440	A equals memory register Y
SHD	1400	right half A unequal to specified half of memory register Y
SNS N	0440+N	SENSE switch N is set
SKP	0467	unconditional skip
AZE	0450	A equals 0000 or 7777
APO	0451	A contains positive number
LZE	0452	link bit equals 0
IBZ	0453	between blocks on LINC tape
FLO	0454	add overflow is set
QLZ	0455	bit 11 of Z register equals Ø
SXL N	0400+N	external level N is preset
KST	0415	keyboard has been struck
SRO	1500	rotate memory register right one place; then if bit 0 of Y equals 0, skip next instruction
XSK	0200	contents of Y equal 1777; index memory register if I bit set
STD	0416	tape instruction completed
	INPUT/OUTPUT	
ATR	0014	A to relay buffer
RTA	0015	relay buffer to A
SAM N	0100+N	sample analog channel N
DIS	0140	display point on oscilloscope
DSC	1740	display character on oscilloscope (2 x 6 matrix)
PDP	0002	change to PDP-8 mode
RSW	0516	RIGHT SWITCH register to A
LSW	0517	LEFT SWITCH register to A
IOB	0500	I/O bus enable
	MEMORY	
LIF	0600	change instruction field
LDF	0640	change data field

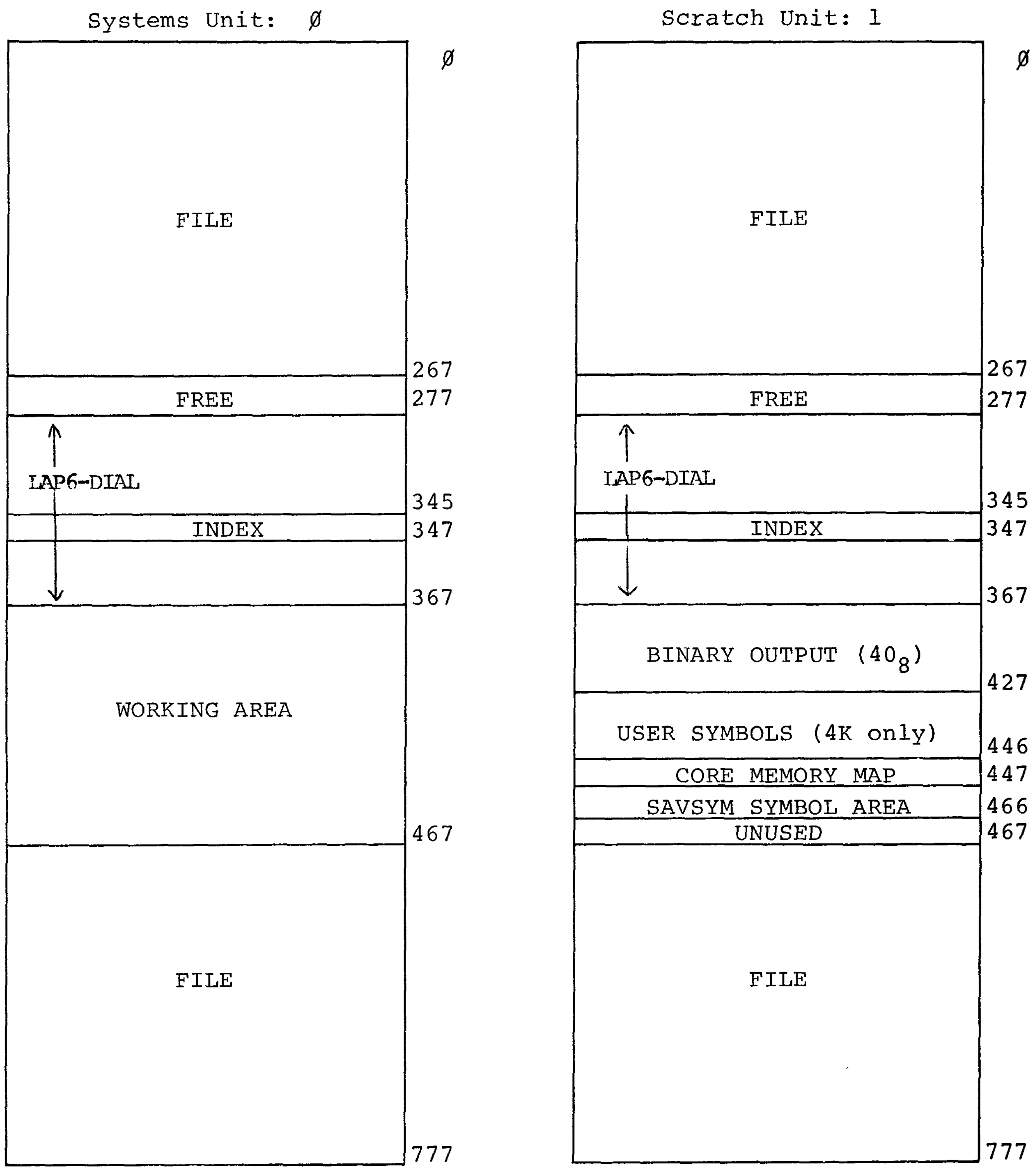
LINC Symbols (Cont)

Mnemonic	Octal	Operation
	LINC TAPE	
RDE	0702	read one block into memory
RDC	0700	read and check one block
RCG	0701	read and check N consecutive
WRI	0706	write one block on tape
WRC	0704	write and check one block
WCG	0705	write and check N blocks
CHK	0707	check one block of tape
MTB	0703	move tape toward selected block
	EXTENDED OPERATIONS	
ESF	0004	enable special functions
TAC	0003	tape control register to A
TMA	0023	A to tape control register
AXO	0001	A to extended operations buffer
DJR	0006	disable Jump Return Save
MSC	0000	miscellaneous
SFA	0024	special functions to A

C.5 OPERATORS AND SPECIAL CHARACTERS

Char	Mode	Operation
,	8/L	Assign symbolic address
*	8/L	Origin - dependent on mode (LINC or PDP-8)
=	8/L	Define parameters
+	8/L	Combine symbols or numbers
-	8/L	Combine symbols or numbers
.	8/L	Has value of current location counter
/	8/L	Comment
U	L	Add 10_8 to instruction
I	L	Add 20_8 to instruction
	8	Add 400_8 to instruction
;	8/L	Terminate coding line
SPACE	8/L	IOR
&	8/L	Logical AND
!	8/L	Logical IOR
\	L	Operator $x\backslash y= 1000_8x+y$

C.6 LAP6-DIAL TAPE ALLOCATION DURING ASSEMBLY



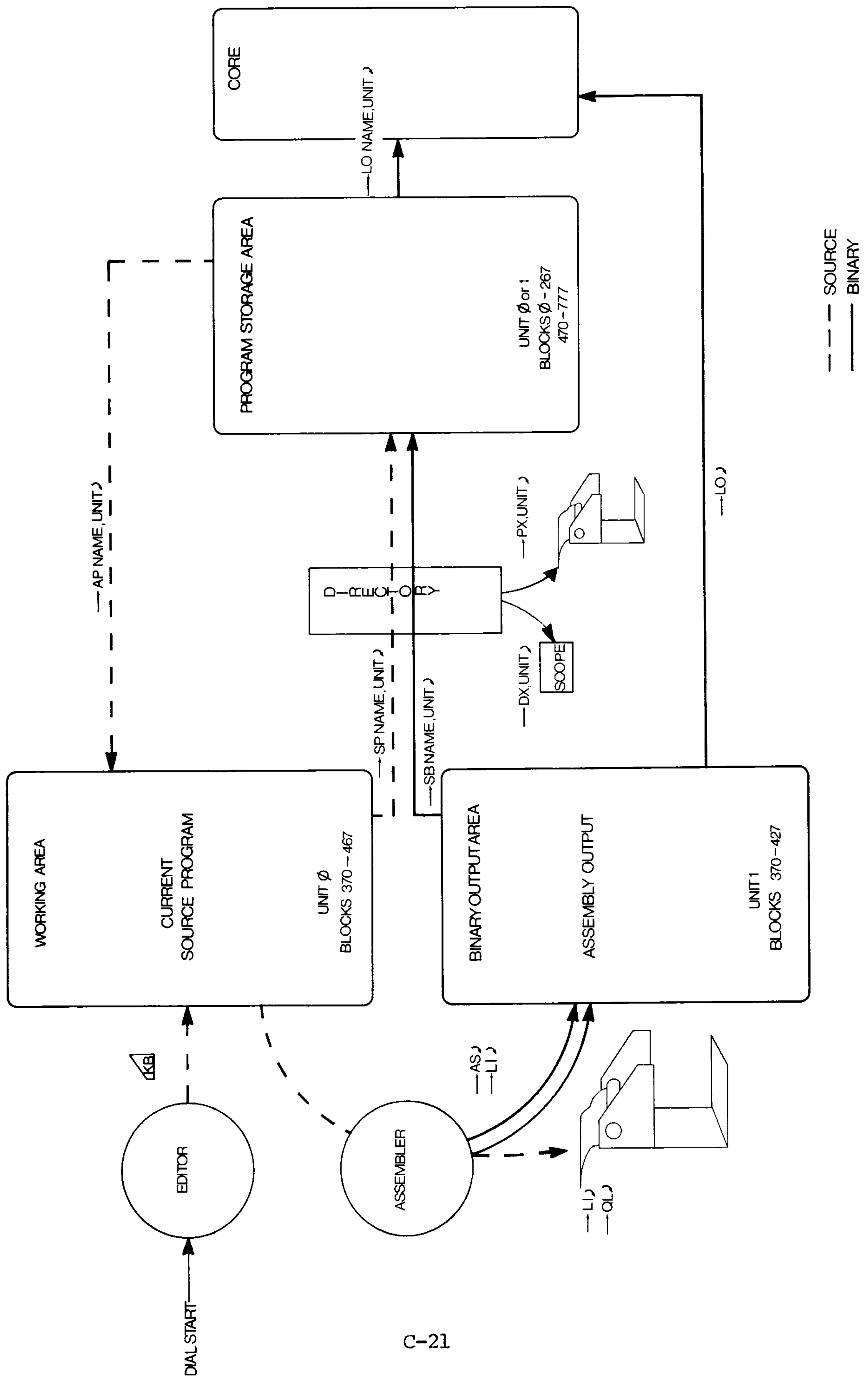
C.7 SAMPLE PROGRAM

The following listing is an example of a program which will read in a paper tape in image mode. (PIP will read in only source or binary). This program reads in 8 bits from the Teletype and puts it in one word, right justified, and packed with zeroes. The data is then written out onto the Working Area of unit 1. Sense switch Ø is used to indicate when the end of the tape has been reached and then to restart DIAL. Once the data is written on tape, it can be copied and used as input to another user program. Remember that this program is an example and is not the only way to do this task.

```
                *2Ø
                /
                /       THIS PROGRAM WILL READ IN A PAPER TAPE
                /       AND PLACE THE CONTENTS OF IT ON LINC TAPE.
                /       THE PAPER TAPE IS READ AS AN 8 BIT CODE
                /       AND IS PACKED ONE CHARACTER PER WORD,
                /       RIGHT JUSTIFIED, WITH THE LEFT BITS
                /       CONTAINING ZERO.
                /       THE DATA IS WRITTEN ONTO THE WORKING
                /       AREA OF UNIT 1, STARTING AT BLOCK 37Ø.
                /       SENSE SWITCH Ø IS THE END OF INPUT SIGNAL.
                /       IF THERE IS TOO MUCH DATA, THE PROGRAM
                /       WILL HALT WITH 7777 IN THE AC.
                /       WHEN FINISHED, THE PROGRAM WILL RESTART DIAL.
                /
                /
                        *1
                /
                /
ØØØ1  ØØØØ      BLOCKN, Ø               /CURRENT BLOCK NUMBER
                                        /BEING WRITTEN  ON.
ØØØ2  ØØØØ      BLOCKC, Ø               /NUMBER OF BLOCKS LEFT
                                        /IN THE WORKING AREA.
ØØØ3  ØØØØ      POINT,  Ø               /POINTS TO NEXT FREE
                                        /SPACE IN BUFFER.
ØØØ4  ØØØØ      BCOUNT, Ø               /NUMBER OF FREE WORDS
                                        /LEFT IN BUFFER.
                /
                /
                /         THIS IS THE BOOTSTRAP TO RESTART DIAL.
                /
                          *15
ØØ15    Ø643    DIAL,     LDF      3         /SET DATA FIELD TO 3
ØØ16    Ø7Ø1              RCG                /READ IN DIAL
ØØ17    Ø3ØØ              3ØØ                /8 BLOCKS FROM BLOCK 3ØØ
                /
                /
                /         DIAL WILL START HERE.
                /
                          *2Ø
ØØ2Ø    1Ø2Ø    START,    LDA I
ØØ21    ØØ2Ø              2Ø                 /I-O PRESET CODE
ØØ22    ØØØ4              ESF                /DO I-O PRESET
ØØ23    ØØ61              SET I    BLOCKN    /INITIALIZE POINTER
ØØ24    Ø367              WKAREA-1           /TO WORKING AREA-1
ØØ25    ØØ62              SET I    BLOCKC    /SET COUNTER TO WORKING
ØØ26    7676              -WKSIZE-1          /AREA SIZE+1 TO ALLOW
                                             /FOR RESET
ØØ27    6Ø56    LOOP,     JMP      RESET     /START OF A NEW BLOCK
ØØ3Ø    Ø46Ø    INLOOP,   SNS I    Ø         /SENSE SWITCH Ø NOT UP?
ØØ31    6Ø42              JMP      DUMP      /IT IS UP. DUMP BUFFER
ØØ32    Ø5ØØ              IOB
                          PMODE              /DO PDP-8 IOT
4Ø33    6Ø31              KSF                /IS A CHARACTER READY?
                          LMODE
ØØ34    6Ø3Ø              JMP      INLOOP    /NO CHAR THERE. WAIT
ØØ35    Ø5ØØ              IOB                /CHAR THERE
                          PMODE
4Ø36    6Ø36              KRB                /READ IN THE CHAR
                          LMODE
ØØ37    1Ø63              STA I    POINT     /PLACE IN BUFFER
ØØ4Ø    Ø224              XSK I    BCOUNT    /BUFFER FULL?
ØØ41    6Ø3Ø              JMP      INLOOP    /NO. GET ANOTHER WORD.
ØØ42    1ØØØ    DUMP,     LDA                /YES. WRITE OUT BUFFER
ØØ43    ØØØ1              BLOCKN
ØØ44    156Ø              BCL I              /ONLY 9 BITS OF INTEREST
ØØ45    7ØØØ              -777               /AND WITH 777
ØØ46    162Ø              BSE I              /SET  CORRECT FIELD BITS
                          PMODE              /FOR 12 BIT CALCULATION
4Ø47    1ØØØ              BUFFER+4ØØ&3ØØØ /SETS CORRECT BITS ON
                          LMODE
ØØ5Ø    4Ø52              STC      .+2       /GET BLOCKN NUMBER AND PLACE
                                             /IN WRITE INSTRUCTION
ØØ51    Ø714              WRC      U         /WRITE OUT ON UNIT 1
ØØ52    ØØØØ              ØØØØ               /BLOCK NUMBER GOES HERE
ØØ53    Ø44Ø              SNS      Ø         /SENSE SWITCH Ø UP?
ØØ54    6Ø27              JMP      LOOP      /GO BACK AND GET SOME MORE
ØØ55    6Ø15              JMP      DIAL      /YES. LOAD IN DIAL.
                /
```

```
                          /
      ØØ56  Ø221  RESET,  XSK I   BLOCKN  /INCREMENT BLOCK NUMBER
      ØØ57  ØØ63          SET I   POINT   /SET BUFFER POINTER
      ØØ6Ø  4377          BUFFER-1
      ØØ61  ØØ64          SET I   BCOUNT  /SET BUFFER WORD COUNTER NOW
      ØØ62  7377          -4ØØ            /4ØØ OCTAL=256 DECIMAL
      ØØ63  Ø222          XSK I   BLOCKC  /TOO MUCH DATA?
      ØØ64  6ØØØ          JMP     Ø       /NO. RETURN
      ØØ65  1Ø2Ø          LDA I
      ØØ66  7777          7777
      ØØ67  ØØØØ          HLT             /YES. HALT WITH 7777 IN AC
      ØØ7Ø  6Ø65          JMP     .-3     /ONLY A RESTART  PERMITTED NOW
                          /
                          /
                          /
                          WKSIZE=1ØØ      /WORKING AREA SIZE IS
                                          /1ØØ BLOCKS
                          WKAREA=37Ø      /WORKING AREA STARTS FROM
                                          /BLOCK 37Ø
                          BUFFER=44ØØ     /TAPE BUFFER AREA
                          /
                          /


ØØØØ  ERRORS




BCOUNT 4ØØ4
BLOCKC 4ØØ2
BLOCKN 4ØØ1
BUFFER 44ØØ
DIAL   4Ø15
DUMP   4Ø42
INLOOP 4Ø3Ø
LOOP   4Ø27
POINT  4ØØ3
RESET  4Ø56
START  4Ø2Ø
WKAREA Ø37Ø
WKSIZE Ø1ØØ
```

C.8 WORK FLOW IN THE DIAL SYSTEM



INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

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

Digital Software News for the PDP-8 Family
Digital Software News for the PDP-9/15 Family
PDP-6/PDP-10 Software Bulletin

These newsletters contain information applicable to software available from Digital's Program Library.

Please complete the card below to place your name on the newsletter mailing list.

Questions or problems concerning DEC Software should be reported to the Software Specialist at your nearest DEC regional or district sales office. In cases where no Software Specialist is available, please send a Software Trouble Report form with details of the problem to:

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

These forms, which are available without charge from the Program Library, should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

New and revised software and manuals, Software Trouble Report forms, and cumulative Software Manual Updates are available from the Program Library. When ordering, include the document number and a brief description of the program or manual requested. Revisions of programs and documents will be announced in the newsletters and a price list will be included twice yearly. Direct all inquiries and requests to:

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

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

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

Send Digital's software newsletters to:

Name ______________________
Company Name ______________________
Address ______________________
____
(zip code)

My computer is a	PDP-8/I ☐	PDP-8/L ☐	
	LINC-8 ☐	PDP-12 ☐	
	PDP-9 ☐	PDP-15 ☐	Please specify
	PDP-10 ☐	OTHER ☐	______________

My system serial number is ______________ (if known)

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

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

Postage will be paid by:

digital

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

READER'S COMMENTS DIAL PROGRAMMER'S REFERENCE MANUAL DEC-12-SE2A-D

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

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

Did you find errors in this manual?

How can this manual be improved?

DEC also strives to keep its customers informed of current DEC software and publications. Thus, the following periodically distributed publications are available upon request. Please check the appropriate boxes for a current issue of the publication(s) desired.

- ☐ Software Manual Update, a quarterly collection of revisions to current software manuals.
- ☐ User's Bookshelf, a bibliography of current software manuals.
- ☐ Program Library Price List, a list of currently available software programs and manuals.

Please describe your position. ____________

Name ____________ Organization ____________

Street ____________ Department ____________

City ____________ State ____________ Zip or Country ____________

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS

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

Postage will be paid by:

digital

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

Digital Equipment Corporation
Maynard, Massachusetts

digital

printed in U.S.A.