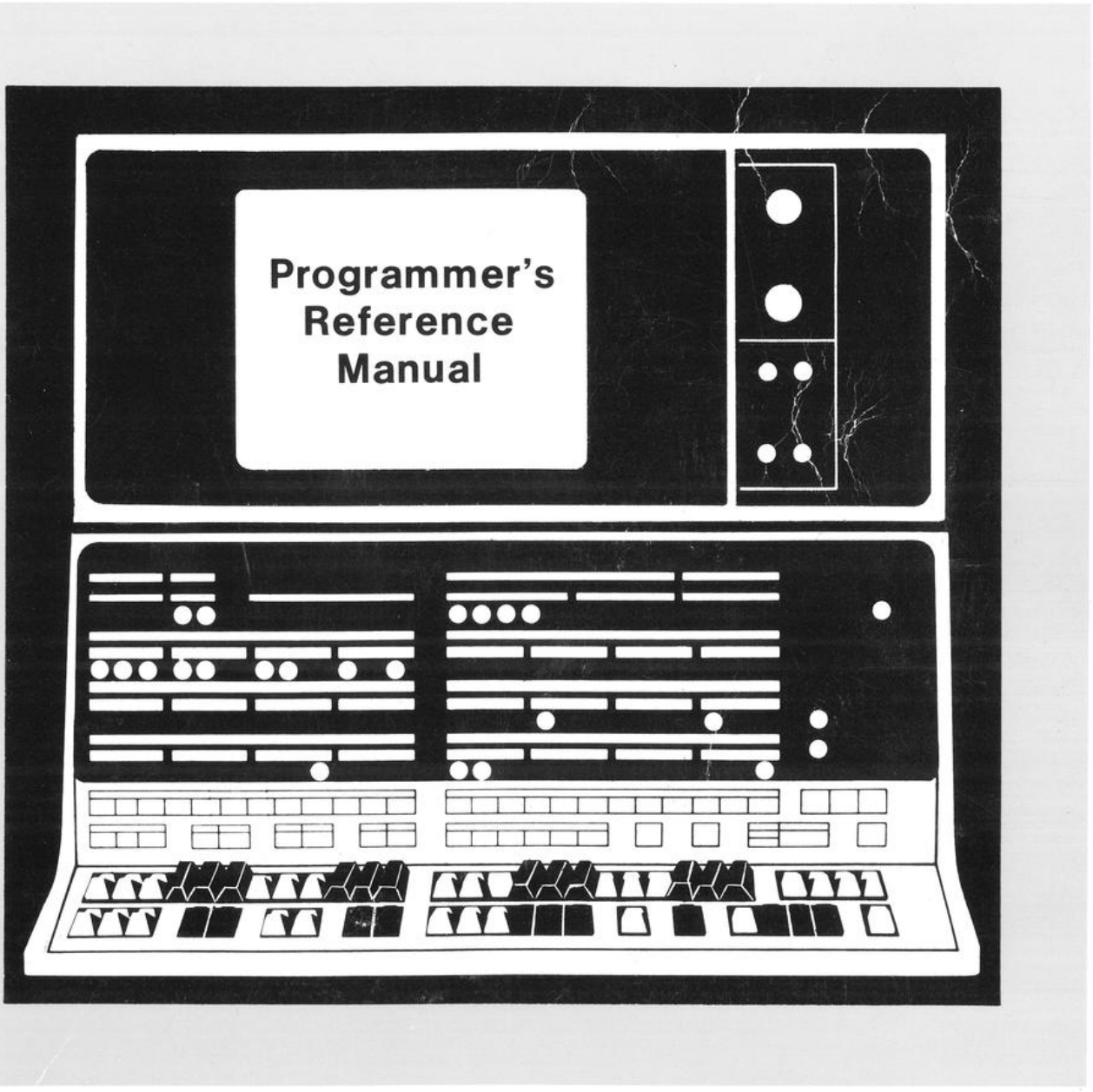
Digital Equipment Corporation
Maynard, Massachusetts

digital

PDP-12

LAP6-DIAL



PDP-12
LAP6-DIAL
PROGRAMMER'S REFERENCE MANUAL

For additional copies of this document, order No. DEC-12-SE2D-D from Program Library
Digital Equipment Corporation, Maynard, Mass. 01754 Price: $2.00

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Edition February 1970
2nd Printing (Rev) August 1970

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 (DEC) 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

CONTENTS (Cont)

APPENDICES (Cont)

ATTENTION

The LAP6-DIAL[1] tapes distributed by the Digital Equipment Corporation Program Library contain two versions of DIAL: DIAL-V2, for 4K tape systems, and DIAL-MS, for 8K and larger systems, particularly those using disks. Users may decide to use one or both systems. See Appendix A.1 for differences between the two systems. The system in the system area (blocks 300-345, 350-367) of the distributed tapes is DIAL-V2. Therefore, any user who wants this system may use it immediately via the startup procedure in Appendix A.4. DIAL-MS is distributed as four binary files, with a program (GENASYS) for combining them into a DIAL-MS system. Appendix A.2 describes the procedure for initially building a DIAL-MS system on tape. Appendix A.3 describes the DIAL-MS initialization procedure, which must be performed on a DIAL-MS tape whenever:

a. The tape is used with a new configuration
b. The disk is overwritten
c. A non-DIAL disk is in the drive (RK08 only).

Note that the GENASYS procedure must always be followed by the initialization procedure. GENASYS will allow the user to perform the initialization automatically if the DIAL-MS tape built by GENASYS is on unit 0.

[1] LAP6-DIAL is referred to as DIAL in this manual.

Chapter 1
Using DIAL

1.1 INTRODUCTION

DIAL 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-V2 is a PDP-12B system, composed of:

a. 4096-word, 12-bit, 1.6 usec core memory
b. TC12 LINCtape automatic control
c. Two TU55 magnetic tape transports
d. VC12 LINCscope control and character buffer
e. VR12 7 in. x 9 in. CRT display (scope)
f. ASR33 Teletype®

Note that the DIAL character Editor is designed to be operated with an AD12 Analog-to-Digital Converter and Multiplexer.

DIAL-MS requires 8K of core memory and will support the additional hardware:

a. One or two DF32 disks (32K to 64K words)
b. One to four RS08 disks (256K to 1M words) and RF08 controller
c. One RK08 disk (800K words)

The PDP-12 includes a display for viewing source programs in the source working area[1]. About 400 characters can be displayed at a time on the scope.

The DIAL system is provided to the user on LINCtape[2]. Each tape contains:

a. A reserved area occupied by DIAL
b. A working area for temporary storage of user files
c. A file area for permanent storage of user files

---

[1] The source working area may be on tape or on disk (refer to Appendix C, Paragraph C.6).

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

®Teletype is a registered trademark of Teletype Corporation.

The DIAL area of the tape contains the DIAL Editor, Assembler, command routines, and a file index. User programs are saved as named files in the file area of the system tape(s) and/or disk(s). DIAL tape and disk allocation is detailed in Appendix C, Paragraph C.6.

1.2 SYSTEM OPERATION

A LINCtape containing DIAL must be designated as the system tape and assigned to tape transport 0. Most DIAL operations may be performed with only one LINCtape containing DIAL, but many procedures in DIAL-V2, such as assembling programs, require another tape on unit 1. Most efficient operation is achieved when both tapes contain DIAL systems.

When the system is started (refer to Appendix A, Paragraph A.3), it automatically enters the text mode. A source program may then be typed into the source working area character by character via the Teletype keyboard and will be displayed as it is entered. DIAL automatically assigns lines numbers and formats the input. The DIAL Editor may be used to add, modify, or delete characters, lines or large sections of the program, as detailed in Chapter 2.

A Monitor command may also be issued to a system program via the Teletype keyboard. 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 pointers.

The DIAL Assembler is implemented by three Monitor commands that provide user control of the type of listing generated. The Peripheral Interchange Program, PIP, permits any file to be moved between interfaced I/O devices. The rest of this chapter details basic concepts and terminology of the DIAL system[1].

1.3 SOURCE PROGRAMS

A DIAL source program is a group of lines of program input via the Teletype in a symbolic language composed of PDP-8 and LINC mode instructions (refer to Appendix C, Paragraph C.4). Using DIAL, the program can be typed into the source working area and altered with the Editor, stored in a named file, displayed on the scope, or printed on the Teletype. A line number ($1\text{-}777_8$) appears to the left of each line on the scope and indicates the sequential location of that line in the source.

The program text is placed in an input buffer in core; the buffer is 256 words long and can accommodate 512 ASCII characters. Up to 64 blocks of source program can be input to DIAL. As the input buffer is filled, the text is written out in the source working area which starts at block 370 (see tape 0 in Appendix C, Paragraph C.6). Only the program currently in the working area can be edited. If source program input exceeds 64 blocks

---

[1] Refer to Introduction to Programming, DEC C-18, for detailed information on computer fundamentals, such as decimal-to-octal conversion.

or 4094 lines, it can not be assembled; 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, Paragraph C.2.

1.4 BINARY PROGRAMS

A binary program is the result of an assembly of a source program. The Assembler converts the symbolic source language into binary machine instructions. The binary program is stored in the binary working area and can be saved as a named file (refer to section 4.4) or loaded for immediate execution (refer to Section 4.5).

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 (two 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 source 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 which maintains an approximate constant number of lines.

1.6 LINE CALLS

By issuing a line call when using the DIAL Editor, the designated line in the source program is displayed as the last line of text on the scope and a new line number will appear as the current line. To display a new current line, use either of the following methods:

a. Type → L ↵ where L is the number of the line to be the new current line. The right arrow → indicates pressing the LINE FEED key and ↵ means press the RETURN key. The display will now be positioned with line L as the last line displayed.

b. Type ALTMODE and then one of the following keys:

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

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

These ALTMODE key combinations must be typed as the first characters on a new line. A frame is defined as the number of lines currently on the display, as set by A/D knob 7.

The last line of the source program in the working area can be displayed by requesting a line number greater than the last line number. Similarly, the first line of the source program can be displayed by calling line 0 or line 7777.

1.7 FILES

The DIAL system is file oriented. A program, either source or binary, is saved as a file in contiguous blocks of tape (disk) in the file area, blocks 0 through 267 and 470 through 777. Binary files require a header block for pointers. Every tape contains a file index in blocks 346 and 347 for the binary and/or source programs on that tape. (Refer to Appendix C, Paragraph C.6 for tape and disk allocation.) File names, starting block, and length in blocks are recorded in the index. When a file is entered, the user gives it a name which must be 1 to 8 displayable keyboard characters in length, of which at least one character 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. A full index can accommodate 63 different names; however, any name in the index can describe both a source and a binary program, thereby doubling the number of possible file entries to 126.

When a file is being saved, the unused file space nearest the index within the file 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 and 347 during file assignment.

1.8 MONITOR COMMANDS

The DIAL programs are requested through DIAL Monitor commands and cause DIAL to enter command mode. To issue a command, use the following procedure:

1. Turn A/D knob 3 all the way to the right.
2. Press the LINE FEED key on the Teletype and observe the right arrow in the lower left corner of the scope.
3. Type the command (refer to Chapters 4 and 5).
4. Press the RETURN key.

Improperly formatted or spelled commands are ignored and are automatically erased from the scope. The

Monitor commands, including PIP, are summarized in this section and are treated in greater detail in Chapters 4 and 5. In this manual, a reference to tape unit 0 means the eighth channel on TU55 LINCtape transports. Items in parentheses in the table are optional; if NAME is omitted, the user's program that was most recently manipulated is used.

The unit number required in DIAL commands can be between 0 and 7 in DIAL-V2 and between 0 and 17 in DIAL-MS, as follows:

	Device	Acceptable Unit Numbers	Logical Disk Units
	8 LINCtapes	0 - 7	0 - 1
DIAL MS only:			
	1 RS08 disk	10 - 11	0 - 1
	2 RS08 disks	10 - 13	0 - 3
	3 RS08 disks	10 - 15	0 - 5
	4 RS08 disks	10 - 17	0 - 7
	1 RK08 disk	10 - 15	0 - 5

Each RS08 or RK08 disk is considered to be broken into smaller logical disks, each with its own directory. A logical unit is the equivalent of one LINCtape, 1000_8 blocks; thus, one RS08 disk is considered to be made up of two logical disk units. When issuing a DIAL command, the logical disk unit is addressed by an acceptable unit number. Note that DF32 disks cannot be addressed; they are used only to hold DIAL-MS and the working areas. If no unit is specified, unit 0 is assumed.

Command	Function
→ AS (N,U)	Assemble
→ LI (L,L,)(N,U)	Assemble and List
→ QL (L,L,)(N,U)	Assemble and Quick List
→ LO (N,U)	Load Binary
→ PS (L,)(L,)(N,U)	Print Source
→ AB (A,)N,U	Add Binary
→ SB N,U(,M(FA))	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 Source Working Area
→ ZE	Zero Binary Working Area
→ PI	Peripheral Interchange
→ EX	Exit

Command (cont)	Function (cont)
→ MC X(Y),U	User's Monitor Command
→ L	Line Call

Legend:

N	=	File name
U	=	Tape (0-7) or disk (10-17) unit
L	=	Line number
M	=	Start mode (L for LINC or P for PDP-8)
A	=	Address (4 digits)
F	=	Field
B	=	Block number
X(Y)	=	Character in Accumulator
( )	=	Optional parameter
→	=	Press LINE FEED key
↵	=	Press RETURN key

1.9 ERROR RECOVERY

If an illegal command is requested at any time while using DIAL, NO is displayed on the scope. Press RETURN to return to DIAL. No data is lost and the correct procedure may be requested. If the error occurs while using PIP (refer to Chapter 5), press RETURN to generate the initial PIP display.

If at any time while using DIAL a typing error occurs before RETURN has been pressed to terminate the line, the RUBOUT key may be used to delete the incorrect input. Press the RUBOUT key once for each character to be deleted from the left. Each time RUBOUT is typed, a backslash (\) is echoed on the Teletype.

Chapter 2
The Editor

2.1 TEXT AND COMMAND MODES

After starting DIAL (refer to Appendix A), a portion of the source working area is displayed on the scope. DIAL is said to be in text mode at this time. Issuing a Clear command at this time will remove any previous data from the source working area. Data may then be typed in the cleared working area when the Editor is in the text mode. At the conclusion of an editing operation, the command EXIT should be issued so that the Editor will save its pointers on the tape or disk. If a command is issued (refer to Section 1.8), the Editor enters the command mode. Both modes are under control of the Editor.

2.2 USING THE EDITING CURSOR

DIAL provides a powerful and flexible character Editor which is used in text mode to:

a. Add or delete a single character
b. Add or delete a text line
c. Delete the current line
d. Delete an entire portion of the display
e. Add text at any location in the program

Input is automatically formatted and assigned line numbers by the Editor.

The Editor is controlled by a cursor that appears as an inverted T (⊥) on the scope with the text. The cursor 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 location of the cursor is at the current character; all editing operations occur at the current character. 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 location has been determined by knob 3, it will move along with incoming text from the keyboard, always at that number of spaces from the end of the text. At any time, the location of the cursor may be changed by simply turning knob 3. Rotating the knob clockwise moves the cursor to the right. 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. For normal input of data, knob 3 is initially rotated all the way to the right.

For a PDP-12B system without an AD12 and multiplexer, the Right and Left Switches are used instead of A/D 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 Left Switches locates the cursor that many characters from the end of the text. (All discussion in this manual assumes a system configuration with an AD12 and multiplexer.)

Monitor commands, to enter the command mode, and line calls, to display a new current line, can be issued only when the cursor is located at the start of a current line that has no text on it yet. Turn knob 3 all the way to the right and press the RETURN key before issuing a Monitor command or a line call to locate the cursor correctly.

When a mistake has been noted in the previous text while in text mode, set the cursor to indicate where the correction is to be made and use the RUBOUT key. After the correction has been completed, rotate knob 3 all the way to the right; text input can be continued. If a typing error occurs while issuing a command, only the RUBOUT key is required to correct it.

To facilitate understanding the operation of the Editor, the procedures described in this chapter should be tried at the keyboard directly.

2.3 CHARACTER EDITING

A single text character may be deleted or added at any point in the displayed text during text mode. To delete a character, turn knob 3 to locate the cursor so that character is the current character, and press the RUBOUT key. The character will be removed, and the rest of the line and the cursor will move one character to the left. The current character is now the one that preceded the character that was deleted.

Note that spaces and carriage RETURNs are considered to be one character each. To delete a carriage RETURN, locate the cursor under the first space immediately after the last character on the line and press RUBOUT. As much of the following line as can fit on the line after the last character is moved up to that line on the scope.

In the following example, the letter, A, is deleted from line 1.

before

```
1       SCOPE.  ADSC 12
```

(press RUBOUT)

after

```
1       SCOPE.  DSC 12
```

In this example, a carriage return is deleted.

before

```
        CLR
        JMP  SETUP
SCOPE.  DSC  12
        CLR                 /CURSOR DISPLAY
        STC  CURFLG         /CLEAR FLAG
```

(press RUBOUT)

after

```
1
2
3       CLR
        JMP  SETUP
        SCOPE.  DSC  12  CLR                    /CURSOR
        DISPLAY
4               STC  CURFLG         /CLEAR FLAG
5
```

In the same manner, characters may be inserted in a line of text. Turn Knob 3 to locate the cursor under the character that is to immediately precede the first new character. As the additions are typed, they are echoed on the Teletype and scope.

The cursor moves one space to the right for each character that is inserted[1]. The characters 12 are added to line 1 below.

before

```
1       SCOPE, DSC
```

(type 12)

after

```
1       SCOPE, DSC 12
```

2.4 LINE INSERTION

To insert a line after Line L

1. issue a line call for Line L and insert the new line as the current line (L+1) or
2. position the cursor after the last character of Line L (i.e., the blank representing carriage RETURN), and type the new line, including its RETURN.

before

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

(type SET 12 ❯)

after

```
1               CLR
2               JMP  SETUP
3       SCOPE,  DSC  12
4               CLR                 /CURSOR DISPLAY
5               STC  CURFLG         /CLEAR FLAG
6               LDA  I
                -10
10              JMP  ENT
11
```

2.5 CURRENT LINE DELETION

The current line of a display is the last line preceded by a line number that is visible on the scope (refer to Section 1.5). 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 typing the keys ALT-MODE and D. The location of the cursor does not affect current line deletion. Only the current line is removed by this operation. The preceding line becomes the new current line. Line 5 of the source display is omitted in the following example.

[1] In any editing operation, the cursor moves according to the setting of A/D Knob 3. This means the cursor is always located the same number of spaces from the end of the text at any time in any procedure, if the knob is unchanged. Similarly, the number of lines specified by A/D Knob 7 will be displayed on the scope after any operation, assuming sufficient source in the Working Area.

before

```
 1                CLR
 2                JMP  SETUP
 3        SCOPE,  DSC  12
 4                CLR              /CURSOR DISPLAY
 5                STC  CURFLG      /CLEAR FLAG
 6                LDA  I
 7                -10
                  ▲
10                JMP  ENT
11
```

(type ALTMODE and D)

after

```
 1                CLR
 2                JMP  SETUP
 3        SCOPE,  DSC  12
 4                CLR              /CURSOR DISPLAY
 5                STC  CURFLG      /CLEAR FLAG
 6                LDA  I
                  ▲
 7                -10
10                                                  _
```

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 when the call is issued.

2.6 LARGE SECTION DELETION

To delete a large section of the displayed text, the Editor can eliminate all of the displayed text to the right or left of the cursor. By making the current character the last character of a string of text to be deleted to the left of that character and then typing the keys ALTMODE and L, the current character and all the characters to its left on the preceding lines visible on the display are deleted. In the following example, all characters on lines 1 through 4 are deleted.

before

```
 1                CLR
 2                JMP  SETUP
 3        SCOPE,  DSC  12
 4                CLR              /CURSOR DISPLAY
                                                  ▲
 5                STC  CURFLG      /CLEAR FLAG
 6                LDA  I
 7                -10
10                JMP  ENT
11                                                  _
```

(type ALTMODE and L)

after

```
 1                STC  CURFLG      /CLEAR FLAG
        ▲
 2                LDA  I
 3                -10
 4                JMP  ENT
 5                                                  _
```

To delete a large section of code that can not be displayed on the scope all at once use the following procedure:

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 that the last line of code to be deleted is the current line.
4. Type ALTMODE and L as many times as needed until the first line of the section of code to be deleted appears on the scope.
5. Type ALTMODE and D as needed to delete text current line by current line through the first line of code to be deleted.

Typing the keys ALTMODE and R performs a delete-right operation, removing the current character and all the characters to its right on succeeding lines of the display. The remaining text is redisplayed, and enough preceding lines are added above them on the display to satisfy the knob 7 displayed line number requirement.

In the following example, two and a half lines are deleted from the right of the cursor. (Remember that TAB and carriage RETURN count as one character each.)

before

```
 1                CLR
 2                JMP  SETUP
 3       SCOPE,   DSC  12
 4                CLR          /CURSOR DISPLAY
 5                STC  CURFLG  /CLEAR FLAG
 6                LDA  I
 7                -10
10                JMP  ENT
11
```

(type ALTMODE and R)

after

```
 1                CLR
 2                JMP  SETUP
 3       SCOPE,   DSC  12
 4                CLR          /CURSOR DISPLAY
 5                STC  CURFLG  /CLEAR FLAG
 6                LD
```

2.7 CURRENT LINE FORMATTING

When a new line is typed as the current line of the displayed text, it is automatically formatted by the Editor. Each text line is considered to be composed of three fields; each field in a line has a number of displayable spaces designated to it by DIAL. Of the 40 spaces available, 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, as follows:

```
   tag    /   instruction   /   comment
--------/----------------/----------------
```

A horizontal tab takes eight scope spaces, thus permitting five tabs per line. When a new line is encountered, the first characters are displayed in the instruction field, unless the first character is a slash. If a comma is then typed, the preceding characters are moved to the tag field, and subsequent input is displayed, starting at the instruction field. This operation is demonstrated with the input line SCOPE, DSC 12.

```
during              SCOPE
after     SCOPE,    DSC 12
```

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

	Displayed In Field		
Keys Typed	Tag	Instruction	Comments
/L	/L		
P/L		P	/L
P,/L	P,		/L
P TAB /L		P	/L

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

2.8 HANDLING LARGE PROGRAMS

The DIAL Editor can handle text files that are up to 64 blocks long. (Assuming an average of 16 characters per line, about 2048 lines of code can be accommodated.) 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 is greater than 64 blocks, the program can be edited in two or more sections and then combined during Assembly by using the assembly pseudo-ops SAVSYM, LODSYM, and LISTAPE (Refer to Appendix C.2).

If a line is entered when the source working area is full, the line is automatically deleted when carriage 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 (refer to Section 2.6) to remove one block from the program. (The LINCtape on unit 0 will move or the disk lights will blink when this has been accomplished.) Use the EXIT command (refer to Section 2.10), and then press the CONTinue key on the computer console to perform the required editing while maintaining the current source program.

The Editor does 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. Note that a maximum of 40 characters can be displayed on a scope line, so that a logical line of 120 characters occupies three scope lines.

2.9 OVERLAPPED I/O

When using DIAL-V2, the Editor writes on the core buffer on tape as they are being filled. 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. DIAL-MS

does not provide this overlap feature because writing is accomplished so quickly on the disk. Note, however, that if DIAL-MS is run using tape, the lack of overlapped I/O is noticeable.

2.10 USE OF EXIT COMMAND

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

2.11 LEFTMOST CURSOR POSITION

If the setting of the cursor, as determined by A/D knob 3, provides for more characters than presently appear 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 the actual A/D knob 3 setting. For example, if the cursor is set to ten characters from the end of the text, but only six 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 DIAL source program by translating PDP-8 and LINC mode mnemonic operation codes into binary codes for the corresponding instructions, relating symbols to their numeric values, assigning absolute core addresses for data and instructions, and listing the program with error messages.

This chapter discusses the DIAL syntax and semantics acceptable to the Assembler. Refer to Introduction to Programming, DEC C-18, for more detailed information on programming techniques.

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 pressing carriage RETURN. The Assembler interprets and processes these statements, generating one or more binary instructions or data words. DIAL statements can be typed without having to adhere to any column formatting.

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:

tag, operator, operand/comment

A statement must contain at least one of these fields and may contain all four.

3.1.1 Tags

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

3.1.2 Operators

An operator may be one of the mnemonic machine instruction codes (refer to Appendix C, Paragraph C.4), or pseudo-op codes which directs assembly processing (refer to Section 3.5). Operators are terminated with a space if an operand follows or with a semicolon, slash, or carriage return if no operand follows. Note that a semicolon anywhere in a statement, except in a comment, is interpreted by the DIAL-V2 Assembler as the logical end of the statement. It terminates the line at that location and in-

crements the program's line numbers by 1 line for each semicolon; this causes the line numbers on an Assembly listing to not agree with those seen on the scope. This incrementation allows several instructions to be written on one line as, for example, RTR; RTR; RTR. For DIAL-MS, semicolon does not increment the listing line numbers so that they agree with those on the scope.

3.1.3 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.

3.1.4 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 SYMBOLS

3.2.1 Types of Symbols

There are two main groups of symbols, as follows:

a. Permanent Symbols

The assembler has in its permanent symbol table definitions of its operation codes, operate commands, and many input-output transfer (IOT) micro-instructions (refer to Appendix C, Paragraph 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 the DIAL system unit. As the source program is processed, new symbols defined in the source program are added to the user's symbol table.

If the programming mode is to be changed to PDP-8 mode, the pseudo-op PMODE must precede the new program input to alert the Assembler[1]. This instructs the Assembler to retrieve the PDP-8 mode permanent symbols in memory. (The same core memory block is used for the permanent symbols of both modes.) Similarly, the pseudo-op LMODE must precede a change to LINC mode programming.

b. User-defined Symbols

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

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

(2) The first character must be alphabetic. Leading numeric characters are ignored.

---

[1] Distinguish between PMODE and the LINC mode instruction PDP which modifies the hardware at run time.

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

(4) The maximum number of symbols is 895.

(5) The Assembler assigns values according to the following table:

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

(See Section 3.6 for definition of LMODE and PMODE).

The programmer can use two methods to specify the value to be assigned to a symbol:

a. When the first symbol of a statement is terminated by a comma, it is assigned a value equal to the Current Location Counter (CLC). Any instruction or data word in the program may be so tagged.

For example:

```
            *1ØØ
TAG,        CLR
            JMP A
B,          Ø
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. (Refer to Appendix B.)

b. 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, the symbols ALPHA and BETA are assigned numeric values by

```
ALPHA=5
BETA=7
```

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.

---

[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. Note also that an address defined in LMODE is not the same when referenced in PMODE.

```
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. The statement

```
ALPHA=7
```

changes the value assigned to ALPHA in the first example from 5 to 7 for any future occurrence of ALPHA. 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.

3.2.2 Use of Symbols

Symbols are used in three ways.

a. To tag an instruction or data word at any point in the program, the symbol 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.

   If a symbol is to be used as an address in LINC mode, it should be assigned a 10-bit value.

b. As an operator, the symbol must be predefined by the Assembler or by the programmer. If a statement has no tag, the operator may appear first in the statement, and must be terminated by a space, tab, semicolon, or carriage return. Examples of operators follow.

TAD	permanent symbol
OCAL	Assembler pseudo-op
ZIP	legal only if defined by the user

c. Symbols used as operands should have a value defined by the user. This value may be symbolic references to previously defined labels where the arguments to be used by this instruction are to be found, or 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, DECIMAL and OCTAL, indicate to the Assembler the radix to be used in number interpretation. DECIMAL indicates that all numbers are to be interpreted as decimal until the next occurrence of the pseudo-op OCTAL, which 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 operations are:

+	Plus	2's complement addition (modulo 4096) after a PMODE pseudo-op. 1's complement addition (modulo 1024) after an LMODE pseudo-op.
-	Minus	2's complement subtraction (modulo 4096) after a PMODE pseudo-op. 1's complement subtraction (modulo 1024) after an LMODE pseudo-op.
!	Exclamation Mark	Boolean inclusive OR (union).
&	Ampersand	Boolean AND (intersection).
␣	Space	Inclusive OR when used to separate two symbolic operators. For example: TAG, CLA CLL

Note that there should be no spaces between operands and the 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 lo-

cation 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 CLC. It may be used as any integer or symbol (except to the left of an equal sign). The following is equivalent to the statement 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 CLC; 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.).

3.5.1 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 (refer to Expressions, Section 3.4)
+	Plus	Combines symbols or numbers (add)

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

-	Minus	Combines symbols or numbers (subtract)
!	Exclamation Mark	Combines symbols or numbers (inclusive OR)
↵	Carriage Return	Terminates a statement or a line
→|	Tab	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 CLC; redefines origin
.	Dot	Has value equal to CLC
\	Backslash	$X\backslash Y = 1000_8$ X+Y (X must be a single octal digit.)
/	Slash	Indicates start of comment
&	Ampersand	Combines symbols or numbers (AND)

3.5.2 Illegal Characters

All characters other than those listed above are illegal when not in a comment or TEXT field; 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.

3.6.1 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.

3.6.2 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.

3.6.3 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 field is divided into four 1024_8 word segments, generally used when in LMODE.

For example:

SEGMNT 2	sets CLC to location 4000_8
SEGMNT 1	sets CLC to location 2000_8

Without an argument, the CLC is reset to the first location of the succeeding segment. CLC = .+1777 & 6000; more than 1 sequential SEGMNT pseudo-op with no arguments has the effect of 1 SEGMNT pseudo-op with no argument.

3.6.4 FIELD n (n=0 or 1)

The FIELD pseudo-op indicates the 4K field, 0 or 1. If the field is not specified, 0 is assumed as the initial condition.

3.6.5 PAGE n ($0<n<40_8$)

PAGE is used to reset the CLC to the first location of the specified PDP-8 page. If n is not specified, the CLC is reset to the first location of the succeeding page. CLC = .+177 & 7600. Multiple PAGE pseudo-ops with no argument have the same effect as one PAGE pseudo-op with no argument.

3.6.6 LISTAPE n ($0 \leq n \leq 7777$) DIAL-MS only

The value of n specifies the operation performed by LISTAPE. If n is negative ($4000 \leq n \leq 7777$), the header block (block 447) from the previous assembly is read in and preserved. The present assembly will then alter this bit table rather than destroy it. If n is positive ($0 \leq n \leq 3777$), the four low-order bits (bits 8-11) of n are taken as a unit number (0-17). If the six console Sense Switches are all set to 1 before an assembly, any output encountered by the Assembler after LISTAPE is encountered in pass 1 is written on the specified unit starting at block 0. No check is made for existing files on that unit. The characters are placed one per word, right justified; the leftmost bits are meaningless. If the Sense Switches are not all set to 1, any assembly error messages

are printed on the Teletype and the LISTAPE pseudo-op does nothing. Remember that all listing occurs on pass 2 of the Assembler and therefore goes to the indicated device regardless of the actual location of the pseudo-op in the source.

3.6.7 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.

3.6.8 OCTAL

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

3.6.9 NOLIST

NOLIST is used to prevent printout during an LI assembly (refer to Section 4.2). The NOLIST appears in the listing, but subsequent printout is suppressed until a LIST pseudo-op is encountered.

3.6.10 LIST

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

3.6.11 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 without finding the end of string delimiter, the error message IE (illegal expansion) is printed, and the assembly is immediately terminated.

3.6.12 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. EJECT performs no operation if the listing is to the Teletype.

3.6.13 ASMIFx n

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

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

In each case the expression is evaluted, and if the value of the expression matches the condition specified (zero, nonzero, 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 by setting some values in the beginning of the assembly program.

---

[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.

3.6.14 ASMSKP n

When the pseudo-op ASMSKP and the expression, n, following it are encountered, the expression on the right is evaluated. The number of lines equal to the 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 ASMSKP pseudo-op 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 simply 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 0000.

Consider the following program listing.

```
0042                        ASMIFN TAPE
0043    0202    4401        JMS I WRITE
0044                        ASMIFN TAPE-DISK
0045                        ASMSKP 2
0046                        JMS I READ
0047                        JMS I WRITE
0050    0203    1005        TAD M10
```

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

The two pseudo-ops described next, SAVSYM n and LODSYM, are sophisticated tools which should be used only in large system programs. They are used to link sections of program where the source is not all in one file or where more than one program or subroutine is to be assembled together. These pseudo-ops are discussed briefly here and in detail in Appendix C, Paragraph C.2.

3.6.15 SAVSYM n (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 two cases when a user will want to save his symbols; first, when the user has defined some common definitions which are to be used with all his programs, and second, when the program is too large and must be split into two or more smaller programs that can communicate with each other.

.....

3.6.16 LODSYM

The pseudo-op LODSYM loads a symbol table previously saved by a SAVAYM 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
DIAL Monitor Commands

The DIAL Monitor commands are discussed in this chapter. Remember that the Monitor must be in command mode to accept a command. To enter command mode, the cursor must be under the first character (which must be blank) on a new line. All commands are issued by typing LINE FEED, the command in the correct syntax and RETURN. All DIAL-V2 assembly operations require two tapes and can only access files on unit 0 or 1.

4.1 ASSEMBLE PROGRAM

→AS (NAME, UNIT)↵

NAME = name of filed program to be assembled

UNIT = tape or disk unit on which NAME file is to be found (0-17)

The Monitor command AS performs on assembly of the NAMEd source file on the specified UNIT. If no NAME is given, the source program in the working area on unit 0 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 carriage RETURN. The binary coding is placed in the binary working area which is on tape 1 if no disks are present or on disk 1 (unit 11) if disks are in the configuration.

For all DIAL programs, NAME can be one to eight characters long and must have at least 1 non-numeric character and no? or / characters. For a machine with a line printer, the Assembler will automatically put 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.

4.2 ASSEMBLE AND LIST PROGRAM

→LI (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 assembled and listed

UNIT = tape or disk unit on which NAMEd file is to be found (0-17)

The LIST command performs the same functions as the AS command, but, in addition, produces an octal-symbolic listing on the Assembler output device. It will assemble and list from the working area if no program NAME and UNIT are specified. In DIAL-V2, the value of UNIT can only be 0 or 1. As with the ASSEMBLE command, AS, the binary working area on tape unit 1 is used for the binary output and tag table.

If two line numbers are supplied, the entire program in the working area or a NAMEd file is assembled, but only the portion of the program between the two line numbers is listed. To assemble and list only that part between lines 140 and 160 of the source working area, 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 LISTAPE, NOLIST, and LIST (refer to Section 3.6), the output listing can be controlled. Carriage RETURN may be typed at any time to return to the Editor.

4.3 ASSEMBLE AND 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 assembled and listed
UNIT = tape or disk unit on which NAMEd file is to be found (0-17)

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. In DIAL-V2, UNIT must be 0 or 1. No line numbers are printed 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.

```
Number  Location  Octal Code      Mnemonic          Comment

0000                 *20
0001                            PMODE             /START IN PDP-8 MODE
0002                            LINEP=0           /NO EJECTS
0003                            ASMIFN  LINEP     /EJECT?
0004                            EJECT             /YES.
0005      4020    6032  START,  KCC               /CLEAR KEYBOARD FLAG
0006      4021    6046          TLS               /SET TELETYPE FLAG
0007                            AFSMIFN LINEP     /EJECT?
0010                            EJECT             /GO TO TOP OF PAGE.
0011      4022    6031  LOOP,   KSF               /CHAR. TYPED?
0012      4023    5222          JMP       .-1     /NO. WAIT
0013      4024    6036          KRB               /YES. READ IT IN.
0014      4025    6041          TSF               /TELEPRINTER READY?
0015      4026    5225          JMP       .-1     /NO. WAIT
0016      4027    6046          TLS               /YES. OUTPUT THE CHAR
0017      4030    5222          JMP       LOOP    /GO GET ANOTHER CHAR.

0000  ERRORS

LINEP  0000
LOOP   4022
START  4020
```

This listing was produced by a QUICK LIST command.

```
                0000            *20
                0001             PMODE /
line            0002             LINEP=0 /
numbers         0003             ASMIFN LINEP /
                0004             EJECT /
core            4020    6032    START, KCC /
locations       4021    6046     TLS /
line            0007             ASMIFN LINEP /
number          0010             EJECT /
                4022    6031    LOOP, KSF /
                4023    5222     JMP .-1 /
                4024    6036     KRB /
core            4025    6041     TSF /
locations       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 is to start in LINC mode P if program is to start in PDP-8 mode
ADDRESS	=	starting address (5 digits)

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. If another program is assembled before this assembled binary program is stored, the one assembled first is destroyed. Two tapes are required for this operation with DIAL-V2.

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 when loaded, the unit number is followed by an L. If the program is to start at 04020, the L is followed by typing RETURN. If the program is to be started elsewhere, a full 5-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 causes the program to be started at location 00200; otherwise the full 5-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 04020

→SB PGMNAM,1,L)

To save the program PGMNAM on unit 0 so it will load and start in LINC mode at location 4020 of field 1 (location 20, segment 6)

→SB PGMNAM,0,LI4020)

To save the program PGMNAM on unit 0 so it will load and halt

→SB PGMNAM,0)

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 in DIAL-V2). 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?. Type R to replace the existing file entry with this new entry; type RETURN if the existing file is not to be replaced. Control is then returned to DIAL.

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 NAME is not specified, the program is loaded from the binary working area and the loader halts (at location 7774 for DIAL-V2 and at 7775 for DIAL-MS). If NAME is specified, but the designated program is not self-starting, the program is loaded and the loader halts as above. If the NAMEd program is self-starting, the loader will start the program after loading.

Note that the DIAL-V2 loader overlays locations 7761-7777 of field 0; the DIAL-MS loader will not load locations 7750-7751 of field 0 and locations 7600-7777 of field 1. For both versions of DIAL, location 0 of the starting segment for self-starting LMODE programs is destroyed by the loader.

4.6 ADD BINARY (DIAL-MS only)

→ AB(ADDR,(FIELD,))NAME,UNIT↵

ADDR = 4-digit (12-bit) address
FIELD = field number (0 or 1)
NAME = name of a binary file on UNIT
UNIT = unit on which file NAME may be found (0-17)

The binary file is copied to the binary working area, omitting zeroes. Typically, ADD BINARY will be used without specifying ADDRESS and FIELD to combine standard subroutines with user-written main programs. For example, if a user has written a program which requires QANDA which is a binary file on unit 0, and his program is in the source working area, the correct sequence is to ZERO the binary working area (refer to Section 4.7) and then assemble the program. If there are no errors, use ADD BINARY for QANDA, and test the result.

→ZERO↵
→AS↵
→AB QANDA,0↵
→LO↵

By adding QANDA in binary form, rather than source form with the ADD PROGRAM command, the user has saved considerable assembly time and avoided the possibility of tag duplications between his program and QANDA.

If his program also needs FPP, for example, it can be added by the command

```
→AB  FPP,0
```

Thus, many subroutines can be combined with the user's program, without necessitating re-assembling each program. The result may be saved, with SAVE BINARY, as if the whole had been assembled together.

ADD BINARY is also useful for patching binary programs, as follows:

```
→CL
→ZERO
→AB
  *X              /X IS THE FIRST LOCATION TO PATCH
  .               /INSTRUCTIONS OF PATCH GO HERE
  .
  .
  LISTAPE -1      /PRESERVE HEADER
→AS
→SB PROGRAM,U
```

Refer to Section 3.6.6 for an explanation of the LISTAPE -1 pseudo-op.

Note that, in general, for ADD BINARY to function properly it must be used in conjunction with the ZERO command. ZERO should always be used before assembling a program whose binary may later be added, and before adding a binary or binaries to the working area.

The binary file is added to the binary working area via a word-wise OR operation. Thus, each nonzero word from the binary file replaces the old word at the corresponding (relocated) location in the binary working area. All zero words in the binary file do not replace their counterparts.

Advanced users may want to use the ADDRESS and FIELD parameters to specify a new core location for the binary file. If ADDRESS and FIELD are not specified (or both are zero), it will be moved to its assembled address. If ADDRESS is specified and FIELD is not, it will be moved to ADDRESS in field 0. FIELD can not be specified without ADDRESS. No address adjustment within the assembled code is performed.

ADDRESS, if specified, is taken to be the new address of the first word of the first block of the binary file. The relocation of the binary is equal to

$$[(\text{FIELD}*10000 + \text{ADDRESS}) - (\text{ORG}-7400)]$$

where ORG is the lowest origin in the assembly.

A binary may not be moved to field 0, location 0, because this is the condition recognized as no relocation. Any program, therefore, which may be used starting at 0 must be assembled with origin 0. It may be moved to any other address by specifying the new address in the ADD BINARY command.

If ADDRESS and FIELD are specified such that any portion of the binary file would be moved to an address above 20000 (i.e., in field 2 or higher), that portion of the file is ignored.

No address adjustment is performed by ADD BINARY when a binary is relocated. It is thus necessary that a program which is to be moved include self-relocating code, so that it can determine its location at execution time.

The ADD BINARY facility allows the user to build large sophisticated programs (binary) from the results of many small assemblies and standard subroutine packages. This means that bugs can be corrected by re-assembling a small portion of the entire program.

A sophisticated program might consist of a main program, MAIN, user-written subroutines, SUB1 and SUB2, and standard subroutines, QANDA and FPP. The binary might be constructed as follows (MAIN is assumed to be in the source working area):

```
→ ZERO
→ AS SUB1, 0
→ SB SUB1, 0
→ ZERO
→ AS SUB2,0
→ SB SUB2,0
→ AB SUB1,0
→ AB 7000,QANDA,1
→ AB FPP,1
  LISTAPE-1              /PRESERVE BIT MAP
→ AS
→ LO
```

If a bug is found in MAIN, the procedure to correct it is to restart DIAL-MS, change the source, re-assemble MAIN (using LISTAPE -1) and try again. Note that the Assembler will store explicitly defined zeros (HLT, AND 0, etc.), but will not zero storage skipped over by an origin or segment pseudo-op.

4.7 ZERO (DIAL-MS only)

The ZERO command fills the entire binary working area with zeroes and clears the block map, guaranteeing that any location not used in a subsequent assembly will be zero. There are three major applications of ZERO:

a. 0000 is HLT in LINC mode. Therefore, filling the binary working area with zeroes is equivalent to filling unused core locations with HLT. Thus, a program being tested halts if it jumps to an unused location.

b. The paper tape output option in PIP, DZ, (refer to Section 5.5), when combined with the ZERO command, allows the user to assemble and punch short patches to binary programs, with the resulting tape only as long as the patch.

c. The ADD BINARY command depends on the use of the ZERO command in two instances:

(1) Before assembling a program to be saved and later added.

(2) Before a group of ADD BINARY commands, the ZERO command is required because ADD BINARY does not copy zeroes from the file in the working area. This is done to enable the user to make effective patches and overlays easily.

Press the RETURN key to return to the source working area display and text mode at any time.

4.8 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 pressed at any time. This will interrupt the command and return to the source display with no effect, since DIAL has not 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.

To save the source file PETE on disk unit 12, the command is →SP PETE,12)

4.9 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 in the source working area or to call a previously stored source program into the working area for editing, etc., 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. Lines 15 through 361 of the source file EXP6 on unit 4 are added to the current source by the command

→AP 15,361,EXP6,4)

4.10 CLEAR

→CL)

The source working area on tape unit 0 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 source working area without having to manipulate any console switches.

4.11 DISPLAY INDEX

→DX (,UNIT))

UNIT = unit whose index is to be displayed

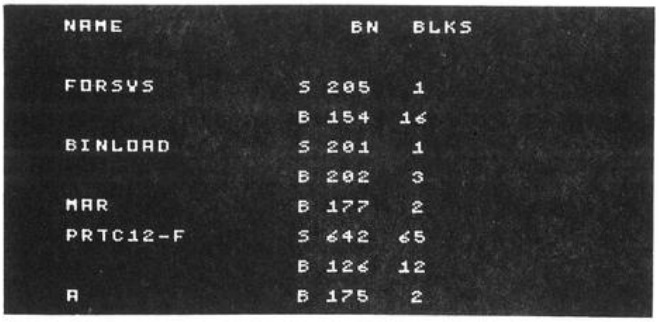
The tape file index of the specified UNIT is displayed on the scope by the command →DX. For each program on the tape, its name, source and/or binary, starting block number, and length in blocks are indicated.

To view the entire index, use the following keys to modify the display. A frame is the number of lines defined by the setting of A/D knob 7.

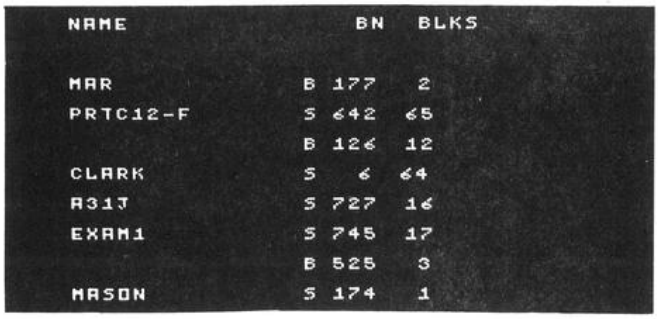
Key	Action
1	Forward one frame
2	Forward one entry
Q	Backward one frame
W	Backward one entry

Press the RETURN key to return to the source working area display and text mode at any time.

Entries may be deleted from the index. 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 typing the colon key (:) after erasing the desired entries. The DIAL working area display is returned to the scope. Note that if there is no index or an empty index, NO is displayed on the scope. Consider the index of unit 11.



Delete the binary file, A, by typing RUBOUT. Move the display forward to view the rest of the index by typing 1.



To delete the source file EXAM1, type W twice so that it is the last entry on the display. Then type RUBOUT. Make this deletion permanent by typing colon.

4.12 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 working area display.

To check the modified index on unit 11 above, type

→ PX,11↵

The index printed is

	SOURCE		BINARY	
NAME	BN	BLKS	BN	BLKS
FORSYS	205	1	154	16
BINLOAD	201	1	202	3
MAR			177	2
PRTC12-F	642	65	126	12
CLARK	6	64		
A31J	727	16		
EXAM1			525	3
MASON	174	1		

4.13 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 program is printed on the Teletype from the specified UNIT. The source currently in the working area is printed when no NAME and/or UNIT are designated. If two line numbers are specified, that portion of the NAMEd file is printed.

Any DIAL source can be printed with the PS command. No words are 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 printout. When only one line number is specified, it is assumed to be the start of the printout and the rest of the source on unit 1 is printed. To print lines 23 through 151 of file 4JA-1 on to the Teletype, use the command

→ PS 23,151,4JA-1,1↵

4.14 EXIT

→ EX↵

The EXIT command completes the updating of the source working area from the memory buffers, thus assuring the user of leaving DIAL without losing the current source program in the working area. An EXIT command should be issued when editing is completed with the PDP-12. After → EX↵ , DIAL halts. Press the CONTinue console switch to return to DIAL. The program that was in the working area when the EXIT command was issued is still there, and any legal DIAL operation can now be performed.

4.15 USER'S MONITOR COMMAND

→ MC X(Y),UNIT↵

X(Y) = argument(s) to be passed by the Editor to a user program via the AC (argument Y is optional)

UNIT = unit to be read

The USER's MONITOR command allows access to the free blocks of a DIAL tape. When the MC command is issued, block 270 of UNIT is read into core locations 4000-4377; the arguments XY are placed in the AC and the Editor turns program control over to the initial free block of code at location 4020 in LMODE.

Tape blocks 270 through 277 are defined as the free blocks of a DIAL system device. (Refer to Appendix C, Paragraph C.6 for tape and disk allocation.) Except for the MC command, DIAL never accesses these TBLKs; they are meant for the user. The contents of block 270 are left to the user's discretion. The MC command has no meaning if the user's binary at block 270 is not aware of the Editor's action during the execution of an MC command. The MC command is obviously not meant for the beginning programmer.

Consider the following examples.

If the user wanted to expand DIAL to process additional Monitor commands, X_1X_2 and X_3X_4, that are not defined in DIAL, they can be defined in the free area and called via the Editor by:

→ MC X_1X_2, U)
→ MC X_3X_4, U)

Then, depending upon the contents of the AC, the program loaded from block 270 would either do the function X_1X_2 or X_3X_4.

If the user had a program to be loaded into Locations 7600-7777 of field 1 (the DIAL loader does not access these locations), the binary can be placed in the last 200 words of TBLK 270 and a bootstrap routine in the first 200 words to move locations 4200-4377 to 7600-7777. Then the program can be loaded by

→ MC A, U)

Chapter 5
Peripheral Interchange Program (PIP)

The Peripheral Interchange Program (PIP) for the PDP-12 permits the user to transfer source or binary files between I/O devices, including disk (RS08 or RK08), LINCtape, high-speed and Teletype paper tape reader/punch, card reader, and line printer.

5.1 INITIALIZING PIP (DIAL-MS)

The command for calling PIP, →PI↵ , loads PIP from the system unit; for DIAL-V2 users or DIAL-MS users running without an RS08 or RK08 disk, →PI↵ is equivalent to the command →LO PIP,0; for DIAL-MS users with RS08 or RK08 systems, it is equivalent to →LO PIP,10↵ .

To implement the →PI↵ command call on a system using DIAL-MS with an RS08 or RK08 disk, PIP must be copied onto unit 10 after initialization (refer to Appendix A, Paragraph A.3). Copy tape 0 to disk unit 10 by

a. Typing →LO PIP,0↵ .
b. Responding to the PIP displays with A↵ , U↵ , L0↵ , R0↵ , in that order.

If just PIP is to be copied to the system device, respond to the PIP displays with B↵ , L0;PIP↵ , R0;PIP↵ , in that order.

Note that when using PIP, disk units may be called as 0 through 7 rather than 10 through 17, as required for the other DIAL-MS programs. Either value is acceptable. Thus, R0 above is equivalent to R10.

5.2 CONTROL COMMANDS

PIP implements four control commands that may be issued at any time while it is being used.

Command	Action
CTRL/P	Return to PIP options display
CTRL/D	Return to DIAL
CTRL/U	Rewind each transport that is set to REMOTE and has a tape on it and position unit 0 at block 0. Repeated typing of CTRL/U will cause tape 0 to be unloaded also. This option is used to minimize tape handling for all units and to decrease duplication time for unit 0.
CTRL/T	Rewind all LINCtapes completely (as with CTRL/U) and position tape 0 to block 300 to permit a faster DIAL-MS restart. Typing CTRL/T several times will unload unit 0.

5.3 MODE OPTIONS

When PIP is started, the following message is displayed. Note that lower-case letters are used in this chapter 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 three single letter abbreviations above needs to be typed after reply.

This PIP display and all subsequent displays 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. Terminate a reply with a carriage RETURN. 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.

5.4 BINARY OR SOURCE INPUT

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

```
                INPUT DEVICE

        c------card reader
        h------high speed reader
        l------linc tape
        r------rs08,rk08 disk
        t------teletype

    reply:
```

The auxiliary mode option is described in Section 5.6.

If LINCtape or disk is the selected input device, the user's response to the second PIP display must be in the format Ln;NAME or Rn;NAME, where L indicates LINCtape, R indicates disk, n is the input unit number (0-17) followed by a semicolon, and NAME is a 1- to 8-character file name. Thus, to input a file named ABC3 from LINCtape unit 4, the correct command is L4;ABC3.

If a card reader is to be the input device, type C after having chosen source mode. 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 through 110 (octal), the correct reply string is C50;THRU,110. The character codes used are not the card reader codes in the PDP-12 User's 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
0-8-2	NONE	]
0-8-5	_ (underscore)	←
11-8-7	¬ (Logical not)	\
12-8-2	¢ (Cent Sign)	[
12-8-7	| (Vertical Bar)	↑

If the user desires, he may substitute an entirely different character set into PIP (with the exception of BLANK). Refer to the PIP Internal Description, DEC-12-ZW2A-D.

If, for binary input, the device is to be the high-speed reader or the Teletype (neither is file or unit oriented), the user's response must be 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 is not automatically started when loaded. The Mode must be specified for the program to start after it has been read. More than one tape can be input during one PIP session (refer to Section 5.5). The punctuation marks are always required if the items after them are specified. If no Address is given, a LINC mode operation starts at location 4020, and a PDP-8 mode operation starts at location 200.

When the high-speed reader is in 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 was originally punched by PIP, a CTRL/Z is already present at the end of it.

Consider the following examples.

H;P	Input is from the high-speed reader and the program, when loaded, will be started in PDP-8 mode at location 200.
H1;P,1000	Input is from the high-speed reader and the program, when loaded, will be started in PDP-8 mode at location 1000 in memory field 1.
T;L,6000	Input is from the Teletype and the program, when loaded, will start in LINC mode at location 6000.
H	Input is from the high-speed reader and the program, when loaded, 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.

5.5 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
p --- line printer
r --- rs08, rk08 disk
t --- teletype

reply:
```

If LINCtape or disk is to be the output device, a response in the same format as was used to specify them 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 destroys the NAMEd file. To locate the file PETE on LINCtape unit 0, the correct command string is L0;PETE.

When using LINCtape or disk (DIAL-MS) as the input or output device for these PIP operations, be sure to specify the name of the file. If the file name is omitted when specifying LINCtape or disk 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 LINCtape or disk 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, unit 0 of the specified device type is assumed.

If there is not sufficient room on the indicated device to perform the requested action, 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 generate 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 Teletype or high speed, paper-tape punch is chosen as the output device, an option may be used so that zeros are not punched. It is specified by the reply T;DZ or H;DZ to the output device display. Note that the tape produced does not contain a halt instruction if in LINC mode. If zeros are removed from the middle of a section, however, a new origin is punched, so that the remaining code will be loaded properly. This option is especially useful for patching programs in conjunction with the ZERO command (refer to Section 4.7). For example, the following sequence generates a short paper-tape of the program SCW on the high-speed punch. The initial ZERO command causes all locations to contain zeros and therefore are not punched by the H;DZ statement.

```
→ ZERO
  SEGMNT 1
  *100
   1
   2
   3
→ AS
→ SB SCW,11
→ PIP
   B
   R11;SCW
   H;DZ
```

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 and a carriage RETURN. A reply of A will store the next file immediately after the first one. Each tape must be terminated by a CTRL/Z; if CTRL/Z is not on the tape, it must be typed on the Teletype after the tape has been read. 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.

5.6 AUXILIARY MODE

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

```
                   OPTIONS

          c------copy specified blocks
          d------duplicate tape 0 onto 1
          s------copy system
          u------copy unit

reply:
```

In addition to specifying the option, a number can follow any of the replies to indicate the number of consecutive units onto which the specified blocks of tape are to be transferred.

The S option copies blocks 300-345 and 350-370, thus copying the entire DIAL system except for the index. The U option duplicates the specified unit onto other specified units. A reply of D will perform that operation immediately. A response of C, S, or U will produce the second PIP display requesting the input device. The only acceptable input devices are disk and LINCtape. For the C option, the reply must be in the format Dn;fb,nb, or Ln;fb,nb where D is disk and L is LINCtape, n is the unit number, fb is the first block, and nb is the number of blocks to be read. Thus L2;63,24 will start input from LINCtape unit 2, block 63, and continue for 24 blocks. Note that block numbers are octal. For the S or U option, only the device and unit must be specified; block limits are not required.

When the input response has been accepted, the PIP output display is seen on the scope for the C, S, and U options. The only acceptable output devices are disk and LINCtape. For the C option, response must be in the form Dn;fb or Ln;fb where D is disk and L is LINCtape, n is the number of the first consecutive tape unit, and fb is the block where output is to start. The S and U options require only that the device and unit number be specified.

5.7 ERRORS

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

```
DISK (TAPE) ERROR
  AT BLOCK nnnn

a------accept as is
r------ try again (repeat tape operation)
s ------try to skip past error

reply:
```

For a disk error, the leftmost digit of the block number, nnnn, is the disk unit number (0-7).

It is up to the user's discretion to choose one of the three options above. For a disk error, the disk status register can be checked to determine the type of error. If the problem is a minor hardware error, for example, 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 be followed only by a return to PIP or DIAL.

For an RK08 system, the message SEEK ERROR is displayed for 5 seconds if the movable head on the disk is mispositioned. PIP will automatically recalibrate the disk and try again. If this message arises often, the disk should be serviced. Note that if this message appears during startup, it is because the disk head may be randomly positioned. Remember that systems with RK08 disks only permit units 10-15. Addressing additional units with PIP will generate one of several error messages on the display, possibly SEEK ERROR.

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

5.8 RECOVERY

If the input for a PIP operation is binary paper tape and output is to tape or disk and NO is displayed, a SAVE BINARY command may be issued after CTRL/D to save the program on another unit without reloading the tape. Call the program from the unit to which it was output. (This procedure is illegal for systems with two or more DF32 disks.) To reclaim a source program if PIP displays NO, issue the following appropriate command after typing CTRL/D

```
→ AP370,1↵      if no disk or a DF32 disk is present
or
→ AP370,11↵     if an RS08 or RK08 disk is present
```

5.9 PIP EXAMPLES

The following command sequences are examples of PIP operations.

1. Copy the binary file BINFILE from unit 0 to unit 6.

```
B
L0;BINFILE
L6;BINFILE
```

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

```
S
L3;HOW
L7;HOW
```

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

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

4. Read in a binary tape via the high speed reader, and store it as file J10 on unit 0, where it will start at location 200 in PMODE.

```
B
H;P
L0;J10
```

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

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

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

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

This sequence could result in the destruction of the file COPY1.

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

```
S
L3;HERE
H
```

7. Read in a tape on the Teletype that was not originally punched by PIP, and store it as file MARTY on tape unit 0.

```
S
T
L0;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.

8. Duplicate tape 0 onto tape 1.

```
A
D
```

9. Duplicate tape 0 onto tapes 1, 2, 3, 4, and 5.

```
A
D5
```

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

```
A
C
L4;300,30
L6;200
```

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

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

12. Copy the system area tape from unit 0 to LINCtape unit 3.

```
A
S
L0
L3
```

13. Copy the tape on unit 1 onto logical disk units 2 and 3.

```
A
U2
L1
R2
```

Appendix A
DIAL-V2 vs. DIAL-MS

A.1 FEATURES

In addition to the general system capabilities outlined in Section 1.1, the following features are unique to each version of DIAL:

a. DIAL-V2

(1) Overlapped tape I/O to accept user input during a tape read or write operation. (When DIAL-MS is run on tape, input is not accepted while the tape is moving.)

(2) 4K system.

b. DIAL-MS

(1) Fully integrated tape-disk system with mass storage support for one or two DF32, one to four RS08 or one RK08 disks.

(2) Editor commands to clear the binary working area (which is used in conjunction with the PIP option not to punch zeros) and to merge binary files.

(3) I/O routines to read, write, or move data.

(4) Increased Assembler facilities for processing large programs.

(5) 8K system.

A.2 SYSTEM BUILD FOR DIAL-MS

The following system-build procedure must be executed to "build" a DIAL-MS tape. Part of the procedure is running the program, GENASYS, provided on the system tape

Step	Procedure
1	Start the system by the procedure in Appendix A.4.
2	Type →LO GENASYS,0) to load GENASYS to specialize a tape for DIAL-MS on this system configuration.
3	After the message TAPE UNIT CONTAINING GENASYS: is printed, type 0.
4	GENASYS then asks TAPE UNIT FOR DIAL-MS: Type an octal digit to indicate the tape unit on which DIAL-MS is to be built. If the reply is not 0, GENASYS does not initialize DIAL-MS, the procedure in Section A.3 must be performed to initialize the tape. If the reply is 0, another message is printed on the Teletype after GENASYS has built DIAL-MS: PRESS CONTINUE TO INITIALIZE DIAL-MS: Press the CONTinue key. The last message is

Step	Procedure
	WHEN EDITOR DISPLAY APPEARS, TYPE (LINE FEED) EX (RETURN)
	After the message is printed, GENASYS initializes DIAL-MS. When that is done, the Editor display appears. Type→EX↵ and then press CONTinue. DIAL-MS is ready to accept user input.

The following error conditions can occur.

a. If GENASYS is not able to find 8K of core, the following message is printed and GENASYS returns to DIAL-V2.

   THIS MACHINE HAS ONLY 4K, DIAL-MS REQUIRES 8K.

b. If GENASYS is not able to find one of the four binary files needed to build DIAL-MS, the following message is printed:

   THIS TAPE DOES NOT CONTAIN BINARY DIAL-MSx, NEEDED FOR GENASYS.

   where x is a digit, 1 to 4. Continue at Step 4 above.

c. If one of the binary files is the wrong length, the following message is printed:

   LENGTH ERROR IN DIAL-MSx.

   where x is a digit, 1 to 4.

A.3 SYSTEM INITIALIZATION

A system using DIAL-MS must be initialized. This procedure causes:

a. the initial tape to modify itself to become a startup tape by building a set of I/O routines for handling the user's particular disk configuration.

b. the initialization routine to copy the DIAL-MS system area (blocks 270 to 345 and 350 to 467) from tape 0 into the appropriate area on the disk.

System initialization may be automatically carried out by GENASYS (by a reply of 0 in Step 4) or may be performed when the equipment configuration changes, if the contents of the disk are lost for any reason, or GENASYS does not perform the procedure, e.g., because a tape unit other than 0 was specified in Step 4.

The procedure is

Step	Procedure
1	Mount the DIAL-MS tape on tape unit 0.
2	Mount another LINCtape on tape unit 1 (required if configuration has no disk or only one DF32 disk).
3	Set the scope channel knob to position 1 & 2.
4	Set the switches of both tape units to REMOTE and set unit 0 to WRITE ENABLE.
5	Set all disk units to WRITE ENABLE. A single DF32 disk unit must be set to 0. A second DF32, if present, must be set to 1.
6	Set the mode switch to LINC mode and press the I/O PRESET key.

Step	Procedure
7	Set the Left Switches to 0701 and the Right Switches to 7310 by pushing down the front part of the switches indicated by ↑ and pushing down the back part of those indicated by ↓ in the following diagram.

↑↑↑ ↓↓↓ ↑↑↑ ↑↑↓ — Left Switches

↓↓↓ ↑↓↓ ↑↑↑ ↑↑↑ — Right Switches

8	Press the DO key.
9	When the tape has stopped spinning, press the START 20 key.
10	Type →EX) when part of the DIAL-MS program appears on the display in order to preserve the DIAL-MS pointers. Press CONTinue. (This step is essential - it updates the Editor's pointers.)
11	Set the A/D knob 3 all the way to the right. A DIAL command may now be issued.
12	If PIP is to be used, refer to Section 5.1.

A.4 SYSTEM STARTUP

The following procedure is performed every time DIAL-MS (and DIAL-V2) is to be used. For DIAL-MS it is assumed that the tape and disk have been initialized (refer to Section A.3) so that the DIAL-MS system loaded from tape is aware of available disks and will use them appropriately.

Step	Procedure
1	Mount DIAL tape on tape unit 0.
2	Mount another LINCtape (scratch) on unit 1.
3	Set scope channel knob to position 1 & 2.
4	Set the switches of both tape units to REMOTE and WRITE ENABLE.
5	Set all disk units to WRITE ENABLE. A single DF32 disk must be set to 0; a second DF32, if present, must be set to 1.
6	Set the mode switch to LINC mode and press I/O PRESET.
7	Set the Left Switches to 0701 and the Right Switches to 7300 by pushing down the front part of the switches indicated by ↓ and pushing down the back part of those indicated by ↑ in the following diagram.

↑↑↑ ↓↓↓ ↑↑↑ ↑↑↓ — Left Switches

↓↓↓ ↑↓↓ ↑↑↓ ↑↑↑ — Right Switches

8	Press the DO key.
9	Press the START 20 key when the tape has stopped spinning.

The version of DIAL on the tape is started and ready for any DIAL operation.

It is possible for an error condition to result while initializing the system, causing the computer to halt with the status register[1] for the device in the Accumulator; this informs the user of the exact nature of the error. The Right Switches are then set to indicate the error recovery method. If the user sets the Right Switches to anything but 7777, the operation is tried again. This retry facility is activated by pressing the CONTinue console key. It is good practice to retry the operation that produced the error at least once. If the Right Switches are set to 7777, the data will be accepted by the computer after pressing the CONTinue key.

At the completion of this operation, the tape on unit 0 has been modified for the user's particular configuration. It may be copied with the auxiliary "S" option of PIP if multiple copies are needed (refer to Section 5.6). The initialization procedure may be repeated at any time, but it is necessary only when the DIAL-MS system on the disk is lost or when the configuration changes.

[1] Refer to PDP-12 User's Handbook, DEC-12-SRZA-D.

Appendix B
Assembly Error Messages

During source program assembly, error messages in the form of a two-letter code are included in the program listing. These messages, defining illegal syntax or insufficient space errors, 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 means other than the equal sign; the symbol was not redefined.
IE	Illegal Expansion - Delimiter missing in text.
IR	Illegal Reference - An off-page reference was made.
SE	Symbol table Exceeded - Assembly is terminated and control is returned to DIAL; up to 895 user symbols, maximum.
US	Undefined Symbol - A symbol has been processed during pass 2 that was not defined before the end of pass 1.
WA	Binary 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 (refer to Appendix C, Paragraph C.2).
PS	Push-down Stack exceeded - Too many symbols to be evaluted on one line.

Appendix C
Summaries

C.1 COMMAND SUMMARY

All commands are issued in the form

→ Command)

Command	Function
AS (N, U)	Assemble
LI (L, L,)(N, U)	Assemble and List
QL (L, L,)(N, U)	Assemble and Quick List
LO (N, U)	Load Binary
PS (L,)(L,)(N, U)	Print Source
AB (A,)N, U	Add Binary
SB N, U(, M(FA))	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 Source Working Area
ZE	Zero Binary Working Area
PI	Peripheral Interchange
EX	Exit
MC X(Y), U	User's Monitor Command
L	Line Call

Legend:

() indicates an optional parameter	F = Field
N = File name	B = Tape block number
U = Tape (0-7) or disk (10-17) unit	X(Y) = Characters in accumulator
L = Line number	→ = LINE FEED
M = Mode (L or LINC or P for PDP-8)	) = Carriage RETURN
A = Address (4 digits)	

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	Set 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
LISTAPE	8/L	Header block and Assembler listing control
LMODE	8	Causes subsequent coding to be interpreted as LINC instructions
LODSYM	8/L	Load saved symbol tape (see below)
NOLIST	8/L	Inhibit octal-symbolic listing during assembly
OCTAL	8/L	Set octal radix for integer input
PAGE n	8/L	Start new page at n·200. If no parameter, start at next page ($0 \leq n \leq 40_8$)
PMODE	L	Causes subsequent coding to be interpreted as PDP-8 instructions
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 two cases requiring use of the SAVSYM pseudo-op and the procedure for assembling large programs with the SAVSYM, LODSYM, and LISTAPE 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             /three disks
TTYIN=7423           /address of Teletype input routine
TTYOUT=7520          /address of 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 four user-defined symbols and all symbols to follow are 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 are 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.

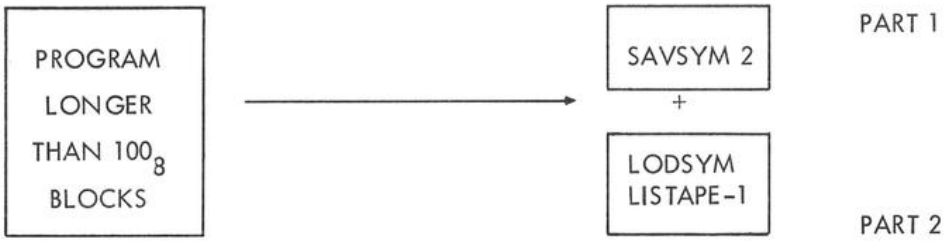
In either case, the symbols will be permanently saved unless one of the following conditions occurs:

a. The tape is erased.
b. The tape is used for unit 0 workspace (source) and is overwritten.
c. Another SAVSYM command is given at some later time and replaces the old symbol table with a new one.

C.2.1 Assembling Large Programs with SAVSYM, LODSYM, and LISTAPE

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, LODSYM and LISTAPE (in DIAL-MS) are employed in the following manner.

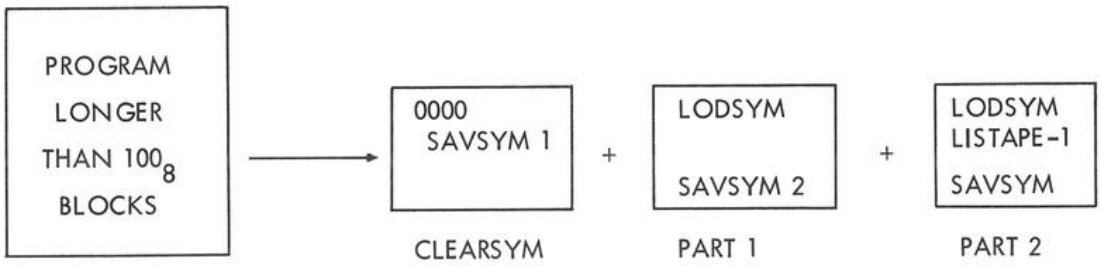


A SAVSYM 2 during the assembly of PART 1 will save all the symbols from PART 1. The LODSYM at the start of PART 2 loads in the saved symbol table, thus putting all the symbols from PART 1 and PART 2 into a common symbol table. For DIAL-MS, LISTAPE with a negative argument can be used to preserve the bit map from the assembly of PART 1.

The program is then assembled by the commands

```
→ AS PART 1
→ AS PART 2
```

This method will work only 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.

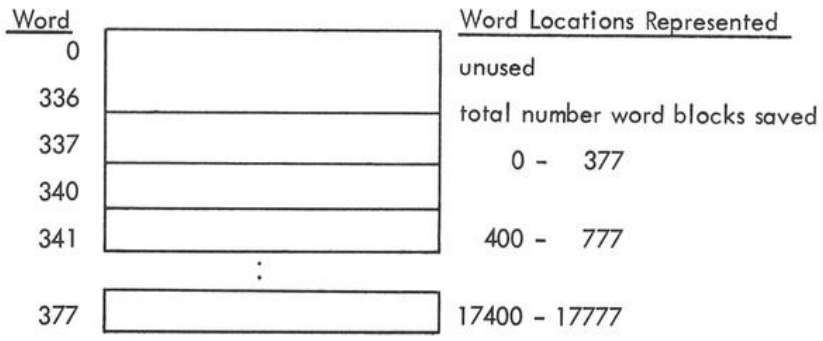
When starting an assembly with DIAL-V2 only, the Assembler destroys the binary coding in locations 4000-4377. Therefore, the user must assemble the part containing this section of code last. If the program was split so that locations 4000-4377 are in PART 2, then the sequence above will assemble the 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 in DIAL-V2 it 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.

BLOCK 447



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. Use of the LISTAPE pseudo-op in DIAL-MS inhibits destruction of the header block at the start of a new program assembly.

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	323	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
+	253	53
,	254	54
-	255	55
.	256	56
/	257	57
0	260	60
1	261	61

Keyboard	External (ASCII)	Internal
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 (not displayed)
RUBOUT	377	None (not displayed)
CONTROL/I (TAB)	211	47 (not displayed)

C.4 INSTRUCTIONS

C.4.1 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

Mnemonic	Octal	Operation
	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
	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	62n4	read data field into AC
RIF	6224	read instruction field into AC
RMF	6244	restore memory field
RIB	6234	read interrupt buffer

Mnemonic	Octal	Operation
	IOT MICROINSTRUCTIONS (cont)	
Processor Mode Change		
LINC	6141	change to LINC mode processing

C.4.2 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	0240	rotate left N places
ROR N	0300	rotate right N places
SCR N	0340	scale right N places
	OPERATE	
HLT	0000	halt
NOP	0016	no operation
CLR	0011	clear AC and LINC
SET	0040	set register N to contents of register Y
JMP	6000	jump to register Y
QAC	0005	MQ transfer to AC
	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	0017	complement AC
	SKIP	
		Skip next instruction if:
SAE	1440	AC equals memory register, Y
SHD	1400	right half AC unequal to specified half of memory register, Y
SNS N	0440+N	sense switch N is set
SKP	0456	unconditional skip
AZE	0450	AC equals 0000 or 7777

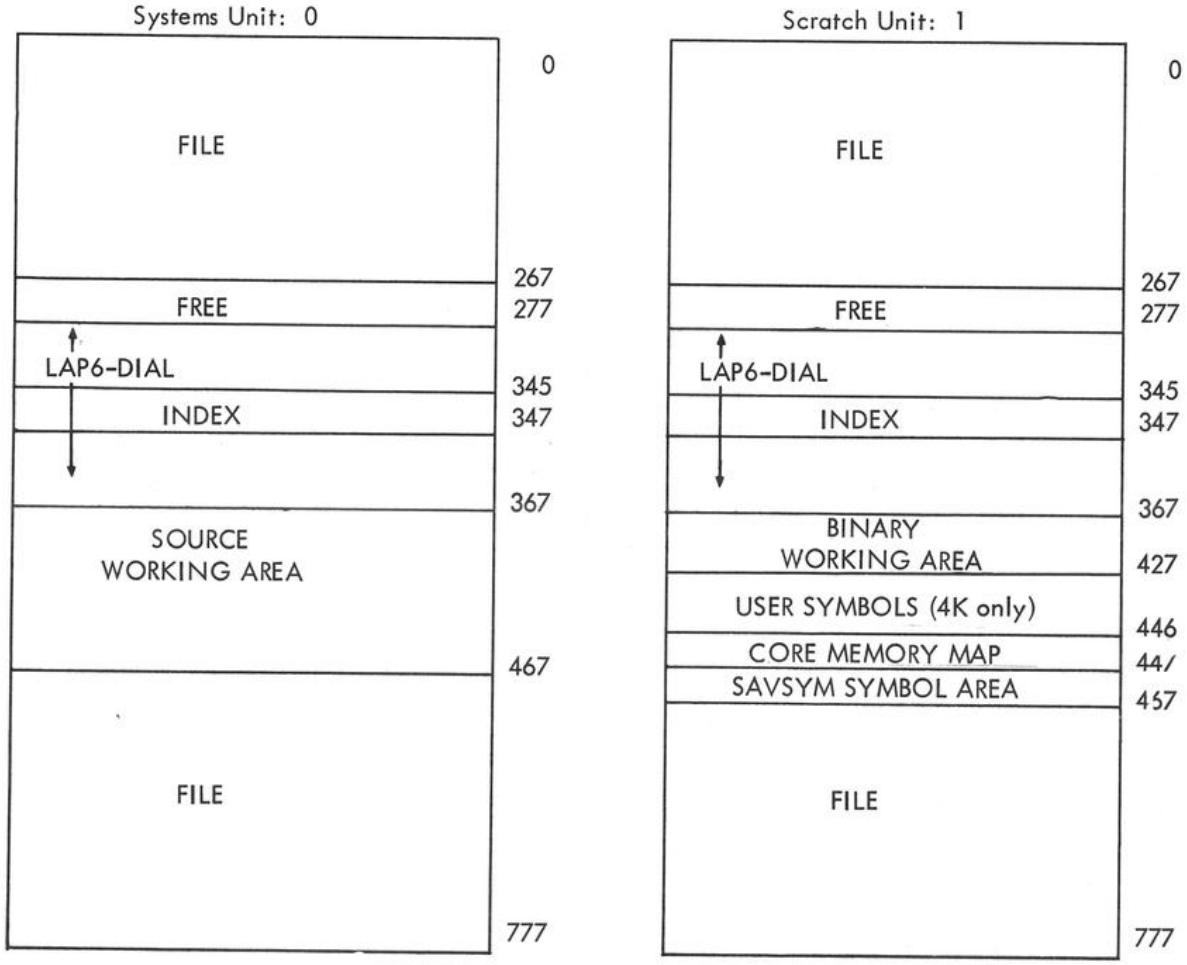
Mnemonic	Octal	Operation
	SKIP (cont)	
APO	0451	AC contains positive number
LZE	0452	link bit equals 0
FLO	0454	add overflow is set
QLZ	0455	bit 11 of Z register equals 0
SXL N	0400+N	external level N is preset
KST	0415	keyboard has been struck
SRO	1500	rotate memory register right one place; then if bit 0 of Y equals 0, skip next instruction
XSK	0200	contents of Y equal 1777; index memory register if 1 bit set
STD	0416	tape instruction completed
	INPUT/OUTPUT	
ATR	0014	AC to relay buffer
RTA	0015	relay buffer to AC
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 AC
LSW	0517	LEFT SWITCH register to AC
IOB	0500	I/O bus enable
	MEMORY	
LIF	0600	change instruction field
LDF	0640	change data field
	LINC TAPE	
RDE	0702	read one block into memory
RDC	0700	read and check one block
RCG	0701	read and check N consecutive tape blocks
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
XOA	0021	extended tape operations buffer to AC
	EXTENDED OPERATIONS	
ESF	0004	enable special functions
TAC	0003	tape control register to AC
TMA	0023	AC to tape control register
AXO	0001	AC to extended operations buffer
DJR	0006	disable Jump Return Save
MSC	0000	miscellaneous
SFA	0024	special functions to AC

C.5 OPERATORS AND SPECIAL CHARACTERS

Char	Mode	Operation
'	8/L	Assign symbolic address
*	8/L	Origin - dependent on mode (LINC or PDP-8)

Char	Mode	Operation
=	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
|	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\y = 1000_8x + y where x is a single octal digit

C.6 LAP6-DIAL TAPE ALLOCATION DURING ASSEMBLY



C.6.1 Allocation of LAP6-DIAL Area

DIAL-V2

Blocks	Content
300-321	EDITOR
322-345	ASSEMBLER
346-347	INDEX
350-353	FILECOMS
354-355	LOADER
356-360	SAVE BINARY
361-362	PX, DX
363	TTY
364	PS
365-366	UNUSED
367	SCRATCH

DIAL-MS

Same as DIAL-V2 except:

Blocks	Content
310	SYS BUILD
322-323	I/O ROUTINES
324-344	ASSEMBLER
345	I/O CONTROLLER
356-360	UNUSED
363	PS
364	TTY
365-366	MASTER I/O ROUTINES

C.6.2 DIAL-MS Disk Allocation

Disk allocation for the available PDP-12 mass storage configurations with DIAL-MS is as follows.

Disk Used	Contents
1 DF32	Disk contains DIAL-MS system and source working area
2 DF32's	First DF32 as above; second disk contains binary working area
1-4 RS08's	First disk (logical Units 10 and 11) contains DIAL-MS system program and source and binary working areas and files assigned to logical units 10 and 11. Up to three additional units may be used. Each RS08 disk is equivalent to two LINCtapes and each may be used for storage of source and/or binary files.
1 RK08	Logically equivalent to six LINCtapes; DIAL-MS system programs and source working area are on first unit. Binary working area is on the second, source and/or binary files are on all six units.

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 0 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.

```
0000                    *20
0001                    /
0002                    /       THIS PROGRAM WILL READ IN A PAPER TAPE
0003                    /       AND PLACE THE CONTENTS OF IT ON LINC TAPE.
0004                    /       THE PAPER TAPE IS READ AS AN 8 BIT CODE
0005                    /       AND IS PACKED ONE CHARACTER PER WORD,
0006                    /       RIGHT JUSTIFIED, WITH THE LEFT BITS
0007                    /       CONTAINING ZERO.
0010                    /       THE DATA IS WRITTEN ONTO THE WORKING
0011                    /       AREA OF UNIT 1, STARTING AT BLOCK 370.
0012                    /       SENSE SWITCH 0 IS THE END OF INPUT SIGNAL.
0013                    /       IF THERE IS TOO MUCH DATA, THE PROGRAM
0014                    /       WILL HALT WITH 7777 IN THE AC.
0015                    /       WHEN FINISHED, THE PROGRAM WILL RESTART DIAL.
0016                    /
0017                    /
0020                            *1
0021                    /
0022                    /
0023    0001    0000    BLOCKN, 0               /CURRENT BLOCK NUMBER
0024                                            /BEING WRITTEN ON.
0025    0002    0000    BLOCKC, 0               /NUMBER OF BLOCKS LEFT
0026                                            /IN THE WORKING AREA.
0027    0003    0000    POINT,  0               /POINTS TO NEXT FREE
0030                                            /SPACE IN BUFFER.
0031    0004    0000    BCOUNT, 0               /NUMBER OF FREE WORDS
0032                                            /LEFT IN BUFFER.
0033                    /
0034                    /
0035                    /       THIS IS THE BOOTSTRAP TO RESTART DIAL.
0036                    /
0037                            *15
0040    0015    0643    DIAL,   LDF     3       /SET DATA FIELD TO 3
0041    0016    0701            RCG             /READ IN DIAL
0042    0017    7300            7300            /8 BLOCKS FROM BLOCK 300
0043                    /
0044                    /
0045                    /       DIAL WILL START HERE.
0046                    /
0047                            *20
0050    0020    1020    START,  LDA I
0051    0021    0020            20              /I-O PRESET CODE
0052    0022    0004            ESF             /DO I-O PRESET
0053    0023    0061            SET I   BLOCKN  /INITIALIZE POINTER
0054    0024    0367            WKAREA=1        /TO WORKING AREA=1
0055    0025    0062            SET I   BLOCKC  /SET COUNTER TO WORKING
0056    0026    7676            =WKSIZE=1       /AREA SIZE+1 TO ALLOW
0057                                            /FOR RESET
0060    0027    6056    LOOP,   JMP     RESET   /START OF A NEW BLOCK
0061    0030    0460    INLOOP, SNS I   0       /SENSE SWITCH 0 NOT UP?
0062    0031    6042            JMP     DUMP    /IT IS UP.  DUMP BUFFER
0063    0032    0500            IOB
0064                            PMODE           /DO PDP-8 IOT
0065    4033    6031            KSF             /IS A CHARACTER READY?
0066                            LMODE
0067    0034    6030            JMP     INLOOP  /NO CHAR THERE.  WAIT
0070    0035    0500            IOB             /CHAR THERE
0071                            PMODE
```

```
0072    4036    6036            KRB             /READ IN THE CHAR
0073                            LMODE
0074    0037    1063            STA I   POINT   /PLACE IN BUFFER
0075    0040    0224            XSK I   BCOUNT  /BUFFER FULL?
0076    0041    6030            JMP     INLOOP  /NO.  GET ANOTHER WORD.
0077    0042    1000    DUMP,   LDA             /YES,  WRITE OUT BUFFER
0100    0043    0001            BLOCKN
0101    0044    1560            BCL I           /ONLY 9 BITS OF INTEREST
0102    0045    7000            =777            /AND WITH 777
0103    0046    1620            BSE I           /SET CORRECT FIELD BITS
0104                            PMODE           /FOR 12 BIT CALCULATION
0105    4047    1000            BUFFER+400&3000 /SETS CORRECT BITS ON
0106                            LMODE
0107    0050    4052            STC     .+2     /GET BLOCKN NUMBER AND PLACE
0110                                            /IN WRITE INSTRUCTION
0111    0051    0714            WRC     U       /WRITE OUT ON UNIT 1
0112    0052    0000            0000            /BLOCK NUMBER GOES HERE
0113    0053    0440            SNS     0       /SENSE SWITCH 0 UP?
0114    0054    6027            JMP     LOOP    /GO BACK AND GET SOME MORE
0115    0055    6015            JMP     DIAL    /YES,  LOAD IN DIAL.
0116                    /
0117                    /
0120    0056    0221    RESET,  XSK I   BLOCKN  /INCREMENT BLOCK NUMBER
0121    0057    0063            SET I   POINT   /SET BUFFER POINTER
0122    0060    4377            BUFFER-1
0123    0061    0064            SET I   BCOUNT  /SET BUFFER WORD COUNTER NOW
0124    0062    7377            =400            /400 OCTAL=256 DECIMAL
0125    0063    0222            XSK I   BLOCKC  /TOO MUCH DATA?
0126    0064    6000            JMP     0       /NO.  RETURN
0127    0065    1020            LDA I
0130    0066    7777            7777
0131    0067    0000            HLT             /YES,  HALT WITH 7777 IN AC
0132    0070    6065            JMP     .-3     /ONLY A RESTART PERMITTED NOW
0133                    /
0134                    /
0135                    /
0136                            WKSIZE=100      /WORKING AREA SIZE IS
0137                                            /100 BLOCKS
0140                            WKAREA=370      /WORKING AREA STARTS FROM
0141                                            /BLOCK 370
0142                            BUFFER=4400     /TAPE BUFFER AREA
0143                    /
0144                    /
0000  ERRORS
BCOUNT 4004
BLOCKC 4002
BLOCKN 4001
BUFFER 4400
DIAL   4015
DUMP   4042
INLOOP 4030
LOOP   4027
POINT  4003
RESET  4056
START  4020
WKAREA 0370
WKSIZE 0100
```

Appendix D
I/O Routines for DIAL-MS

D.1 CALLING THE ROUTINES

DIAL-MS provides a set of ready-to-go READ, WRITE, or MOVE I/O routines on tape 0 that may be called to perform block transfers on LINCtape or disks, thus eliminating the need to write or assemble these routines.

The DIAL-MS routines occupy blocks 322 and 323 of LINCtape 0 and may be loaded into locations 7000 through 7777 of any field. A typical sequence to load the routines into locations 7000-7777 of field 1 follows. As will be explained later, placing the routines in field 1 has a definite advantage over any other field.

```
LDF       7
RDC
6\322
RDC
7\323
```

Each of the three DIAL-MS routines, READ, WRITE, and MOVE, are called in PDP-8 mode and in the same manner. A typical call is:

```
CDF N                    /CURRENT IF
CIF M                    /SUBROUTINE
JMS ROUTINE
ARGUMENTS
```

The CDF instruction sets the data field to the present instruction field so the routines will know where to return. (If the data field is already set to the instruction field, this statement is not needed.) The CIF instruction sets the instruction field to the field of the routines. (If the routines are in the same field as the calling programming, this statement may be removed.) The JMS (or JMS I) is the call.

D.2 READ and WRITE

The READ and WRITE routines are called in the same manner, as:

```
JMS READ
POINTER
                         /RETURN IS HERE.
```

POINTER points to the following:

```
POINTER, UNIT NO
         CORELOC
         START
         BLOCKS
```

UNIT # is the logical unit number of the I/O device for the READ or WRITE. CORELOC is the first location of transfer divided by 400; thus 13 refers to location 5400. START is the starting block number of the transfer. BLOCKS is the number of blocks to transfer.

A program to read in eight blocks from block 30 on disk 0 into location 2000 is as follows:

```
        *5ØØ
        LINC
        LMODE
        LDF     7               /READ THE ROUTINES INTO SEGMENT 7
        RDC
        6\322
        RDC
        7\323
        CLR
        PDP
        PMODE
        CDF     Ø               /SAVE THE CURRENT IF FOR RETURN
        CIF     1Ø              /ROUTINES ARE IN SECOND PDP-8 FIELD
        JMS I PREAD
        P1
        HLT

P1,1Ø                           /UNIT NO.
    Ø4                          /CORE LOC.
    3Ø                          /TBLK
    1Ø                          /NO. BLOCKS
PREAD, READ
```

D.3 MOVE

The MOVE routine transfers core locations from one area of core to another, as:

```
        JMS MOVE
        CDF FROMF
        FROML
        CDF TOP
        TOL
        WORDS
                                /RETURN IS HERE
```

FROMF is the "from" field, FROML is the first "from" location, TOF is the "to" field, and TOL is the first "to" location. WORDS is the number of words to be transferred.

The MOVE entry point is at 7200, the READ is at 7774, and WRITE is at 7775. Remember that locations 7750 and 7751 of field 0 are the data break locations and should not be written over for a DF32 or RS08 disks (refer to Section 4.5). If data must be read from the disk into those locations, it is advisable to read the data into another location and then use the MOVE routine to transfer the information to 7750 or 7751.

D.4 BOOTSTRAP ROUTINE

By JMPing to 7777 in field 1, the bootstrap location, DIAL-MS is restarted. The bootstrap will not function properly unless the routines are in field 1, thus the advantage of having the I/O routines in field 1.

INDEX