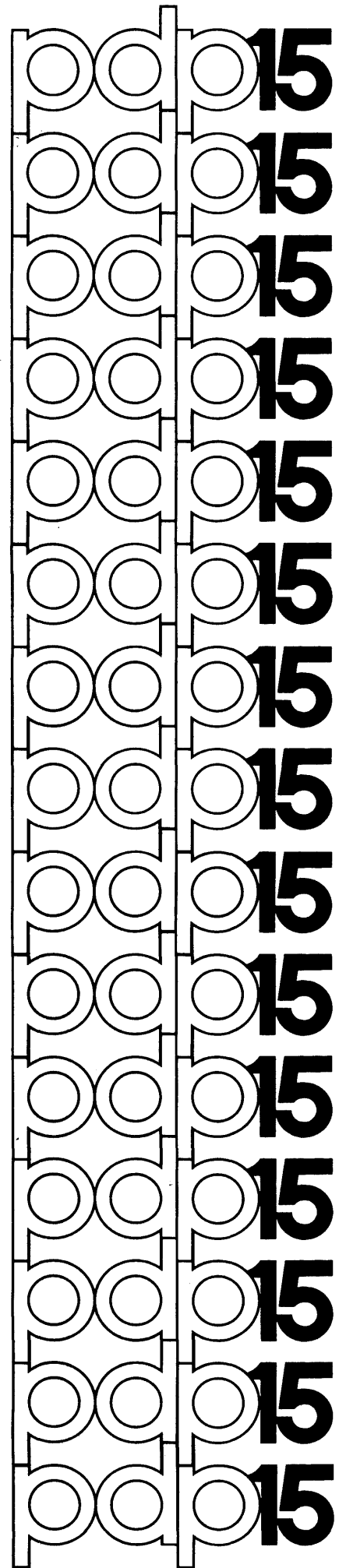


digital

# graphic15

## programming manual

digital equipment corporation



VT-15

GRAPHICS SOFTWARE  
SYSTEM  
PROGRAMMING MANUAL

For additional copies, order No. DEC-15-ZFSB-D from  
Digital Program Library, Maynard, Mass. Price \$3.50

DIGITAL EQUIPMENT CORPORATION . MAYNARD, MASSACHUSETTS

First Printing, November 1970  
Second Printing, October 1971

Copyright © 1971 by Digital Equipment Corporation

The information presented in this manual  
is for informational purposes only and is  
subject to change without notice.



The following are trademarks of Digital Equipment  
Corporation, Maynard, Massachusetts.

DEC

DIGITAL

PDP

## PREFACE

This manual describes the software provided for the VT15 Graphics Display Processor and its optional:

- a) VL04 Light Pen,
- b) LK35 Keyboard, and,
- c) VW01BP Writing Tablet and Control.

The information provided is applicable for users employing either the Disk Operating System (DOS) or Advanced Monitor Software System (ADSS).

It was assumed in the preparation of this manual that the user was familiar with the contents of the software manual describing the operating system (DOS or ADSS) being used. A list of applicable manuals is given in Section 1.

XXXXXXXX

## REFERENCE MANUALS

The user should be familiar with the following DEC documents.

PDP-15/20/30/40 ADVANCED Monitor Software System - ADVANCED Monitor System description. (DEC-15-MRZA-D)

UTILITY PROGRAMS MANUAL - Utility programs common to PDP-15 Monitor Systems. (DEC-15-YWZA-D)

MACRO-15 - MACRO assembly language for the PDP-15. (DEC-15-AMZA-D)

FORTRAN IV - PDP-15 version of the FORTRAN IV compiler language. (DEC-15-KFZA-D)

GRAPHIC-15 REFERENCE MANUAL - This Reference Manual describes the basic Graphic 15 processor and its interfacing arrangement with the PDP-15 computer. The information in this manual is intended to provide the Graphic 15 user with the knowledge necessary for programming at the machine-language level, to familiarize him with the operation of the basic Graphic 15 system, and to provide a brief description of the programming philosophy and principles of operation. (DEC-15-GWSA-D)

CONTENTS  
CHAPTER 1  
INTRODUCTION

Page  
1-1 & 1-2

CHAPTER 2  
SUBPICTURE ROUTINES

2.1	General Restrictions	2-3
2.2	LINE Subroutine	2-3
2.3	TEXT Subroutine	2-5
2.4	COPY Subroutine	2-6
2.5	PRAMTR Subroutine	2-7
2.6	GRAPH Subroutine	2-10
2.7	BLANK Subroutine	2-10
2.8	UNBLNK Subroutine	2-12
2.9	CIRCLE Subroutine	2-12
2.10	ROTATE Subroutine	2-13

CHAPTER 3  
MAIN DISPLAY FILE ROUTINES

3.1	DINIT (Display Initialize) Subroutine	3-3
3.2	DCLOSE (Display Terminate) Subroutine	3-4
3.3	SETPT (Set Point) Subroutine	3-4
3.4	PLOT Subroutine	3-5
3.4.1	PLOT A Subpicture (COPY)	3-5
3.4.2	PLOT a Line (LINE)	3-6
3.4.3	PLOT a Control Command (PRAMTR)	3-6
3.4.4	PLOT a Text String (TEXT)	3-7
3.5	DELETE Function	3-7
3.6	REPLOT Function	3-8
3.7	RSETPT Function	3-9

CHAPTER 4  
INPUT ROUTINES

4.1	LTORPB Function	4-1
4.2	TRACK Subroutine	4-2

CHAPTER 5  
RELOCATING ROUTINES

5.1	DYSET Subroutine	5-1
-----	------------------	-----

	<u>Page</u>
5.2      DYLINK Subroutine	5-2

## CHAPTER 6

### SYSTEM I/O DEVICE HANDLER

6.1      .INIT (Initialize) Macro	6-1
6.2      .READ Macro	6-3
6.3      .WRITE Macro	6-4
6.4      .WAIT Macro	6-4
6.5      .WAITR Macro	6-5
6.6      .CLOSE Macro	6-6
6.7      .FSTAT Macro	6-6
6.8      Ignored Functions	6-6

## CHAPTER 7

### LK35 KEYBOARD HANDLER

7.1      .INIT (Initialize) Macro	7-1
7.2      .READ Macro	7-2
7.3      .WAIT Macro	7-3
7.4      .WAITR Macro	7-3
7.5      .CLOSE Macro	7-4
7.6      .FSTAT Macro	7-4
7.7      Ignored Functions	7-4
7.8      Illegal Functions	7-5
7.9      Legal Control Characters	7-5

## CHAPTER 8

### VW01 WRITING TABLET HANDLER

8.1      .INIT (Initialize) Macro	8-2
8.2      .READ Macro	8-3
8.3      .WAIT Macro	8-4
8.4      .WAITR Macro	8-4
8.5      .FSTAT Macro	8-5
8.6      .CLOSE Macro	8-5
8.7      Ignored Functions	8-6

## CHAPTER 9

### TEXT DISPLAY/EDIT FUNCTIONS

9.1      EDITVT	9-1
9.1.1    Setup Commands	9-1
9.1.2    Controls	9-1

	<u>Page</u>
9.1.3. Display Modes	9-2
9.2. CONTROL X FEATURE	9-2
9.2.1. SCROLL Mode	9-3
9.2.2. PAGE Mode	9-3
9.2.3. TV ON/OFF Monitor Command	9-3
9.2.4. HALF ON/OFF Monitor Command	9-4
9.2.5. Differences Between CTRL X In V5A And In DOS	9-4

## APPENDIXES

- A. Mnemonics Commonly Used in Graphics Subprogram Calls
- B. Display Instruction Groups Generated by Graphics Subprogram Calls
- C. Macro Expansion of Graphics Subprogram Calls
- D. Conditional Assembly of Graphics Subprograms

## ILLUSTRATIONS

- Figure 2-1 Subpicture File Containing Four Vector Commands
- Figure 2-2 Sine Wave Program Example
- Figure 2-4 Operation of BLANK/UNBLNK Subroutine
- Figure 4-1 TRACK Program (FORTRAN Example)
- Figure 4-2 TRACK Program (MACRO 15 Example)
- Figure 5-1 DYSET/DYLINK Program Example

## TABLES

- Table 2.1 Display Parameter Settings
- Table 4.1 Description of CALL TRACK Arguments



## CHAPTER 1

### INTRODUCTION

This manual presents a detailed description of the PDP-15/VT15 Graphics Subprogram Package and is primarily concerned with those display subroutines and calling user programs used to exhibit information and communicate with the computer. The Graphics subprograms generate display commands that allow the user to define display elements and direct the linking, displaying, and deleting of those elements. Their primary purpose is to provide a simplified means of using the VT15 Graphic Display device without requiring detailed familiarity with the hardware.

In this manual, Graphic Routines are described in detail as follows:

- Chapter 2. Subpicture Routines
- 3. Main Display File Routines
- 4. Input Routines
- 5. Relocating Routines
- 6. System I/O Device Handler
- 7. LK35 Keyboard Handler (LKA)
- 8. VW01 Writing Tablet Handler (VWA)
- 9. Text Display/Edit Functions

Subprograms which consist of Graphic Routines mentioned above are called by user programs written in MACRO or FORTRAN IV language. The depth of coverage of these routines is intended to provide a very basic understanding of the use of the VT15 Graphic Display system. Much useful information may be found in appendices following Chapter 6.

The PDP-15 is designed with an autonomous systems structure and the VT15 follows this same philosophy; it operates asynchronously from the basic processor. Features include a cycle time of 750 nanoseconds, a character generator (with 64 printing characters and 4 control characters), a hardware program counter, a fast vector capability (1/4 inch in 1 usec), and a wide range of hardware options. The VT15 Graphics Software is designed for a minimum hardware configuration, as follows:

- PDP-15 with KSR-33
- 8K Core Memory (12K required for Text Editor)
- Paper Tape High-Speed Reader/Punch
- VT15 Display Processor
- VT04 Display Console
- Two DEctape Units

The Graphics Software consists of a group of routines that can be called by user programs. Calls to these routines build display files in a portion of PDP-15 memory that has been allocated by the calling program for such a purpose. The display files contain instructions and data upon which the VT15 Processor operates and to which its digital control and analog outputting circuits respond. The VT15 Processor has a set of 12 basic machine-language instructions which give it excellent versatility in the display of points, basic vectors, graphic plots, and ASCII characters. The commands contained in a main display file link together individual subpicture files causing the desired image to be displayed. Calls to other routines control the flow of the program upon the occurrence of light pen or push button interrupt. In this way, program paths can be enabled to modify the sequence of display commands and therefore modify the picture.

The VT15 Graphics Software is designed to run in Bank/Page Mode and to be used with either FORTRAN IV or MACRO-15 programs. FORTRAN IV programs devised by the user will consist of standard FORTRAN IV statements and calls to routines within the VT15 Graphics package. Other than system software normally used for compilations, assemblies, loading, etc., the VT15 Graphics software does not require use of any other programs.

The following manuals contain information useful in understanding and utilizing the contents of this manual.

### DOS Users

- |                     |               |
|---------------------|---------------|
| a) DOS Users Manual | DEC-15-MRDA-D |
|---------------------|---------------|

### ADSS Users

- |  |               |
|--|---------------|
| a. ADVANCED Monitor Software System for<br>PDP-15/20/30/40 Systems | DEC-15-MRZB-D |
|--|---------------|

### Common Manuals

- |   |               |
|---|---------------|
| a) MACRO15, Macro-Assembler Program                       | DEC-15-AMZC-D |
| b) FORTRAN IV Programmer Reference Manual<br>(8K Systems) | DEC-15-KFZB-D |
| c) Utility Program Manuals                                | DEC-15-YWZB-D |
| d) FORTRAN IV Language Manual<br>(16K Systems)            | DEC-15-GFWA-D |
| e) FORTRAN IV Operating Environment<br>(16K Systems)      |               |

### Hardware Manuals

- |                                |               |
|--------------------------------|---------------|
| a) Graphic-15 Reference Manual | DEC-15-GWSA-D |
| b) VW01 Writing Tablet, Vol. 1 | DEC-00-H4AA-D |

The GRAPHIC-15 Reference Manual is of particular importance to the VT15 programmer. This manual describes the basic Graphic 15 processor and its interfacing arrangement with the PDP-15 computer. The information in this manual provides the user with the data needed for machine level programming and familiarizes the user with the operation of the Graphic System.



## CHAPTER 2

### SUBPICTURE ROUTINES

These routines allow the user to incorporate point plotting, line drawing, and text display in his programs with minimum effort. Calls to these routines together with standard FORTRAN or MACRO statements build self-contained subpicture display files with executable display instructions. Each subpicture file contains all the display instructions needed to generate a specific image on the VT04 Display console. These files are accessed by a Main Display File (described in Chapter 3) in any order or sequence during the execution of the display program. Most Subpicture Routines will normally be called prior to initiating execution of a Main Display File, thus building a library of accessible graphics (i.e., complete or partial pictorial images) from which complex images may be formed. The subpicture display routines and their functions are:

- LINE - Draws a line (intensified) or moves the beam (not intensified) from current position. (Provides for using random vector option, if available.)
- TEXT - Displays strings of 5/7 ASCII text previously defined by the user in dimensioned arrays.
- COPY - Links subpicture files (similar to subroutining) to form a composite display image. Provides for using hardware SAVE/RESTORE feature, if desired.
- PRAMTR - Sets scale, intensity, light pen sensitivity, blink, etc., for this subpicture, or some portion thereof.
- GRAPH - Displays specified data points in graph form.
- BLANK - Inhibits display of any copy of this subpicture.
- UNBLNK - Reverses the action of the BLANK subroutines.

All display file storage is supplied by the FORTRAN user in the form of dimensioned integer arrays; MACRO-15 users must also allocate display file storage in some appropriate manner. To facilitate storage management, the first location of each file contains the length of the file. Limited reuse of storage is provided for in the Main Display File routines.

The first location of a subpicture file (PNAME) contains its current length - this value must be set to zero when the first reference to the subpicture display file is made. After the first reference, the contents of PNAME is set equal to the length of the subpicture file; this value is updated by any subsequent calls to the subpicture routines. (See figure 2-1.)

LOCATION	CONTENTS
PNAME	(6)
+1	return*
+2	vector command
+3	vector command
+4	vector command
+5	vector command
+6	DJMP* PNAME+1

\*Return address stored by any display JMS (DJMS) to this subpicture.

Figure 2-1. Subpicture File Containing Four Vector Commands

Since display files are generated and stored in arrays dimensioned by the user, they are fully accessible to the user and can be written out or read in using FORTRAN unformatted I/O statements.

Storage overhead for each subpicture display file is three words; the first word contains the file length, the second is used for a return address, and the third (last in file) contains the VT15 display command DJMP\* PNAME+1.

The procedure for generating a subpicture file such as that illustrated in Figure 2-1 requires some further explanation. The four calls to subroutine LINE, shown below, will result in such a file. This subpicture file will simply draw a square when accessed by the Main Display File or another subpicture file.

```

DIMENSION IPNAME(10)
IPNAME(1)=0
.
.
.
CALL LINE (100,0,1,IPNAME(1))
CALL LINE (0,100,1)
CALL LINE (-100,0,1)
CALL LINE (0,-100,1)
.
.
.
```

Note in the above example that storage allocation for the subpicture file was provided by the DIMENSION statement. Also, the first location, IPNAME(1), was set to zero before the first reference to it, thus indicating a new file. The identity of a subpicture file is the address of its first location (PNAME) and is given or implied, as an argument in all calls to subpicture routines. Each subpicture file is left in displayable form so that it can be manipulated dynamically

while being displayed.

Limited reuse of storage is provided for in the main display file routines RSETPT, REPLOT, and DELETE which are explained in Chapter 3. In this chapter, the number of locations required for display instructions generated by each subroutine call is indicated in each of the subroutine descriptions. Naturally, the total number of locations that can be allocated for display files is limited by the amount of core memory available.

## 2.1 GENERAL RESTRICTIONS

The following general restrictions apply to all subpicture routines except BLANK and UNBLNK.

- a. All arguments (constants or variables) must be of integer form.
- b. The variable PNAME must be set equal to zero before the first call referencing it.
- c. Sufficient space must follow PNAME to contain the subpicture file.

## 2.2 LINE SUBROUTINE

The LINE subroutine adds to the specified subpicture file the commands necessary to draw a line (beam intensified) or move the beam (not intensified) through a specified displacement from the current beam position.

The call statement has the form:

```
CALL LINE (DELTAX,DELTAY,INT[,PNAME])
```

where the enclosing brackets [ ] indicate an optional argument.

DELTAX represents the horizontal component of beam displacement in raster units and DELTAY represents the vertical component. The integer variable INT indicates whether the line is to be intensified (INT=1, the line will be visible; INT=0 the line will not be visible). The variable PNAME represents the first location of this subpicture display file and is the name by which the subpicture is referred to in later manipulation. For example, if a subpicture is to start in the dimensioned array ILEMNT, the form is:

```
CALL LINE(DELTAX,DELTAY,INT,ILEMNT(1))
```

Subroutine LINE adds one command to the display file if DELTAX and DELTAY define one of the eight basic directions:

```
VN!INCR    (where VN is vector direction n, and INCR is units)
            (the exclamation operator indicates an inclusive OR function)
```

If DELTAX and DELTAY do not define one of the eight basic directions, LINE tests for availability of the random vector option, and, if available, adds two commands to the display file:

SVX!DELTAX (stroke vector, x displacement)

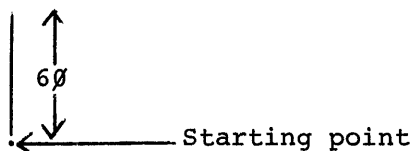
SVY!DELTAY (stroke vector, y displacement)

If not one of the eight basic directions, and if the random vector option is not available, LINE approximates the required line with a series of basic vectors. The number of commands added to the display file is added to the contents of location PNAME.

In addition to the general restrictions (paragraph 2.1) outlined previously for subpicture routines, there is another restriction that should be considered when using subroutine LINE: DELTAX and DELTAY should always be signed integers with magnitudes not exceeding 1023. The following two statements illustrate the use of the LINE subroutine.

CALL LINE (0,60,1,ILINE(1))

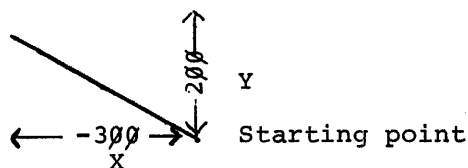
This statement generates a display instruction to draw a vertical line 60 raster units long. The display instruction (a basic vector) is stored at the end of subpicture file ILINE.



The following statement illustrates use of the LINE routine to draw a sloped line:

CALL LINE (IDX,IDY,1,ILINE(1))

where  $IDX=-300$  and  $IDY=200$ , we obtain the following:



Note that the random vector option is assumed to be available (otherwise, such a line would be approximated).

### 2.3 TEXT SUBROUTINE

The TEXT subroutine adds to the specified subpicture file commands necessary to display an identified text string - starting at the current beam position. The standard text font is drawn on a 10 by 14 dot matrix. Each character causes an increment of 14 raster units to the X position of the beam. The form is:

```
CALL TEXT(STR,N[,PNAME])
```

The input variable STR identifies the dimensioned real array that contains the string of characters to be displayed in IOPS ASCII (Hollerith) form - five 7-bit characters packed in two words. The variable, N, is an integer variable that indicates the number of characters to be displayed in the referenced array. If  $N \neq 0$ , an ALT MODE will be inserted after the  $n^{\text{th}}$  character to allow escape from the character mode. If  $N=0$ , ALT MODE will not be inserted in the TEXT array. The variable PNAME is the first location of this subpicture file, as in the call to LINE.

The TEXT subroutine adds three locations to the assembled display file; three is added to the contents of PNAME.

```
CHARS*   .+2
DJMP     .+2
(FULL 15-BIT ADDRESS)
```

#### NOTE

If 5/7 ASCII data is loaded into the array from an external source (as opposed to being defined in a FORTRAN DATA statement), it may contain certain non-printing characters (such as carriage return, line feed, etc.) that must be allowed for when specifying the argument N.

In addition to the general restrictions outlined in paragraph 2.1, the array referred to by TEXT must be of sufficient size to accommodate the escape character that will be inserted by TEXT. Also, to ensure that the display processor is conditioned to escape on ALT MODE, it is necessary to start up an empty Main File with a call to DINIT (described in Chapter 3). When this is done, a display parameter word is inserted in the new Main File to enable escape on ALT MODE only. (The alternative is to escape on carriage return or ALT MODE, whichever comes first; however, this option is not selectable using Main File or subpicture routines.)

The following example illustrates the manner in which TEXT to be displayed is set up and called:

Setup to display "15 ASSABET RD." is

```
DIMENSION ADDR(4)
DATA ADDR(1)/5H15 AS/,ADDR(2)/5HSABET/,ADDR(3)/4H RD./
```

The call statement to display the TEXT from subpicture IPIC is:

```
CALL TEXT(ADDR(1),14,IPIC(1))
```

## 2.4 COPY SUBROUTINE

The COPY subroutine enables two or more subpicture display files to be linked together to generate a composite display image. This is accomplished by a display subroutining technique. COPY adds to one subpicture display file the commands necessary to call a second subpicture. The second subpicture begins at the last beam position specified by the first subpicture. The form is:

```
CALL COPY(RST,PNAME1[,PNAME])
```

The variable, RST, indicates whether to save and restore display parameters<sup>1</sup> when copying the specified subpicture. RST may be set to 0 or 1; 0 indicates no SAVE/RESTORE option and 1 indicates SAVE/RESTORE option is to be used. The variable PNAME1 is the first location of the subpicture to be copied. PNAME is the first location of the subpicture file to which display instructions generated by this call are to be added.

The COPY subroutine adds three locations to the display file when the SAVE/RESTORE option is not specified. These three locations are as follows:

```
DJMS*      .+2
DJMP        .+2
            (ADDRESS of PNAME1+1)
```

However, when SAVE/RESTORE is specified, COPY adds six locations to the display file as follows:

```
SAVE        .+4
DJMS*        .+2
DJMP         .+3
            (ADDRESS of PNAME1+1)
            (STATUS)
RSTR         .-1
```

---

<sup>1</sup>These parameters include (but are not limited to) scale, intensity, blink, offset, and rotate, which can be set by calling subroutine PRAMTR (see paragraph 2.5.1) For a detailed description of parameters effected by the SAVE/RSTR instruction, refer to GRAPHIC-15 Reference Manual (DEC-15-GWSA-D).

where the SAVE instruction stores the effected display parameter settings in the STATUS word before executing the normal sequence of COPY commands. Upon returning from the subpicture, these parameters are restored to their original settings by the RSTR instruction. The contents of PNAME is increased by three or six, as required.

In addition to the general restrictions outlined in paragraph 2.1, PNAME 1 need not be defined when COPY is called but must be a defined subpicture when PNAME is displayed. The following statement:

```
CALL COPY(Ø,WINDOW(1),HOUSE(1))
```

adds a call to the window subpicture file to the file identified as HOUSE. Note that the SAVE/RESTORE option was not specified.

## 2.5 PRAMTR SUBROUTINE

The PRAMTR subroutine allows the user to add to the specified subpicture file the commands necessary to set up the following display features. (See DEC, Graphic-15 Reference Manual, PDP-15, for more detailed information.)

Scale setting - Setting the scale has a different effect, depending on where it is used. If used when plotting characters or vectors, it specifies the number of times (Ø - 15) that the unscaled vector (or stroke of a character) is to be repeated. If used in conjunction with the graph subroutine, the scale specifies the coordinate distance between given points.

Intensity Setting - The brightness of the display can be controlled in eight incremental steps between maximum dark and maximum light by specifying an integer variable or constant to represent the wanted brightness, between Ø and 7.

Light Pen Sensitivity - The ability of the light pen to sense a "hit" can be controlled by means of this feature.

Blink Setting - Use of this feature enables blinking of some portion or all of the displayed image. This feature causes characters as well as vectors to blink at a rate of approximately four times a second.

Dash Setting - This feature enables drawing of dashed lines and can be set from Ø to 3 as follows:

<u>Setting</u>	<u>Illuminated Raster Points</u>
Ø	ALL ON
1	3 ON 1 OFF
2	4 ON 2 OFF
3	4 ON 4 OFF

Offset Setting - Since the VT15 display processor defines a square drawing area, a standard rectangular tube would normally have some unused area. The VT15 makes

use of this area by means of the offset feature. When the offset is enabled, the absolute origin is relocated to the lower right-hand corner of the normal display area. This small area (approximately 9-1/2 x 1-1/2 in.) can be used for light buttons, special figures, etc., without disturbing the normal graphics area.

Rotate Setting - This feature allows the displayed image to be rotated 90 degrees in the counterclockwise direction or returned to its normal orientation if it is currently rotated. This could be useful for labeling graphs on the vertical axis or for any of a number of other applications.

Name Register Setting - The ability to set the Name Register is required to identify the location of light pen hits when using subroutine LTORPB. However, it is a feature which, when used at the programmer's discretion, can be helpful in many other applications. Once set, it retains its value until set to a different value.

Sync Feature - This feature can be used to avoid phosphor burnout when displaying files that require 32 milliseconds or less for execution. The display will halt and remain stopped until a sync pulse, derived from the local power main, enables execution to resume. This essentially locks execution of the display file to the power line frequency, which eliminates a visible swimming effect on the CRT.

By using the PRAMTR call statement, more than one feature (each with its corresponding settings) may be specified, using the following technique:

1. Add together the integer code numbers that identify the selected features and assign this value to the variable FEATR. For example: For scale (1) and Intensity (2), FEATR will have the value 3.
2. List the desired settings, as arguments, in ascending order according to the values of the numeric assigned to their corresponding features (the argument list 3,2,6 would specify a value of 2 for scale (feature 1) and of 6 for Intensity (feature 2)). The general call statement form is:
  - (a) One feature - CALL PRAMTR(FEATR,VALUE[,PNAME])
  - (b) More than one feature and setting -  
CALL PRAMTR(FEATR(S),VALUE1,VALUE2...[,PNAME])

The variable FEATR represents the display feature being set. The variable VALUE is the value to which FEATR is set. (See Table 2.1 for FEATR and VALUE settings.) PNAME is the first location of this subpicture file.

The PRAMTR subroutine adds from one to four commands to the display file, depend-

ing on the type of argument list used.<sup>1</sup> The number of commands added to the file is added to the contents of location PNAME.

Table 2.1 Display Parameter Settings

Parameter	Integer Code for FEATR	Possible Settings
Scale	1	Ø (Low) to 15 (High)
Intensity	2	Ø (Low) to 7 (High)
Light Pen	4	Ø (OFF) and 1 (ON)
Blink	8	Ø (OFF) and 1 (ON)
Dash	16	Ø (Solid) to 3 (Finest dash)
Offset	32	Ø (OFF) and 1 (ON)
Rotate	64	1 (CCW 90°) and Ø (Return CW 90°)
Name Reg.	128	Ø (Lowest) to 127 (Highest)
Sync	256	Ø (OFF) and 1 (ON)

Note: The abbreviation CCW = counterclockwise  
CW = clockwise

In addition to the general restrictions, the PRAMTR subroutine must be used with care, since the setting given is in effect until explicitly changed. Thus, if the blink is turned on at the beginning of a subpicture, it must be turned off at the end, otherwise the entire display image will blink (unless, of course, the SAVE/RESTORE option is used in calls to this subpicture).

The following single feature statement:

```
CALL PRAMTR (2,7,HOUSE(1))
```

specifies an intensity level of 7, for the subpicture display file starting at the first location of array HOUSE. The following multiple-feature statement:

```
CALL PRAMTR (SCALE+INT+LPEN,Ø,4,1,IN(1))
```

specifies the values Ø and 4 for scale and intensity, and turns on the light pen sensitivity. Appropriate display commands are added to the file that begins with

<sup>1</sup>Scale and intensity settings, when combined, generate only one display command. Light pen, blink, offset, and rotate, when combined, generate only one display command. Sync and dash features, when combined, generate only one display command. Setting the Name Register generates one command.

the first location of array IN.

## 2.6 GRAPH SUBROUTINE

The GRAPH subroutine adds to the specified subpicture file the commands necessary to display in graph form the identified set of data points. One coordinate is sequentially set to the value of each data point, the other coordinate is then automatically incremented (in the current scale), leaving the beam positioned one increment past the end of the graph. Note that axes and labeling must be provided separately. The call statement form is:

```
CALL GRAPH (DTA,N,A[,PNAME])
```

DTA represents an INTEGER array that contains the set of data points, one per word, in the range 0 to 1023. The variable N indicates the number of data points to be displayed. The variable A indicates which axis to increment, where A is set to either 0 or 1. (A=0 specifies incrementing the X axis and setting Y to data values; A=1 specifies incrementing the Y axis and setting X to data values.) The variable PNAME specifies the first location of the subpicture file to which the generated display commands are to be added.

The GRAPH subroutine adds to the subpicture file a number of graph-plot commands equal to the number of entries in the data set, as shown below. The number of commands added to the file is added to the contents of PNAME.

```
GX!VAL1      GY!VAL1
GX!VAL2      GY!VAL2
  .          .
  .          .
  .          .
GX!VALn      GY!VALn
or
GX!VALn      GY!VALn
```

One way to summarize the discussion up to this point is to review a program, (Figure 2-2 Sine Wave Program Example) which illustrates the use of GRAPH and other subroutines.

## 2.7 BLANK SUBROUTINE

The BLANK subroutine is used to prevent the displaying of any copy of the specified subpicture. However, the display file length is not changed. The form is:

```
CALL BLANK (PNAME)
```

where the variable PNAME is the subpicture to be blanked.

```

C
C  ARRAY INITIALIZATION
      INTEGER SINWV(300),Y(200)
      DIMENSION TITL(10),MAINFL(20)
      DATA TITL(1),TITL(2),TITL(3),TITL(4)/5HTHIS ,
      1 5HIS A ,5HSINE ,4HWAVE/
C
C  SET UP INTEGER ARRAY OF VALUES TO BE PLOTTED.
C
10      X=0
      DO 20 I=1,200
      Y(I)=IFIX(SIN(X)*256.)+512
      X=X+.0628
20      CONTINUE
C
C  SET UP SUBPICTURE TO PLOT THOSE VALUES.
C
      SINWV(1)=0
      CALL PRAMTR(3,0,7,SINWV(1))
      CALL LINE(1000,0,1)
      CALL LINE(-1000,0,0)
      CALL LINE (0,250,0)
      CALL LINE(0,-500,1)
      CALL LINE (0,250,0)
      CALL PRAMTR (1,4)
      CALL GRAPH (Y(1),100,0)
      CALL GRAPH (Y(101),100,0,SINWV(1))
C
C  SET UP MAIN FILE TO DISPLAY THE GRAPH.
C  (MAIN FILE CALLS BELOW, DESCRIBED IN CHPT. 3)
C
      MAINFL(1)=0
      CALL DINIT (MAINFL(1))
      CALL SETPT (10,512)
      CALL PLOT (0,0,SINWV(1))
      CALL SETPT (100,100)
      CALL PLOT (2,1,1)
      CALL PLOT (3,TITL(1),19)
      CALL DCLOSE
      STOP
      END

```

Figure 2-2. Sine Wave Program Example

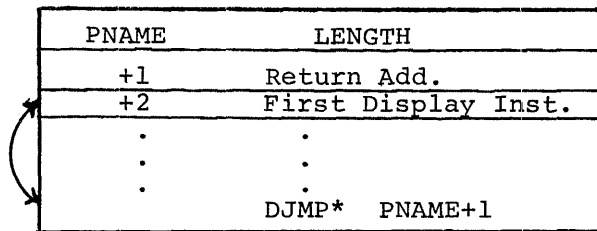


Figure 2-4. Operation of BLANK/UNBLANK Subroutine

In figure 2-4 the command in location PNAME+2 (the first executable command in the subpicture file) is interchanged with the DJMP\* PNAME+1 located at the end of the subpicture file. PNAME must be a defined subpicture file (BLANK has no meaning as the first call referring to PNAME). The subpicture files should not be modified while BLANKed. The following example would prevent the subpicture display file starting at the first location of array IPIC from being displayed.

```
CALL BLANK (IPIC(1))
```

## 2.8 UNBLNK SUBROUTINE

The UNBLNK subroutine reverses the action of the BLANK subroutine, allowing a previously BLANKed subpicture to be displayed. The form is,

```
CALL UNBLNK (PNAME)
```

where the variable PNAME is the subpicture to be UNBLNKed. The command in the last location of the subpicture file (placed there by a call to BLANK) is interchanged with the DJMP\* in location PNAME+2. If the referenced subpicture is not already BLANKed, UNBLNK will return without changing the file.

The following statement will enable the previously BLANKed subpicture IPIC to be displayed.

```
CALL UNBLNK (IPIC(1))
```

## 2.9 CIRCLE SUBROUTINE

The CIRCLE subroutine enables the user to construct approximations of arcs and circles as subpictures by specifying the length of a series of chords and the start and stop points of the arc or circle to be constructed.

The form of the FORTRAN call for the CIRCLE subroutine is:

```
CALL CIRCLE (R,THETA,GAMMA,DEG,PNAME)
```

where the call variables are defined as:

- 1) R, the radius, in raster units, of the circle to be constructed.
- 2) THETA, the start of a constructed arc expressed in degrees from the X-Axis, rotating counterclockwise about the center of the screen.
- 3) GAMMA, the end point of a constructed arc expressed in degrees rotating counterclockwise about the center of the screen.
- 4) DEG, the chord length expressed in degrees.
- 5) PNAME, the name of the location at which the CIRCLE subroutine will start the new subpicture array.

#### NOTES

- 1) Circles are only produced if  
GAMMA - THETA - 360°
- 2) The array at PNAME is as long as there are chords  
in the constructed arc or circle.

The MACRO form of the CIRCLE subroutine using the same variable representations as above is:

.GLOBL	CIRCLE
JMS	CIRCLE
JMP	+.7
.DSA	R
.DSA	THETA
.DSA	GAMMA
.DSA	DEG
.DSA	PNAME

#### NOTE

CIRCLE Subroutines require the arbitrary vector.

### 2.10 ROTATE SUBROUTINE

The ROTATE subroutine enables the user to plot three-dimensional figures from basic + wt.-dimensioned figures. Displayed items may be rotated about a specified axis through a designated angle of rotation. This subroutine achieves the rotation effect by modifying the users array.

A single call to the ROTATE subroutine can effect a rotation about one or more of the X, Y, or Z-axes. The rotation of a display about any other axis requires more than one call to be made to the subroutine.

The ROTATE subroutine utilizes the same left-handed system that is used throughout the graphics software, that is:

- a) X, horizontal movement, positive to the right;
- b) Y, vertical movement, positive is up;
- c) Z, axis into the display screen (positive movement)

The setpoint defines the origin of the axis of rotation.

#### CAUTION

The ROTATE subroutine should be used carefully, particularly when rotating large figures, or off-center origins.

If, during rotation, the end-point of a line of the rotating figure passes off screen, part or all of the figure may be lost. It is good practice in rotating large figures to save the original buffer before calling ROTATE.

The following restrictions must be observed:

- 1) The values in the user's array must be in floating point format.
- 2) The user must calculate the sine and cosine of the angle of rotation before he calls ROTATE.
- 3) The user must change integers into floating point numbers, and make the correct calls for displaying the rotated figure.

The FORTRAN and MACRO formats for calls to ROTATE are:

#### FORTRAN:

```
CALL ROTATE(ISTR,IA,IB,IC,X, Y, Z, SINA, CSA)
```

#### MACRO:

.GLOBL	ROTATE
JMS	ROTATE
JMP	+.12
.DSA	ISTR
.DSA	IA
.DSA	IB
.DSA	IC
.DSA	X
.DSA	Y
.DSA	Z
.DSA	SINA
.DSA	CSA

where the input variables are defined as:

1. ISTR, the array length,
2. IA, specifies whether rotation about the Z axis is desired  
If IA=1, rotation will occur about the Z axis.  
If IA=0, there will be no rotation about the X-axis
3. IB, specifies whether rotation about the Y-axis is desired.  
IB=1 indicates rotation is desired, as with IA.
4. IC, specifies whether rotation about the X-axis is desired.  
IC=1 indicates rotation is desired, as with IA.
5. X, the name of the X array.
6. Y, the name of the Y array
7. Z, the name of the Z array.
8. SINA, the sine of the angle of rotation.
9. CSA, the cosine of the angle of rotation.



## CHAPTER 3

### MAIN DISPLAY FILE ROUTINES

Calls to Main File routines together with standard FORTRAN IV statements will, when compiled, build a "Main Display File" in a portion of the PDP-15 memory that has been allocated by the calling program. The commands contained in this file link together individual subpicture display files causing the desired image to be displayed.

These routines are used to generate a Main Display File to which the display processor is directed when initiating display, and which is presumed to be calling upon the subpicture files generated with the routines described in Chapter 2. As is the case with subpicture files, storage used for the main file is supplied by the calling FORTRAN program as a dimensioned array. This array is identified by only one call to the initializing routine (DINIT) and is implicit in all other calls (which assume that reference is made to the storage identified by DINIT). These call statements are concerned, however, with the identification of each entry to the main display file. Thus most main file routines have as an optional argument the location of the display code generated by that particular call, which provides a "handle" to a particular graphic entity. This supplies the flexibility required to build and modify a display file in an interactive environment, and enables the user to perform limited storage management. The main display file routines and their functions are:

- DINIT - initializes and starts the display via device number  
(.DAT SLOT) 10
- DCLOSE - stops the display and leaves the main file in a form  
such that it can be called as a subpicture file.
- SETPT - sets absolute starting point of display. (Point not  
intensified.)
- PLOT - displays predefined but not necessarily complete sub-  
pictures, individual LINES, or ASCII text; also used  
to define display parameters.
- DELETE - deletes named subpicture file from main display file.
- REPLOT - similar to PLOT, but permits reuse of previously  
defined areas in the main file.
- RSETPT - similar to SETPT, but permits reuse of previously  
defined areas in the main file.

When returned from PLOT or SETPT, CNAME<sup>1</sup> contains a count of the instructions generated by that particular call in the high-order 3 bits. REPLOT and RSETPT use this count to determine whether the required number of locations is available. If there are not enough locations available, these routines check to see if there are enough contiguous locations containing display NOP's to satisfy the requirement. If the requirement is not satisfied, the function fails, an indicator to that effect is returned, and the display file is not changed. If the requirement is satisfied, the new group of instructions is inserted into the file along with enough display NOP's, if necessary, to match the size of the original group of instructions, and a logical success indicator is returned.

The DELETE function operates in a similar fashion, checking for a legal instruction count in the high-order 3 bits of CNAME. If the instruction count is zero, the function fails, an indicator to that effect is returned, and the display file is not changed. Otherwise, the number of instructions indicated (by the high-order 3 bits of CNAME) are replaced with display NOP's and a logical success indicator is returned.

An exception to this file management technique is when random direction lines must be approximated for calls to PLOT. In this case, the count returned in the high order bits of CNAME is set equal to 7 (this count is less than 7 for all other calls). The instructions for line approximation are added to the display file in the following format, where the actual count of instructions added to the file (plus 2) is stored in CNAME+1.

```

SKP
(COUNT=N+2)
V1
V2
.
.
.
VN

```

This difference in file management is invisible to the user since REPLOT and DELETE still operate the same externally.

Smaller groups of instructions can be packed into memory formerly required by a larger group. CNAME must be manipulated to accomplish this, and caution is advised in following this procedure.

As an example, assume that a previous call to PLOT has generated six instructions

---

<sup>1</sup>The optional output argument (CNAME) which is returned from PLOT and SETPT is a pointer to the display code generated by that call. It is a required input argument to subroutines REPLOT and RSETPT that permit reuse of locations in the main display file. It is also a required input to subroutine DELETE.

starting with the tenth location of the integer array IBUF. Also assume that CNAME was requested as an output argument of that call to PLOT, and is assigned to the variable IPLOT. When returned from the call to PLOT, IPLOT would point to IBUF(10), and the three high-order bits would equal six (the number of display commands inserted in the file).

If at a later time you wish to reuse these locations by inserting two successive calls to REPLOT (each of which generates three display commands), you can do so as follows:

```
CALL REPLOT (....,IPLOT)
IPLOT=(IPLOT*8/8)+3
CALL REPLOT (.....,IPLOT)
```

In the second statement of the above sequence, the portion in parentheses zeroes out the count in the high order three bits of IPLOT. (When this count is zero, REPLOT simply checks to see if enough DNOP'ed locations are available to satisfy the requirement.) Three is then added so that IPLOT now points to the first free location. It should be emphasized that simply adding three to the value of IPLOT is not satisfactory, since the count would not be valid for the second REPLOT (in fact, an additional three DNOP's would be added to the file to satisfy the original count of six). Therefore, it is imperative to know the exact number of locations available when using this technique, and to proceed with caution.

### 3.1 DINIT (DISPLAY INITIALIZE) SUBROUTINE

The DINIT subroutine initializes the display via device number (.DAT SLOT) 10. The VT15 device handler (VTA) must be associated with .DAT slot 10 as DINIT contains .IODEV 10, which causes the device handler associated with .DAT slot 10 to be loaded. DINIT can be used to set up for a new display main file, to start up an old one, or to start up any previously defined subpicture as the current main file. The call statement form is:

```
CALL DINIT (MAINFL(1))
```

MAINFL is the first location of the Main Display file. Like PNAME, it is an element of a dimensioned integer array. Location MAINFL contains the length of the Main Display File. This is updated by all main file routines.

Subroutine DINIT stores a DJMP\* MAINFL+1 at the end of the main file, inserts the address of MAINFL+2 into MAINFL+1, initializes the display, and starts the display running at MAINFL+2.

Certain restrictions must be noted when using DINIT. If a new display file is

being formed, location MAINFL must contain zero; if this is a previously defined file, location MAINFL contains the file length and must not be altered. Sufficient storage must follow MAINFL to accommodate the main display file that is to be generated. Only one main display file can be running at a time.

#### NOTE

When a new main display file is being initialized, DINIT inserts a display parameter word to turn off blink, offset, rotate and light pen, and to enable character string escape on ALT MODE (175<sub>8</sub>). To change the initial settings for blink, offset, rotate, and light pen, or to ensure that other display features (i.e., scale, intensity, dash, name register, and sync) are initially set as desired, the calling program should contain a PRAMTR type call to PLOT (described in paragraph 3.4.3) following the call to DINIT.

The following statement initializes the execution of the Main Display File starting at the first location of array MAINFL.

```
CALL DINIT (MAINFL(1))
```

### 3.2 DCLOSE (DISPLAY TERMINATE) SUBROUTINE

The DCLOSE subroutine is used to stop the display. DCLOSE also leaves the current main file in displayable form such that it can later be called as a sub-picture file or restarted as a main file. The call statement form is simply:

```
CALL DCLOSE
```

### 3.3 SETPT (SET POINT) SUBROUTINE

The SETPT subroutine is used to locate the beam on the display surface in absolute display coordinates (raster units). The beam is not intensified with this call. The call statement form is:

```
CALL SETPT (X,Y[,CNAME])
```

where the variable X represents the horizontal coordinate of beam location and Y represents the vertical coordinate of beam location. The variable CNAME is a pointer to the first location of the display commands generated by this call. SETPT adds two commands to the main file, as follows:

```
PY!Y  
PX!X
```

Two is added to the contents of location MAINFL. The location PY!Y is stored in CNAME (if given).

The variables X and Y must be positive integers and their values must not exceed 1023. A call to SETPT causes the beam to be given an absolute location, as opposed to a relative displacement. This action effectively severs any following parts of the display from any preceding parts; if a section of the display is completely defined in terms of relative vectors, then its location on the

display surface depends on where the beam was initially located, and it can be made to move as a unit by changing the initial setting. Giving the beam an absolute location disregards any previous motion and serves as a new reference point in the display.

CNAME is an optional output of this subroutine. Use of the same variable name as one used in a previous call will destroy the previous contents. The following statement establishes an absolute beam position with display coordinates  $X = 10$ ,  $Y = 10$ .

```
CALL SETPT (10,10)
```

### 3.4 PLOT SUBROUTINE

The PLOT subroutine is the prime active agent in the generation of the Main Display File. There are four forms of calls corresponding to the four subpicture routines, COPY, LINE, PRAMTR, and TEXT. These calls are used to display pre-defined (but not necessarily complete) subpictures and individual lines or text strings, and to introduce appropriate display control commands. In all cases, the requested display or control function is identified as a separate entity and may be manipulated independently of the rest of the display. The first entry in the argument list defines the type of call to PLOT as follows:

<u>FIRST ARG</u>	<u>TYPE OF PLOT</u>
0	COPY
1	LINE
2	PRAMTR
3	TEXT

#### 3.4.1 Plot a Subpicture (COPY)

The call statement form is:

```
CALL PLOT (0,RST,PNAME[,CNAME])
```

where the value 0 indicates this is a COPY type call to PLOT. RST is the indicator for the SAVE/RESTORE option (same as COPY). PNAME is the name (first location) of the subpicture to be displayed.

CNAME is an optional output argument that will contain a pointer to the first location of the group of display commands generated by this call. The number of commands added to the display file is added to the contents of MAINFL. In general, the same restrictions apply as for the COPY subroutine. Again, multiple use of the same variable CNAME will destroy previous contents. The following example illustrates use of a COPY type call to PLOT:

```
CALL PLOT (COPI,Ø,HOUSE(1),MAIN)
```

In this example, COPI has the integer value Ø; the next argument (Ø) is the indicator for the SAVE/RESTORE option; HOUSE identifies the subpicture file to be displayed; and MAIN is an optional output argument by which the group of display instructions inserted for this call may be referenced.

### 3.4.2 Plot a Line (or Reposition the Beam)

The call statement form is:

```
CALL PLOT (1,DELTAX,DELTAY,INT[,CNAME])
```

This type of PLOT is basically the same as the LINE subpicture routine, except for the first argument which defines this as a line type call to PLOT. The variable CNAME is an optional output argument and will contain a pointer to the first location of the group of display commands generated by this call. The number of commands added to the display file is added to the contents of MAINFL. The location of the first display command is stored in CNAME (if given).

As in SETPT, CNAME is an output variable and multiple use of the same variable name will destroy previous contents. Otherwise, the same general restrictions apply as for the LINE subpicture routine. The following example illustrates a LINE type call to PLOT.

```
CALL PLOT (LYNE,1ØØØ,1ØØ,ON,IEDGE(1))
```

where LYNE and ON have assigned values of 1 and IEDGE(1) is a display identifier to be used for later reference to this LINE.

### 3.4.3 Plot a Control Command (PRAMTR)

The call statement form is:

```
CALL PLOT (2,FEATR,VALUE[,CNAME])
```

where FEATR and VALUE must be specified in the same manner as for PRAMTR subpicture calls. Also, as with the PRAMTR call, multiple features can be specified in a single PLOT call of the following form:

```
CALL PLOT (2,FEATRs,VALUE1,VALUE2,...,VALUEn[,CNAME])
```

The number of commands added to the display file is added to the contents of MAINFL. The location of the first command is stored in CNAME (if given). The same general restrictions apply as for the PRAMTR subpicture routine. The

following example illustrates the use of this type of PLOT to set the BLINK feature in a Main File.

```
CALL PLOT (2,8,1)
```

The multiple-feature statement

```
CALL PLOT (PRAM,SCALE+INT+LPEN,Ø,4,1,IN)
```

establishes values Ø and 4 for display features SCALE and INT, and turns the light pen sensitivity on. The variable IN is supplied for the optional CNAME output argument. (PRAM=2, to specify a PRAMTR type call to PLOT.)

#### 3.4.4 Plot a Text String (TEXT)

The call statement form is:

```
CALL PLOT (3,STR,N[,CNAME])
```

This type of call to PLOT is essentially the same as that for the TEXT subpicture routine, except for the first argument which defines this as a TEXT type call to PLOT. The number of commands added to the display file is added to the contents of MAINFL. The location of the first generated display command is returned in CNAME (if given). The same restrictions apply as for the TEXT subroutine. The following example illustrates the use of the TEXT type call to PLOT

```
CALL PLOT (3,STRING,15,SAVNAM)
```

where STRING contains the 15 characters to be displayed, and SAVNAM will contain a pointer to the group of display commands inserted by this call.

#### 3.5 DELETE FUNCTION

The DELETE function is used to delete from the Main Display File any display entity formed by a single call to a main file routine and assigned to CNAME. If CNAME does not contain a legal instruction count (1-7), the DELETE fails and has no effect on the display file. The function and call statement forms are:

```
I = DELETE (CNAME)
```

or

```
CALL DELETE (CNAME)
```

The input variable CNAME is the location of the group of display commands to be

deleted. In the function form (I=), the output variable I is a Boolean success indicator; TRUE indicates a successful deletion, FALSE indicates an unsuccessful deletion. CNAME is checked to see if it contains a legal instruction count in high-order bits. If it does, all commands in this group are replaced by display NOP's (DNOPs); otherwise, no action is taken.

The example:

```
CALL DELETE (NAME(2))
```

deletes from the Main Display File the display entity whose first command is pointed at or identified by the second element of array NAME.

### 3.6 REPLOT FUNCTION

The function REPLOT allows use to be made of previously defined locations in the Main Display File. This can serve two purposes: (1) to reuse locations freed by DELETE, and (2) to change an existing group of display commands. REPLOT checks whether the group being inserted is longer than the space pointed at by CNAME, if it is, REPLOT then checks to see if there are enough DNOPed locations following the group to be overlaid. If there still are not sufficient locations available, the REPLOT fails and the display file is not affected. By manipulating CNAME, smaller groups can be packed into the space formerly used by a larger group. For example, up to three control commands could be inserted into the space left by a DELETED copy group. There are four forms of call to REPLOT, each of which is similar to the corresponding call to PLOT (Paragraph 3.4).

The first entry in the argument list defines the type of call to REPLOT as follows:

<u>FIRST ARG</u>	<u>TYPE OF REPLOT</u>
Ø	COPY
1	LINE
2	PRAMTR
3	TEXT

It is important to note that while CNAME is an optional output of PLOT it is a required input of REPLOT since it identifies the locations to be modified in the Main Display File. It also must be recognized that CNAME must have been given as an argument to a PLOT call for it to be available for REPLOT.

Since all of the REPLOT functions are similar to corresponding calls to PLOT, only the COPY type REPLOT is described as an example. The call statement forms for a COPY type REPLOT are:

```
I = REPLOT (Ø,RST,PNAME,CNAME)
```

or

```
CALL REPLOT (Ø,RST,PNAME,CNAME)
```

The input variables are the same as in the corresponding call to PLOT, except CNAME, which points to the first location of a block in which to store the display commands generated. The output variable I is a logical success indicator: TRUE indicates that the REPLOT was successful, and FALSE indicates that there was not enough room at the location pointed to by CNAME. It should be emphasized that if the above form is used, both I and REPLOT must be declared as LOGICAL in a type statement.

The COPY type REPLOT checks whether CNAME points to a large enough block of locations; no action is taken if the block is not large enough. Otherwise, REPLOT inserts the necessary commands starting at the location pointed to by CNAME, and inserts DNOP's in any remaining locations within the block. The same general restrictions apply as for the corresponding call to PLOT. The following example illustrates a COPY type call to REPLOT:

```
CALL REPLOT (Ø,IRST,SLIDE(M),NAME)
```

where Ø indicates that this is a COPY type call. IRST is equal to zero to indicate no SAVE/RESTORE option, M represents the first location of the sub-picture display file (in array SLIDE) and NAME identifies the first location in the display file into which this group of commands is to be inserted.

### 3.7 RSETPT FUNCTION

Like SETPT, the function RSETPT permits absolute beam locations to be defined; it can be used in the same manner as REPLOT to reuse any deleted locations or to change any existing group of commands. The same checking of needed space versus available space is done by RSETPT as in REPLOT.

The call statement forms are:

```
I = RSETPT (X,Y,CNAME)
```

or

```
CALL RSETPT (X,Y,CNAME)
```

The variable X represents the horizontal coordinate of beam location; Y repre-

sents the vertical coordinate of beam location. CNAME is an input argument that points to the first location of a block in which to store the display commands that are generated. If the function form (I=) is used with RSETPT, both I and RSETPT must be declared as LOGICAL in a type statement. RSETPT first checks whether CNAME points to a large-enough block of locations; no action is taken if the block is not large enough. Otherwise, RSETPT inserts two positioning commands at the location pointed to by CNAME:

```
PY!Y  
PX!X
```

RSETPT also inserts DNOPs in any remaining locations belonging to a former command group at this address. The following example illustrates the use of a call to RSETPT:

```
CALL RSETPT (lØ,lØ,NAME)
```

where the value of lØ is assigned to the X and Y coordinates and NAME identifies the starting location of a block within the display file into which the positioning commands are to be inserted.

## CHAPTER 4

### INPUT ROUTINES

Input routines enable the user (through his program) to deal with display console interaction using the light pen and pushbuttons. Routine LTORPB can inform the user whether there has been a light pen or pushbutton action and, if so, return the appropriate information that is required. The user program is not (logically) interrupted when such action occurs. The light pen or pushbutton action at the console merely causes an indicator to be set in the corresponding routine. This may affect the user's flow of control at his discretion. The light pen tracking routine (TRACK) provides a somewhat different use of the light pen, allowing the user to control input and generation of graphics.

#### 4.1 LTORPB FUNCTIONS

The function LTORPB is used to determine whether a light pen or pushbutton hit has occurred. If it has not, the function returns an indicator to this effect. If a hit has occurred, the logical (contents of name register) and physical (Y and X raster coordinates) location of the light pen and the status of the pushbutton box are returned as well as the indicator that a hit has occurred. For example, this routine may be used as a switch in a FORTRAN logical IF statement (see example below). The IF statement could branch to itself if no hit has occurred, or to the user's light pen hit processing code if a hit has occurred.

The function statement form is:

```
I = LTORPB (IX,IY,NAMR,PB,IWICH)
```

LTORPB and the variables I and PB must be declared logical in a type statement.

The output variable I is a logical success indicator; TRUE indicates that a light pen or pushbutton hit has occurred, and FALSE indicates no light pen hit has occurred. It should be emphasized that if I is FALSE, IX, IY, NAMR, and PB have no meaning.

The variable IX is the horizontal coordinate at end of the vector that caused a light pen hit. IY is the vertical coordinate at end of vector which caused a light pen hit. The variable NAMR will contain the value of the name register at the time of the light pen hit. PB should be defined in the calling program as a six-element array. Each element will contain either the logical TRUE or FALSE corresponding to ON or OFF for each of the six pushbuttons. IWICH will be either of two values; IWICH=1 if a light pen hit has occurred, or IWICH=2 if

a pushbutton hit has occurred.

LTORPB issues a .READ on light pen or pushbutton interrupt to the display device handler. It returns if no interrupt was posted. Otherwise, it reads appropriate display registers and returns with appropriate output variables.

The following statement illustrates use of LTORPB as a switch in a FORTRAN IF statement:

```
IF (LTORPB (LPX,LPY,NAME,PB,IWICH)) GOTO 100
```

In the above statement, if a hit has occurred (LTORPB is TRUE) LPX and LPY contain the X and Y coordinates of the end of the vector that was hit. The contents of the name register, the status of the pushbuttons and which hit it was are returned, and the program is directed to execute statement 100.

#### 4.2 TRACK SUBROUTINE

The TRACK subroutine is used for light pen tracking or drawing. Tracking being the input of an x-y coordinate where the tracking pattern is to be initially displayed. The user then moves the tracking pattern with the light pen to the desired location; and presses a pushbutton to escape from the tracking subroutine. The final x-y coordinate is then returned through the x-y input arguments. Drawing is the same as tracking with the addition of an array storing the vectors which describe the tracking path, (the path is displayed during tracking). The vector array is structured the same as a subpicture file and may be recalled by the user. The array must be dimensioned by the calling program, keeping in mind the number of vectors that will be inserted into it. A straight line requires an array of seven locations. The form of the call statement to track is as follows:

```
CALL TRACK (IX,IY,IOPT,IARRAY)
```

Use of the TRACK subroutine requires four arguments. Two already mentioned are the initial x coordinate and the initial y coordinate; these are also used as return arguments of the final x coordinate and final y coordinate. The third required argument is the direction option which allows the user to restrict movement of the tracking pattern in certain directions. The direction option is also used to draw straight lines in the horizontal or vertical direction. If the value of the direction is zero, tracking is allowed in any direction. A description of the direction options available can be found in Table 4.1 (Input Variables). The fourth required argument indicates whether the draw option is desired. A zero indicates tracking only. If drawing is desired, the argument will be the user's array location, where the intensified vectors

will be stored. Examples of the use of these arguments can be found in the following sample program that calls TRACK.

```
C   THE FOLLOWING FORTRAN PROGRAM USES THE TRACKING ROUTINE
C   TO DETERMINE THE DISTANCE BETWEEN (100,400), THE INITIAL
C   POSITION OF THE TRACKING PATTERN AND ANY POINT ON A LINE
C   OF SLOPE 20, DRAWN FROM A SET POINT AT X=750 Y=250
C
      DIMENSION MF(150),IUSER(200)
      MF(1)=0
      IOPT=3
      IX1=100
      IY1=400
C
C   INITIALIZE THE DISPLAY
C   CALL SET POINT TO POSITION BEAM
C   DRAW LINE FROM SET POINT
C   CALL TRACKING ROUTINE
C
      CALL DINIT(MF(1))
      CALL SETPT(750,250)
      CALL PLOT(1,25,500,1)
      IX2=IX1
      IY2=IY1
      CALL TRACK(IX1,IY1,IOPT,IUSER)
C
C   GET CHANGE IN X VALUE
C   GET CHANGE IN Y VALUE
C   CALCULATE DISTANCE BETWEEN POINTS
C
      IDELX=IX1-IX2
      IDELY=IY1-IY2
      IDELAB=SQRT((IDELX**2)+(IDELY**2))
      STOP
      END
```

Figure 4-1. Sample TRACK Program (FORTRAN Example)

Table 4.1 Description of CALL TRACK Arguments

Example:

CALL TRACK (IX,IY,IOPT,IARRAY)

INPUT VARIABLES:																	
IX	Initial Absolute X-Position of Tracking Point (Ø-1Ø23)																
IY	Initial Absolute Y-Position of Tracking Point (Ø-1Ø23)																
IOPT	Tracking Direction Option (Ø-6) <table border="1"> <thead> <tr> <th>OPTION</th><th>ALLOWABLE TRACKING DIRECTIONS</th></tr> </thead> <tbody> <tr> <td>Ø</td><td>ALL AXIS DIRECTIONS</td></tr> <tr> <td>1</td><td>+X -X</td></tr> <tr> <td>2</td><td>+Y -Y</td></tr> <tr> <td>3</td><td>+X +Y -Y</td></tr> <tr> <td>4</td><td>+X -X +Y</td></tr> <tr> <td>5</td><td>-X +Y -Y</td></tr> <tr> <td>6</td><td>+X -X -Y</td></tr> </tbody> </table>	OPTION	ALLOWABLE TRACKING DIRECTIONS	Ø	ALL AXIS DIRECTIONS	1	+X -X	2	+Y -Y	3	+X +Y -Y	4	+X -X +Y	5	-X +Y -Y	6	+X -X -Y
OPTION	ALLOWABLE TRACKING DIRECTIONS																
Ø	ALL AXIS DIRECTIONS																
1	+X -X																
2	+Y -Y																
3	+X +Y -Y																
4	+X -X +Y																
5	-X +Y -Y																
6	+X -X -Y																
IARRAY	- Tracking Draw Option <table border="1"> <thead> <tr> <th>USER DIMENSIONED ARRAY</th><th>DRAW OPTION DESIRED</th></tr> </thead> <tbody> <tr> <td>Ø</td><td>Intensified Vectors Following the Light Pen Movement Are Stored in This Array and are Displayed.</td></tr> <tr> <td></td><td>DRAW OPTION NOT DESIRED</td></tr> </tbody> </table>	USER DIMENSIONED ARRAY	DRAW OPTION DESIRED	Ø	Intensified Vectors Following the Light Pen Movement Are Stored in This Array and are Displayed.		DRAW OPTION NOT DESIRED										
USER DIMENSIONED ARRAY	DRAW OPTION DESIRED																
Ø	Intensified Vectors Following the Light Pen Movement Are Stored in This Array and are Displayed.																
	DRAW OPTION NOT DESIRED																

OUTPUT VARIABLES	
IX -	Final Absolute X - Position of Tracking Point (Ø-1Ø23)
IY -	Final Absolute Y - Position of Tracking Point (Ø-1Ø23)

When TRACK is called, the X and Y input arguments are inserted into the track display file. The track display file is then linked to the main file by inserting into main file a DJMS\* to a second location in the main file; into which has been inserted the address of the track display file. The direction option is then used to increment down a dispatch table which in turn sets up a second table so only light pen hits on certain sides of the tracking octagon are valid. The draw option is tested for and, if desired, the user's vector storage array is set up and linked to the track display file in the same manner that it was linked to the main file. Track then issues a .READ on Light Pen or Pushbutton interrupt, to the display device handler. If a light pen hit on a valid side of the octagon occurs, the tracking octagon is moved two raster units in the appropriate direction. If the draw option was specified, track adds a two raster unit vector to the user's vector storage array or increases the length of the last vector in the array if the hit was on the same side of the octagon as the previous hit. If a pushbutton interrupt occurs, TRACK removes all the created links and restores the main file to its previous form. The final X and Y coordinates of the tracking point are returned and control is released to the calling program.

The macro calling sequence to track is as follows:

```
.GLOBL    TRACK
JMS*      TRACK
JMP        .+5
.DSA      IX
.DSA      IY
.DSA      IOPT
.DSA      IARRAY
```

#### Internal Structures Created by Track:

Main File Link to Track:

```
MFTOP      LENGTH
           .+1
           DJMS*      .+2
           SKP
           ADDRESS
           .
           .
           .
           DJMS*      .+2           /Link to track display file
           DJMP*      MFTOP+1       /Jump to top of MAIN FILE
           TRCK        /Address of Track display file
```

Note: Track requires two temporary locations in the user's main file. A main file must be running when track is called.

Vector Array for Draw Option:

ARTOP	LENGTH	/File length
	Ø	/Return address
	PX	/X set point
	PY	/Y set point
	VI	/Intensified vectors
	VI	
	.	
	.	
	.	
DJMP*	ARTOP+1	/Display Jump to calling file

Note: The X and Y set points must be set by the user if he desires to relocate his array of intensified vectors, when recalling it.

(See MICRO 15 TRACK program page 4-7.)

```

/      THE FOLLOWING MACRO 15 PROGRAM USES THE TRACKING ROUTINE
/      TO LOCATE OR POSITION A SET POINT ON THE DISPLAY SCREEN.
/      THE POSITIONED SET POINT IS THEN USED TO DRAW A FIGURE.
/

SAMP2  .GLOBL  DINIT
      JMS*    DINIT          /INITIALIZE THE DISPLAY
      JMP     .+2
      .DSA    MAINBF

CHK0    LAC     (450          /SET INITIAL POSITION OF TRACKING PATTEN
      DAC     IX1
      DAC     IY1

CHK1    .GLOBL  TRACK
      JMS*    TRACK          /CALL TO TRACKING ROUTINE
      JMP     .+5
      .DSA    IX1            /X-POSITION
      .DSA    IY1            /Y-POSITION
      .DSA    CNST4          /DIRECTION OPTION
      .DSA    CNST4          /DRAW OPTION

CHK2    .GLOBL  SETPT
      JMS*    SETPT          /CALL TO SET POINT ROUTINE
      JMP     .+3
      .DSA    IX1            /X-POSITION RETURNED FROM TRACKING
      .DSA    IY1            /Y-POSITION RETURNED FROM TRACKING

CHK3    .GLOBL  PLOT
      JMS*    PLOT           /CALL TO PLOT ROUTINE
      JMP     .+5
      .DSA    CNST0          /ARG. TO PLOT A LINE
      .DSA    CNST1          /DELTA X
      .DSA    CNST1          /DELTA Y
      .DSA    CNST0          /INTENSIFY THE LINE

CHK4    JMS*    PLOT
      JMP     .+5
      .DSA    CNST0
      .DSA    CNST1
      .DSA    CNST2
      .DSA    CNST0

CHK5    JMS*    PLOT
      JMP     .+5
      .DSA    CNST0
      .DSA    CNST3
      .DSA    CNST4
      .DSA    CNST0
      HLT

MAINBF  .BLOCK  50          /DISPLAY MAIN FILE BUFFER
IX1     0
IY1     0
CNST0   1
CNST1   25
CNST2   -25
CNST3   -50
CNST4   0
      .END

```

Figure 4-2. Sample TRACK Program (MACRO 15 Example)



## CHAPTER 5

### RELOCATING ROUTINES

The subroutines DYSET and DYLINK are used to allow display main or subpicture files, which refer to each other (via COPY or PLOT (Ø....)), to be output and input relocatably. This includes arrays of 5/7 ASCII that are referred to via TEXT or PLOT(3....). Prior to outputting, interdependent display files and their user-assigned ASCII names are listed as arguments in a call to DYSET, which converts each subpicture call to the ASCII name of the subpicture being called. After input, and prior to displaying, a corresponding call is made to DYLINK, which uses the listed ASCII names to reinstate the appropriate subpicture calls or text references. A display file cannot be displayed after having been processed by DYSET; DYLINK must be used to return it to displayable form.

#### 5.1 DYSET SUBROUTINE

The DYSET subroutine converts subpicture calls or text references to a symbolic form independent of core memory location, using specified ASCII strings. The forms are,

```
CALL DYSET (PNAME1,ASCII1,...,PNAMEN,ASCIIIN)
```

or

```
CALL DYSET (PNAME1,ASCII1,...,PNAMEK,ASCIIK,Ø,PNAMEL,ASCIIIL,...  
            ,PNAMEN,ASCIIIN)
```

The variable PNAMEs are the first locations of the interdependent display files, both calling and called. If a Ø argument appears in the argument string, subsequent PNAMEs refer to arrays of 5/7 ASCII text. (These files will not be searched for memory references.) The ASCIIIs are the names of real arrays containing nine characters of 5/7 IOPS ASCII, which may be used for filenames on output.

Subroutine DYSET searches each listed display file (PNAME) for a DJMS or CHARS instruction. When it finds one, it appends the ASCII name of the file referenced to the file being searched, if that name is not already there. The operand of the DJMS is made a relative pointer to the ASCII name of the referenced file. The first location of the file being searched is increased by four each time an ASCII name is appended to the file.

Certain restrictions must be noted; space provided for a display file must include four locations for each subpicture that is called. Display commands must not be added to a display file nor can a file be displayed once it has been processed by DYSET, or until after it has been processed by DYLINK. (Thus DYSET must be called

after DCLOSE for a main display file.) Also, it is the user's responsibility to list all relevant display files when calling DYSET. The subroutine does not check the list for completeness in order to allow multiple calls to it. Once a zero appears in the argument string, all subsequent PNAMEs must refer to arrays of 5/7 ASCII text.

## 5.2 DYLINK SUBROUTINE

The DYLINK subroutine converts file names to appropriate DJMS or CHARS instruction references to the corresponding files. The forms are:

```
CALL DYLINK(PNAME1,ASCII1,...,PNAMEN,ASCIIIN)
```

or

```
CALL DYLINK(PNAME1,ASCII1,...,PNAMEK,ASCIK,0,PNAMEL,  
            ASCII1,...,PNAMEN,ASCIIIN)
```

where the input variables are the same as for DYSET. DYLINK searches each listed display file for a DJMS or CHARS instruction. When it finds one, it searches the argument list for a pointer to an ASCII string equal to the one pointed at by the operand of the DJMS or CHARS instruction. This operand is replaced by the address of the corresponding file, obtained from the argument list. The first location of each display file that is searched is reduced to the actual number of display commands in the file (excluding the ASCII blocks).

It is the user's responsibility to list all relevant display files when calling DYLINK. The subroutine does not check the argument list for completeness, to allow multiple calls. Once a zero appears in the argument string, all subsequent PNAMEs must refer to arrays of 5/7 ASCII text. See Figure 5-1 for DYSET/DYLINK Program.

```

C
C  ARRAY INITIALIZATION
C
      DIMENSION NWPIC0(40), NWPIC1(20), NWPIC2(20)
      DIMENSION RTXTA(2), RTXTB(2)
      DIMENSION IMAIN(40), IPICA(20), IPICB(20)
      DIMENSION TEXTA(2), TEXTB(2)
      DIMENSION TITL1(2), TITL2(2), TITL3(2), TITL4(2), TITL5(2)
      DATA TITL1(1), TITL1(2)/5HJPIC0, 4H BIN/,
      2TITL2(1), TITL2(2)/5HJPICA, 4H BIN/,
      3TITL3(1), TITL3(2)/5HJPICB, 4H BIN/,
      4TITL4(1), TITL4(2)/5HCHRSA, 4H BIN/,
      5TITL5(1), TITL5(2)/5HCHRSB, 4H BIN/
      DATA TEXTA(1), TEXTA(2)/5HI AM ,4HBOXA/,
      1TEXTB(1), TEXTB(2)/5HI AM , 4HBOXB/

C
C  INITIALIZE DISPLAY FILES.
C
      IMAIN(1)=0
      IPICA(1)=0
      IPICB(1)=0

C
C  BUILD BOXB (IPICB)
C
      CALL TEXT (TEXTB(1),9, IPICB(1))
      CALL LINE (100,0,1)
      CALL LINE (0,100,1)
      CALL LINE (-100,0,1)
      CALL LINE (0,-100,1)

C
C  BUILD BOXA (IPICA)
C
      CALL LINE (300,0,1,IPICA(1))
      CALL LINE (0,300,1)
      CALL LINE (-300,0,1)
      CALL LINE (0,-300,1)
      CALL LINE (30,30,0)
      CALL COPY (0,IPICB(1))

C
C  BUILD MAIN (IMAIN)
C
      CALL DINIT (IMAIN(1))
      CALL PLOT (2,19,0,4,0)
      CALL SETPT (20,20)
      CALL PLOT (3,TEXTA(1),9)
      CALL PLOT (0,1,IPICA(1))
      CALL SETPT (534,20)
      CALL PLOT (3,TEXTA(1),9)
      CALL PLOT (0,0,IPICA(1))

C
C  DCLOSE, CALL DYSET, AND OUTPUT TO DECTAPE (DAT 5)
C
      CALL DCLOSE
      CALL DYSET (IMAIN(1),TITL1,IPICA(1),TITL2,IPICB(1),TITL3,0,
      1TEXTA(1),TITL4,TEXTB(1),TITL5)
      CALL ENTER (5,TITL1)
      J=IMAIN(1)+1
      WRITE (5) (IMAIN(I), I=1,J)
      CALL CLOSE (5,TITL1)

```

Figure 5-1. DYSET/DYLINK Program Example

```

CALL ENTER (5,TITL2)
J=IPICA(1)+1
WRITE (5) (IPICA(1), I=1,J)
CALL CLOSE (5,TITL2)
C
C
CALL ENTER (5,TITL3)
J=IPICB(1)+1
WRITE (5) (IPICB(1), I=1,J)
CALL CLOSE (5,TITL3)
C
C
CALL ENTER (5,TITL4)
WRITE (5) (TEXTA)
CALL CLOSE (5,TITL4)
C
C
CALL ENTER (5,TITL5)
WRITE (5) (TEXTB)
CALL CLOSE (5,TITL5)
PAUSE 222
C
C INPUT FROM DECTAPE, CALL DYLINK AND DINIT
C
CALL SEEK (5,TITL1)
READ (5) J, (NWPIC0(I+1), I=1,J)
NWPIC0(1)=J
CALL CLOSE (5,TITL1)
C
C
CALL SEEK (5,TITL2)
READ (5) J, (NWPIC1(I+1), I=1,J)
NWPIC1(1)=J
CALL CLOSE (5,TITL2)
C
C
CALL SEEK (5,TITL3)
READ (5) J, (NWPIC2(I+1), I=1,J)
NWPIC2(1)=J
CALL CLOSE (5,TITL3)
C
C
CALL SEEK (5,TITL4)
READ (5) RTXTA
CALL CLOSE (5,TITL4)
C
C
CALL SEEK (5,TITL5)
READ (5) RTXTB
CALL CLOSE (5,TITL5)
C
C
CALL DYLINK (NWPIC0(1),TITL1,NWPIC1(1),TITL2,NWPIC2(1),TITL3,0,
1RTXTA(1),TITL4,RTXTB(1),TITL5)
CALL DINIT (NWPIC0(1))
STOP
END

```

## CHAPTER 6

### SYSTEM I/O DEVICE HANDLER

The VT15 Graphic Display Device Handler provides an interface between the user and the hardware. In general, it conforms to the conventions of the Keyboard Monitor System, as described in DEC manual Monitors, ADVANCED Software System. Input or output functions are initiated by standard user program commands and all display interrupt management is done automatically by the handler. The primary goals of the device handler are to relieve the user from writing his own device handling subprograms and to centralize all direct communication between the PDP-15 and the display processor. To start up a display, the user generates a display file consisting of display commands then calls the device handler to start it running. To interact with it, the device handler is used to read display controller registers and to dispatch on appropriate interrupts.

#### 6.1 .INIT (INITIALIZE) MACRO

The macro .INIT causes the display to be initialized and must be given before any other I/O macro to the display is issued. The display is initialized according to four words of standard settings contained in the handler. The user may substitute his own settings for any of these.

The Device Handler is connected to the Monitor Interrupt system (PIC or API) in the same manner as other system device handlers.

The form is:

```
.INIT  A, F, R
```

A = Device Assignment Table (.DAT) slot number  
F = initialization flag  
     $\emptyset$  use standard display initialization  
    1 user's initialization is pointed to by R  
R = optional pointer to user's initialization settings  
If F = 1, R points to a word containing initial settings.  
If F =  $\emptyset$  and R = 1, clearing the READ BUSY switch is the only action taken by the handler.

The expansion is:

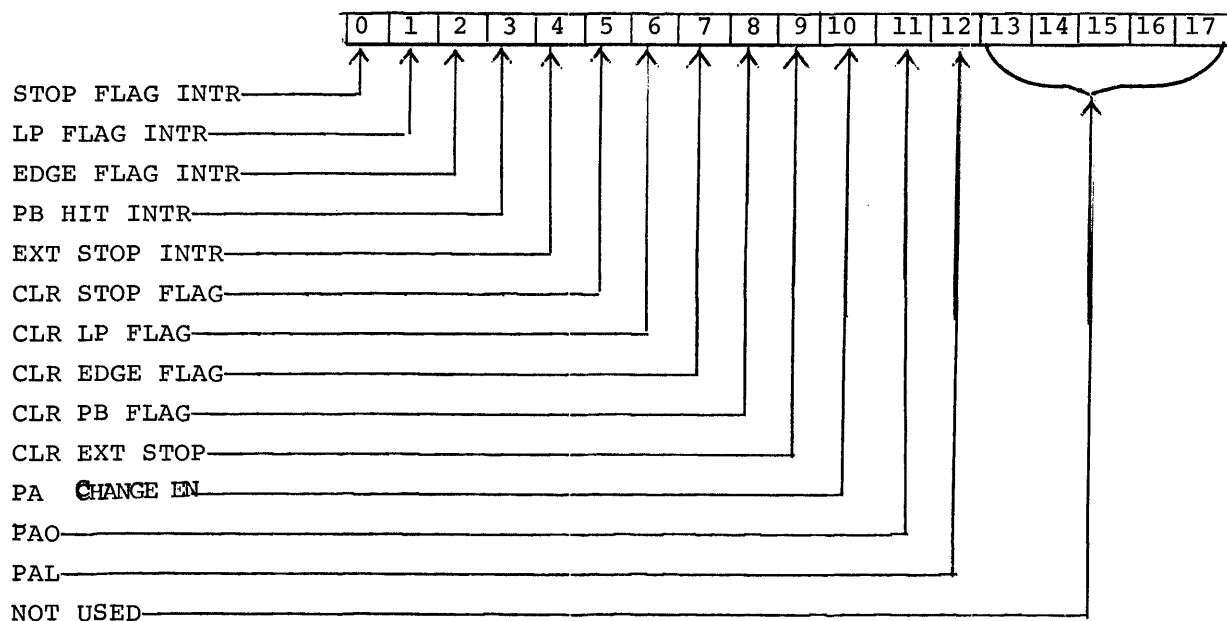
```
LOC      CAL + F(7-8) + A(9-17)
LOC+1    1
LOC+2    R
```

The normal settings are:

- a. Set display status to
  1. DISABLE edge flag interrupts
  2. ENABLE light pen interrupts
  3. ENABLE pushbutton interrupts
  4. ENABLE external stop interrupts
  5. ENABLE full 12 Bit X and Y beam position registers
  6. ENABLE internal stop interrupts
- b. Connect handler to PIC or API
- c. Clear READ BUSY switch

### Initialization IOT

SIC (703024) Set Initial Conditions - SIC sets up a number of status registers in the display. The instruction enables five display flags onto the Interrupt Line. The IOT is issued with settings loaded in the AC in the following format:



- 0 Sets the Stop Flag Interrupt Enable Flop
- 1 Sets the LP Flag Interrupt Enable Flop
- 2 Sets Edge Flag Interrupt Enable Flop
- 3 Sets PB Hit Interrupt Enable Flop
- 4 Sets External Stop Interrupt Enable Flop
- 5 Clears Stop Flag
- 6 Clears LP Flag
- 7 Clears Edge Flag
- 8 Clears PB Flag
- 9 Clears External Stop Flag
- 10 Allow a Change in Virtual Paper Size
- 11 New Virtual Paper Size
- 12 New Virtual Paper Size

### Bits 11 & 12 (New paper Size)

- 00 = 9.5 inch (10 bits)  
1024 raster units
- 01 = 19 inch  
2048 raster units
- 10 = 28.4 inch  
3072 raster units
- 11 = 38 inch (12 bits)  
4096 raster units

## 6.2 .READ MACRO

The .READ macro is used for input to the user program from the hardware registers of the display controller. The user may select standard groups of registers to be read, in response to each possible display interrupt flag, or he may indicate his own group of flags and registers. This is done with an optional descriptive word following the .READ macro, the first five bits of that word indicate which interrupts are of interest and the next nine indicate the registers to read if any of those interrupts are set.

The form is,

```
.READ A, M, L, W
NSTD
```

where NSTD = optional word describing non-standard groups.

The variables A = .DAT slot number, M = type of read:

Ø = READ,PB,XP,YP,S1,S2	Read now, no interrupts
1 = READ,PB,XP,YP,DPC,S1,S2,NR	If stop flag interrupt flag is set
2 = READ,PB,XP,YP,DPC,S1,S2,NR	If pushbutton interrupt flag is set
3 = READ,PB,XP,YP,DPC,S1,S2,NR	If light pen interrupt flag is set
4 = READ,PB,XP,YP,DPC,S1,S2,NR	If edge flag interrupt flag is set
5 = READ,PB,XP,YP,DPC,S1,S2,NR	If external stop interrupt flag is set
7 = NSTD	specifies registers and interrupt flags as follows:

- Bit Ø on service internal stop interrupt
- Bit 1 on service pushbutton interrupt
- Bit 2 on service light pen interrupt
- Bit 3 on service edge flag interrupt
- Bit 4 on service external stop interrupt
- Bit 5 on read pushbuttons (PB)
- Bit 6 on READ X position register (XP)
- Bit 7 on READ Y position register (YP)
- Bit 8 on READ DISPLAY program counter (DPC)
- Bit 9 on READ STATUS one (S1)
- Bit 10 on READ STATUS TWO (S2)
- Bit 11 on READ NAME REGISTER (NR)
- Bit 12 on READ SLAVE GROUP 1 (SG1)
- Bit 13 on READ SLAVE GROUP 2 (SG2)

L = return buffer address, C(1) = descriptive word showing what this interrupt was and which registers were read in the order listed above. C(L+1) = contents of first register actually read, C(L+2) = contents of second register read, etc.  
W = 1 (W must equal 1).

The expansion is:

```
LOC      CAL + M(6-8) + A(9-17)
LOC+1    1Ø
LOC+2    L
LOC+3    -W      /DECIMAL
LOC+4    NSTD
```

.READ determines interrupts to be serviced and turns on read busy flag.

### 6.3 .WRITE MACRO

The .WRITE macro is used to transmit information from the user program to the display controller, once a display file has been generated. Its location is passed on to the display controller by a call to .WRITE, and the display starts up.

.WRITE is also used to stop the display, by issuing an external stop, and to start the display if it has been stopped. A .WRITE to the display is done immediately and requires no waiting.

The form is,

```
.WRITE A, M, L, W
```

A = .DAT slot number

M = type of write,

where Ø = restart display (L not required)

1 = resume display after internal stop

Note: The display is automatically resumed after  
LP or EDGE violation interrupt.

2 = stop display (external stop)

4 = start display pointed to by L

L = display file starting address

W = not used

The expansion is:

```
LOC      CAL + M(6-8) + A(9-17)
LOC+1    11
LOC+2    L
LOC+3    .DEC
LOC+3    -W      /DECIMAL
```

### 6.4 .WAIT MACRO

The .WAIT macro is used to synchronize the user program with the interrupt activity of the display. .WAIT is only defined with respect to .READ. If a .WAIT is given, the user program waits until the previous .READ has completed,

that is, the interrupt has occurred. If the previous .READ specified more than one kind of interrupt flag, the descriptive word(s) in the input buffer can be interrogated to determine what flags were set. .WAIT does not initiate any display activity.

The form is,

```
.WAIT A
```

The variable A = .DAT slot number.

The expansion is,

```
LOC      CAL + A(9-17)
LOC+1    12
```

.WAIT allows a previous .READ to be completed and turns off input busy flag.

#### 6.5 .WAITR MACRO

The .WAITR macro allows the user program to proceed in line if the previous .READ is complete. If the previous .READ is not complete, control is given to the location in the user program specified by the .WAITR call. This allows the user to branch to some other part of his program while waiting for the .READ to finish. The user must continue to check for completion by periodically issuing .WAITRs or by issuing a .WAIT.

The form is,

```
.WAITR A, ADDR
```

The variables A = .DAT slot number, and ADDR = location in the user program to branch to if input is not completed.

The expansion is,

```
LOC      CAL+1000 + A(9-17)
LOC+1    12
LOC+2    ADDR
```

#### 6.6 .CLOSE MACRO

The .CLOSE macro is used to terminate the current display. External STOP and CLEAR flags IOTs are issued. It is up to the user to save the display file if desired.

The form is .CLOSE A where A = .DAT slot number.

The expansion is,

LOC	CAL + A(9-17)
LOC+1	6

#### 6.7 .FSTAT MACRO

The .FSTAT macro checks the status of a file specified by the file entry block. On return, the AC will contain zero and bits 0-2 of LOC+2 will also be zero, stating that the device was non-file oriented.

The form is,

.FSTAT A, D

where the variables A = .DAT slot number, and D = starting address of three word block of storage in user area containing the file name and extension of the file name whose presence on the device associated with .DAT slot A is to be examined.

The expansion is,

LOC	CAL+3000 + A(9-17)
LOC+1	2
LOC+2	D

#### 6.8 IGNORED FUNCTIONS

The following system I/O macros are ignored by the VT15 display device handler:

1. .DELETE
2. .RENAM
3. .ENTER
4. .CLEAR
5. .MTAPE
6. .SEEK
7. .TRAN

## CHAPTER 7

### LK35 KEYBOARD HANDLER

The LK35 Keyboard device handler (LKA) provides an interface between the user and the hardware. In general, the handler, LKA, conforms to the conventions of the DOS and ADSS monitor software systems. Since the LK35 is a send-only device, the LKA handler provides only input functions. Input functions are initiated by standard user program commands; all interrupt management is done automatically by LKA.

The LKA handler relieves the user of the task of writing his own device handling subprograms and centralizes all direct communications between the PDP-15 computer and the LK35 Keyboard. This handler only inputs IOPS ASCII or IMAGE ASCII data into a user-designated buffer; it is up to the user to develop the display of any input text on the VT04 display CRT or output it to any other device. The LK35 Keyboard is connected to either an LT15 or LT19D controller.

The LKA handler is a resident program, it resides with the Keyboard Monitor and other required device handlers. It does not require EAE and it operates with both PI and API.

#### 7.1 .INIT (Initialize) MACRO

This macro initializes the LK35 Keyboard; it must be called before any other I/O macro is issued to this device.

When .INIT is issued it initializes the LKA handler, which returns the size of the current line buffer (34<sub>10</sub> standard) to the macro.

If .INIT is issued during a .READ, it will abort this operation.

The form of this macro is:

```
.INIT a,f,r
```

where:

```
a = .DAT slot number
f = ignored by LKA
r = control p address
```

The expansion of this macro is

```
LOC      CAL+f7-8+a9-17
LOC+1    1
LOC+2    r
LOC+3    n      (standard buffer size 3410)
```

## 7.2 .READ MACRO

This macro performs the operations required to input data from the LK35 Keyboard and transfer it to the memory input line buffer. In performing this function, the .READ macro:

- a) allows any previous input operation to terminate,
- b) sets the "input underway" indicator,
- c) accepts and performs the operations indicated by:
  - 1) RUBOUT - delete previously entered (typed) character,
  - 2) CTRL U (↑U) - delete all entries made prior to ↑U.
- d) recognizes IOPS ASCII string terminators ALT MODE and RETURN (carriage return ).
- e) is terminated, during IMAGE ASCII read operations when the given line buffer word count (see form) is reached.

The form of the .READ macro is:

```
.READ a,M,L,W
```

where:

```
a      = .DAT slot number
M      = Data Mode
        2 = IOPS ASCII
        3 = IMAGE ASCII
L      = Line buffer address
W      = Line buffer word count (including 2-word header pair)
```

The expansions of this macro are:

```
LOC      CAL+M6-8+a9-17
LOC+1    10
LOC+2    L
LOC+3    -W
```

### 7.3 .WAIT MACRO

The .WAIT macro is used to detect the availability of the user's line buffer for data transfer operations. If the buffer is unavailable when tested, control remains with the macro; if the buffer is available, control is returned to the user.

The form of this macro is:

.WAIT a

where a represents a .DAT slot number

The expansion of the macro is:

```
LOC      CAL+a9-17
LOC+1    12
```

### 7.4 .WAITR Macro

This macro enables the user to test the status of a previously initiated .READ operation. If the .READ operation is complete the user's program is permitted to proceed in line; if the .READ operation is not complete control is given to a user-specified location expressed in the .WAITR macro call. The latter feature permits the user to branch to some other part of his program while waiting for the completion of the .READ operation.

The form of this macro is:

.WAITR a,ADDR

where:

```
a      = .DAT slot number
ADDR   = location to branch to if .READ operation is incomplete.
```

The expansion of this macro is:

```
LOC      CAL+1000+A9-17
LOC+1    12
LOC+2    ADDR
```

### 7.5 .CLOSE MACRO

The LKA handler regards the .CLOSE macro as being the same as the .WAIT macro (see 7.3).

The form of this macro is:

.CLOSE a

where A = .DAT slot number.

The expansion of this macro is:

```
LOC      CAL+a9-17
LOC+1    6
```

### 7.6 .FSTAT Macro

If used, this macro will return a zero to the AC since the LK35 is a non-directoried device. The form of this macro is:

.FSTAT a,D

where:

a = .DAT slot number  
D = ignored by LKA.

The expansion of .FSTAT is:

```
LOC      CAL+3000+a9-17
LOC+1    2
LOC+2    D
```

### 7.7 IGNORED FUNCTION

The .SEEK macro is ignored by the LKA handler.

## 7.8 ILLEGAL FUNCTIONS

The following macros are illegal with regard to the LKA handler.

.WRITE  
.DELETE  
.RENAM  
.ENTER  
.CLEAR  
.MTAPE  
.TRAN

## 7.9 LEGAL CONTROL CHARACTERS

The following keyboard control entries are recognized by LKA:

ENTRY	OPERATION
1) CTRL C (↑C)	Performs on .EXIT to the Monitor.
2) CTRL P (↑P)	Transfers control to the address given in the .INIT cal.
3) CTRL D (↑D)	Gives an End-of-Medium header word pair to the user.



## CHAPTER 8

### VW01 Writing Tablet Handler

The VW01 Sonic Digitizer Writing Tablet converts graphical information, in the form of X- and Y-coordinates, to digital data that can be input to a digital computer. The major components of the VW01, are the writing tablet, spark pen, component box, and computer interface logic.

The user places a sheet of paper on the writing tablet and draws sketches, schematics, and hand-written symbols or characters using the special ball-point spark pen. The sound of the spark emitted by the pen is picked up by microphones located along the X- and Y-axes of the writing tablet. The time lapse, from spark emission until the sound is picked up by each bank of microphones, is accurately measured to provide a digital record of the X- and Y-coordinates of the spark pen location on the paper.

The digitized graphic data is input to a digital computer via the VWA handler for immediate or delayed processing.

The VW01 operates in either of two modes: Single Point or Data Input.

In the Single Point mode of operation, a single spark is generated each time the spark pen is pressed against the writing surface. The spark is initiated by the closure of a microswitch within the spark pen. The Single Point mode is used if the operator desires to plot points. For example, to plot points at four different locations, he positions the pen point at each location. Then, by pressing and releasing the pen at each position, the corresponding X-Y coordinate pairs are sensed and digitized.

In the Data Input mode, a continuous series of sparks are generated at a constant rate, under control of clock pulses. The X-Y coordinate pairs are continuously generated and input to the computer. This mode allows the user to draw continuous lines, circles, curves, etc., that can be displayed on the CRT.

At the time a spark is generated, X- and Y- clock pulses are initiated which increment X- and Y- hardware registers until the sound of the spark is received by the X- and Y-microphones. As soon as a microphone detects the sound, the associated X- or Y-clock pulses are inhibited, and the register stops incrementing. The binary numbers contained in the X- and Y-registers will then be directly proportional to the X- and Y- coordinates of the position at which the spark was emitted.

The VWA device handler for the VW01 Sonic Digitizer Writing Tablet provides an interface between the user and the hardware. In general, it conforms to the conventions of the Keyboard Monitor System in either the ADSS or DOS software in DEC manual ADVANCED Monitor Software system. Initialize and input functions are initiated by standard user program commands (system macros). The device handler relieves the user from writing his own device handling subprograms

The Writing Tablet handler makes no tests on incoming X- and Y-coordinates. All coordinates are handled directly back to the user. This means that if the pen stays on the same spot (Data Input mode) or is pushed on at the same spot more than once (Single Point mode) the same X- and Y-coordinates are handled to the user. Repetitive X- and Y-coordinates should not be sent directly to the VT-handler since they could cause a hole to be burned on the display-screen. For this reason it is the user's responsibility to ignore X- and Y-coordinates which are generated on one and the same spot. The number of times the same coordinates could be accepted also depends on the intensity.

#### 8.1 .INIT (INITIALIZE) MACRO

The macro .INIT causes the Writing Tablet to be initialized and must be given prior to any other I/O command referencing this device.

The .INIT macro clears one software and two hardware flags. These flags are:

- |                      |           |
|----------------------|-----------|
| 1) Handler Busy flag | /Software |
| 2) Data Ready flag   | /Hardware |
| 3) Pen Data flag     | /Hardware |

The form is:

```
.INIT A,F,R,n
```

where:

A	=	Device Assignment Table (.DAT) slot number
F	=	Not used
R	=	Not used
n	=	Not used

The expansion is:

LOC	CAL+F(7-2)+A(9-17)	
LOC+1	1	/Function code for .INIT
LOC+2	R	
LOC+3	n	

## 8.2 .READ MACRO

The .READ macro is used for input point data to the user from the Writing Tablet. The input always consists of one status word and two words containing the X- and Y-point coordinates.

The status word has the following format:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Input .DAT flag

Writing Tablet Identifier Bits

where:

means:

Bit 0 = 0	Input from "DATA READY"
= 1	Input from "PEN DATA"
Bit 14=1	Input from Writing Tablet 1
Bit 15=1	Input from Writing Tablet 2
16=1	Input from Writing Tablet 3
17=1	Input from Writing Tablet 4

The form is:

.READ A,M,L,W

where:

A = Device Assignment Table (.DAT) Slot Number

M = Data Mode:

      0 = Single Point

      1 = Single Point multiplexed

      2 = Data Input (not scan!)

      3 = Data Input multiplexed (scan!)

L = Line buffer address

      Points to a three word data buffer

W = Writing tablet to be selected (1-4)

The expansion is:

LOC	CAL+M(6-8)+A(9-17)	
LOC+1	10	/Function code for .READ
LOC+2	L	
LOC+3	W	

### 8.3 .WAIT MACRO

The .WAIT macro is used only with respect to the .READ macro. If a .WAIT is given the user program waits until the .READ has completed, that is, when the line buffer is filled and is again available for the user program. If the line buffer is available, control is returned to the user immediately after the .WAIT macro expansion (LOC+2). If the input of data has not yet been completed, control is returned to the .WAIT macro.

The form is:

```
.WAIT    A
```

where: A = Device Assignment Table (.DAT) slot number

The expansion is:

```
LOC      CAL+A(9-17)
LOC+1    12                      /Function Code for .WAIT
```

### 8.4 .WAITR MACRO

The .WAITR macro is also used only with respect to the .READ. If the previous .READ is done, control is returned to the user immediately after the .WAIT in order to proceed in line. If the input of data has not yet been completed, however, control is given to a location in the user program specified in the .WAITR call.

The form is:

```
.WAITR   A,ADDR
```

where: A = Device Assignment Table (.DAT) slot number

ADDR = Location in the user program to which control must  
be transferred if input is not completed.

The expansion is:

```
LOC      CAL+10008+A(9-17)
LOC+1    12                      /Function code for .WAITR
LOC+2    ADDR
```

### 8.5 .FSTAT MACRO

The .FSTAT macro checks the status of a file specified by the file entry block. On return the AC will contain zero and bits 0-2 of LOC+2 will also be zero, stating that the device was non-directoried.

The form is:

`.FSTAT A,D`

where:

A = Device Assignment Table (.DAT) slot number

D = Address of a three word block of storage (directory entry block) in user area containing the file name and the extension of the file whose presence is to be examined.

The expansion is:

LOC	CAL+3000 <sub>8</sub> +A(9-17)	
LOC+1	2	/Function code for .FSTAT
LOC+2	D	

### 8.6 .CLOSE MACRO

Once input has been initiated (.INIT and .READ) it must be terminated by the .CLOSE macro. The hardware flags (Data Ready and Pen Data) are cleared and the Writing Tablet(s) is disabled in order to prevent illegal interrupts.

The form is:

`.CLOSE A`

where: A = Device Assignment Table (.DAT) slot number

The expansion is:

LOC	CAL+A(9-17)	
LOC+1	6	/Function code for .CLOSE

## 8.7 IGNORED FUNCTIONS

The following macros are ignored by the VWA device handler:

- 1) .SEEK
- 2) .ENTER
- 3) .CLEAR
- 4) .MTAPE
- 5) .WRITE
- 6) .TRAN
- 7) .DELETE
- 8) .RENAM

## CHAPTER 9

### TEXT DISPLAY/EDIT FUNCTIONS

The VT15 GRAPHICS software provides the user with a complete text editing program, EDITVT, and a soft copy display feature, CONTROL X.

The EDITVT program has the same command and editing structure as the standard Editor (i.e., EDIT, refer to DEC-15-YWZB-DN ) except that the majority of the text presentation takes place on the VT04 display CRT. The Control X (CTRL X) feature enables the user to, essentially, replace the console printer with the display CRT when desired.

#### 9.1 EDITVT

Systems which have a VT15 Graphics Display unit permit the user to employ program EDITVT for editing purposes. Program EDITVT enables the user to perform soft copy editing of files using the VT15 display as a file data display device. Data is displayed in sets of either 56, 72-character lines or 28, 72-character lines. The EDITVT commands and the editing functions performed are essentially the same as those of the standard Editor program (EDIT, refer to DEC-15-YWZB-DN ).

##### 9.1.1 Setup Commands

The following command must be issued to the monitor prior to loading EDITVT:

- |                   |   |
|-------------------|---|
| a) \$ VT ON       | Enables the VT display unit.  |
| b) \$ HALF ON/OFF | This command is optional; it enables the user to set up a half-screen display (i.e., 28, 72-character lines) condition in which only half the screen is used for display. |

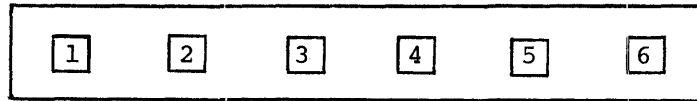
The program EDITVT is loaded into core by the command "EDITVT" given to the Monitor. Once loaded, the program announces itself by outputting its name and version number on the console printer. The user must then input the command "TV ON" to initiate the VT15 display operations. VT15 display operations may be stopped at any time by the command "TV OFF".

##### 9.1.2 Controls

The VT-15 Display console contains a horizontal strip of six square push-to-light pushbuttons which are used in display operations. These pushbuttons

are unmarked since their function is determined by software and may vary according to the particular program (system or user) which is in control of the system.

In EDIT operations, only the two rightmost pushbuttons are needed; these switches are referred to as numbers 5 and 6, based on the following numbering scheme:



### 9.1.3 Display Modes

The VT-15 Display operates in two display modes:

- a) SCROLL Mode - When the number 5 pushbutton is in the OFF (unlit) position, the display is in the SCROLL mode. In this mode, when the display screen is full, the next line of data to be displayed causes the displayed material to "roll" upwards, line-by-line, with new data displayed at the bottom of the screen.
- b) PAGE Mode - When the number 5 pushbutton is in the ON (lit) position, the display is in the PAGE mode. In this mode, when the screen is full, the next entered material for display causes the complete, full-screen display (i.e., PAGE) to be erased; the new material is then displayed starting at the top line of the screen. When a large file is to be displayed in the PAGE mode, the number 6 control pushbutton is used to advance the display through the file page (screen) by page (screen). Each time this pushbutton is actuated, the screen is cleared and the next set (page) of data available is displayed.

### 9.2 CONTROL X FEATURE

The Control X feature gives the user the ability to change from hard to soft copy at any time during Monitor operation. When Soft copy is desired the user types TV ON when under Monitor Control and then a Control X. The TV ON command sets up the necessary linkage in the teletype handler and also reserves a segment of core to be used for the Display Buffer. The Control X command may be typed during Monitor Control or during System Program Control; it switches output from the device presently being used to the alternate device. (Teletype to display or display to teletype.) When the display is being used, teletype input is echoed on both the teletype and on the display while teletype output appears only on the display.

### 9.2.1. SCROLL Mode

When text is being output to the display and the display screen is filled (56) lines, the next incoming line appears on the bottom of the screen and the oldest or top line on the screen disappears. It appears as if the text is rotating from screen bottom to screen top. The display screen may be cleared at any time and new text begins at screen top by changing the position of pushbutton number 6; and then typing a carriage return.

### 9.2.2 PAGE Mode

The display may be put in page mode operation. That means that when the display has 56 lines being presented it stops output to it so the user can inspect the text and it then waits for the user to advance to the next page. This feature is useful for doing a PIP transfer of a large file to the display; the file can be read on the display a page at a time. It is also useful for looking at Macro Assemblers and FORTRAN compilations on the display. Page Mode operation is entered by setting pushbutton number 5 to the ON position; for normal operation (text rotation across screen) pushbutton number 5 should be in the OFF position. When in page mode, a page is advanced by changing the position of pushbutton number 6.

### 9.2.3 TV ON/OFF Monitor Commands

The TV ON command sets up the interface between the VT15 Display System and the Teletype Handler Section of the Resident Monitor. The Display Interface Code is moved to a position directly above the Resident Monitor and essentially becomes a part of the resident monitor. The TV ON command also reserves a segment of core for use as the Display Buffer. Once the TV ON command has been issued the user has the ability to switch his output device from Teletype to display and from display to teletype. The output device switching is accomplished by typing a ↑X (Control X); and may be done when under monitor control or user program control.

The feature gives the user the ability to work from an extremely fast, soft copy output device; and easily switch to hard copy when it is desired. When ↑X is typed, an Up-Arrow (↑) is echoed on the device selected for output. The TV OFF command releases the reserved core segment and it frees the area of core directly above the Resident Monitor where the Display Interface Code was moved. The TV ON command remains in effect until TV OFF is issued or the Monitor System is bootstrapped. If the VT15 Display System is desired as the primary output device, TV ON may be set at System Generation time. The TV OFF command can override the System Generation setting, allowing selection of hard copy output.

#### 9.2.4 HALF ON/OFF Monitor Command

The HALF ON/OFF command can be used in ↑X operations.

#### 9.2.5 Differences Between CTRL X in V5A (i.e. ADSS) and in DOS.

Control X will work slightly differently under DOS15 than under the Advanced Monitor System (ADSS). The DOS15 Monitor also includes some additional features associated with ↑X.

The System Generator under ADSS does not ask questions concerning TV ON/OFF and half buffer ON/OFF settings. Under DOS15 the VT can be set to the ON position at system generation time and then it will not be necessary for the user to type TV ON; the same applies to HALF buffer ON. Under DOS15 the loading of a system CUSP does not cause the display to be cleared thus requiring another ↑X to be typed. Under DOS15 the display will continue its text presentation not only when system CUSPs are loaded but also when the Monitor is refreshed. Control X under ADSS (V5A) is NOT supported but is available to the user.

## APPENDIX A

### MNEMONICS COMMONLY USED IN GRAPHICS SUBPROGRAM CALLS

The following mnemonics are commonly used in describing subroutine call statements throughout this manual.

<u>Mnemonic</u>	<u>Definition</u>
1. DELTAX	An integer number or variable which represents in raster units the amount the CRT beam is to be displaced from its current position in a horizontal direction. This quantity is signed to indicate the direction of displacement (i.e., + = move beam right - = move beam left).
2. DELTAY	Same as DELTAX except that the indicated displacement is made on a vertical direction and the directions indicated by the sign are: + = move beam up - = move beam down.
3. INT	This variable is restricted to the Integer values 1 and 0 to indicate if the CRT beam movement is to be visible, (INT = 1) to draw a line, or invisible (INT = 0).
4. PNAME	The subpicture display files generated by the graphic subpicture calls are stored in dimensioned integer arrays specified by the user. The integer variable PNAME specifies the first element of the array into which commands generated by a particular call are to be stored. PNAME is always represented as a subscripted variable; it will contain the length of the file and is the variable by which the file is referenced in later manipulations. NOTE: The variable PNAME may be dropped from the statement argument lists; if dropped, the last given value for PNAME will be assumed.
5. STR	Identifies the dimensioned real array which contains the string of characters to be displayed in IOPS ASCII (Hollerith) form (five 7-bit characters per word).
6. FEATR	An integer number which identifies a hardware feature(s) to be specified in the call (e.g., 1 = scale, 2 = intensity, 4 = light pen, and 8 = blink).

<u>Mnemonic</u>	<u>Definition</u>
7. VALUE	A single integer variable or constant that indicates the value or setting is specified for a selected display feature.
8. DTA	Contains the set of data points, one per word, in the range 0 to 1023 (Integer).
9. N	Used by GRAPH subprogram to indicate the number of points to graph. Also used by TEXT subprogram to indicate the number of characters to be displayed.
10. A	An integer variable or constant restricted to the values 0 and 1. Indicates which axis to increment for GRAPH subprogram, 0 = increment X, set Y to data values, 1 = increment Y, set X to data values.
11. MAINFL	Similar to PNAME, the value of MAINFL represents the first array element of the dimensioned Integer array specified by the user for storing main display file commands. MAINFL is represented as a subscripted integer variable, it contains the length of the file and is the variable by which the file is referenced.
12. CNAME	An integer variable that identifies the location or first location which contains the display command(s) generated by the call in which CNAME is an output argument.
13. NAMR	An integer which represents the contents of the name register at the time of a light pen hit (restricted to values ranging from 1 to 127).
14. PB	A six-element integer array which will contain a logical .T, or .F, for each of the six pushbuttons.
15. RST	This variable, restricted to the integer values of 1 and 0, indicates whether the hardware SAVE/RESTORE option is to be used when copying subpicture files. The value 0 indicates that the SAVE/RESTORE option is not to be used; the value 1 indicates that it is to be used.

APPENDIX B  
 DISPLAY INSTRUCTION GROUPS  
 Generated By  
 GRAPHICS SUBPROGRAM CALLS

SUBPROGRAM CALL	NUMBER OF COMMANDS	COMMANDS GENERATED
LINE PLOT( $\emptyset$ ,... RELOT( $\emptyset$ ,...	1	If one of the eight basic directions:  VN!INCR
	2	If random vector option is used:  SVX! DELTAX SVY! DELTAY
	N+2	If not one of the above, required line is approximated with a series of basic vectors:  SKP (COUNT=N+2) V1 V2 . . . VN
TEXT PLOT(3,... RELOT(3,...	3	CHARS* .+2 DJMP .+2 (FULL 15-BIT ADDRESS)
COPY PLOT( $\emptyset$ ,... RELOT( $\emptyset$ ,...	3	When SAVE/RESTORE is not used:  DJMS* .+2 DJMP .+2 (FULL 15-BIT ADDRESS)
	6	When SAVE/RESTORE is specified:  SAVE .+4 DJMS* .+2 DJMP .+3 (FULL 15-BIT ADDRESS) (STATUS) RSTR .-1
PRAMTR PLOT(2,... RELOT(2,...	1-4	Adds from one to four parameter words to the display file, depending on the type of argument list used.
GRAPH	N	Adds N graph plot commands to the dis- play file, where N is equal to the number of points in the data set:  GY!Y1           GX!X1 GY!Y2           GX!X2 . . . or . . . GY!YN           GX!XN
SETPT RSETPT	2	PY!Y PX!X



# APPENDIX C

## MACRO EXPANSION OF GRAPHICS SUBPROGRAM CALLS

### Subpicture Routines

LINE		GRAPH	
.GLOBL	LINE	.GLOBL	GRAPH
JMS*	LINE	JMS*	GRAPH
JMP	.+5	JMP	.+5
.DSA	DELTAX	.DSA	DTA
.DSA	DELTAY	.DSA	N
.DSA	INT	.DSA	A
[.DSA	PNAME ]	[.DSA	PNAME ]
TEXT		BLANK	
.GLOBL	TEXT	.GLOBL	BLANK
JMS*	TEXT	JMS*	BLANK
JMP	.+4	JMP	.+2
.DSA	STR	.DSA	PNAME
.DSA	N		
[.DSA	PNAME ]		
COPY		UNBLNK	
.GLOBL	COPY	.GLOBL	UNBLNK
JMS*	COPY	JMS*	UNBLNK
JMP	.+4	JMP	.+2
.DSA	RST	.DSA	PNAME
.DSA	PNAME1		
[.DSA	PNAME ]		
PRAMTR		CIRCLE	
.GLOBL	PRAMTR	.GLOBL	CIRCLE
JMS*	PRAMTR	JMS*	CIRCLE
JMP	.+N	JMP	.+7
.DSA	FEATR	.DSA	R
.DSA	VALUE	.DSA	THETA
[.DSA	PNAME ]	.DSA	GAMMA
where N=2+(Number of Features		.DSA	DFG
specified)+1 if PNAME		.DSA	PNAME
is given		ROTATE	
		.GLOBL	ROTATE
		JMS	ROTATE
		JMP	.+12
		.DSA	ISTR
		.DSA	IA
		.DSA	IP
		.DSA	IC
		.DSA	X
		.DSA	Y
		.DSA	Z
		.DSA	SINA
		.DSA	CSA

## Input Routines

### LTORPB

```
.GLOBL  LTORPB
JMS*    LTORPB
JMP      .+6
.DSA     IX
.DSA     IY
.DSA     NAMR
.DSA     PB
.DSA     IWICH
DAC      I
```

### TRACK

```
.GLOBL  TRACK
JMS*    TRACK
JMP      .+5
.DSA     IX
.DSA     IY
.DSA     IOPT
.DSA     IARRAY
```

## Relocating Routines

### DYSET

```
.GLOBL  DYSET
JMS*    DYSET
JMP      2*N+.+1
.DSA     PNAME
.DSA     ASCII
.
.
.
.DSA     PNAMEN
.DSA     ASCIIN
        (where n = number
              of files)
```

### DYLINK

```
.GLOBL  DYLINK
JMS*    DYLINK
JMP      2*N+.+1
.DSA     PNAME1
.DSA     ASCII1
.
.
.
.DSA     PNAMEN
.DSA     ASCIIN
        (where N = number
              of files)
```

## Main Display File Routines

### DINIT

```
.GLOBL  DINIT
JMS*    DINIT
JMP      .+2
.DSA     MAINFL
```

### SETPT

```
.GLOBL  SETPT
JMS*    SETPT
JMP      .+4
.DSA     X
.DSA     Y
[.DSA    CNAME ]
```

### DCLOSE

```
.GLOBL  DCLOSE
JMS*    DCLOSE
```

# Main Display File Routines (Cont.)

## PLOT a COPY

```
.GLOBL PLOT
JMS* PLOT
JMP .+5
.DSA (Ø
.DSA RST
.DSA PNAME
[.DSA CNAME]
```

## REPLOT

```
.GLOBL REPLOT
JMS* REPLOT
JMP .+5
.DSA (Ø
.DSA RST
.DSA PNAME
.DSA CNAME
DAC I/ if used as function
```

## PLOT a LINE

```
.GLOBL PLOT
JMS* PLOT
JMP .+6
.DSA (1
.DSA DELTAX
.DSA DELTAY
.DSA INT
[.DSA CNAME]
```

## REPLOT a LINE

```
.GLOBL REPLOT
JMS* REPLOT
JMP .+6
.DSA (1
.DSA DELTAX
.DSA DELTAY
.DSA INT
.DSA CNAME
DAC I/ if used as function
```

## PLOT a PRAMTR

```
.GLOBL PLOT
JMS* PLOT
JMP .+5
.DSA (2
.DSA FEATR
.DSA VALUE
[.DSA CNAME]
```

## REPLOT a PRAMTR

```
.GLOBL REPLOT
JMS* REPLOT
JMP .+5
.DSA (2
.DSA FEATR
.DSA VALUE
.DSA CNAME
DAC I/ if used as function
```

## PLOT a TEXT string

```
.GLOBL PLOT
JMS* PLOT
JMP .+5
.DSA (3
.DSA STR
.DSA N
[.DSA CNAME]
```

## REPLOT a TEXT string

```
.GLOBL REPLOT
JMS* REPLOT
JMP .+5
.DSA (3
.DSA STR
.DSA N
.DSA CNAME
DAC I/ if used as function
```

## DELETE

```
.GLOBL DELETE
JMS* DELETE
JMP .+2
.DSA CNAME
DAC I/ if used as function
```

## RSETPT

```
.GLOBL RSETPT
JMS* RSETPT
JMP .+4
.DSA X
.DSA Y
.DSA CNAME
DAC I/ if used as function
```



## APPENDIX D

### CONDITIONAL ASSEMBLY OF GRAPHICS SUBPROGRAMS

For VT15 configurations that include the Arbitrary Vector Option, the Graphics Subprogram Package (VTPRIM) can be conditionally assembled to eliminate coding required for line approximation. This procedure saves approximately  $174_8$  locations. The standard procedure for conditional assembly may be followed; it is only necessary to define a value for the variable ARBVEC when assembling VTPRIM SRC.

#### WARNING

In writing MACRO routines, the exclamation point (!) must not be used in memory reference type instructions to separate the Op-code and address fields. The symbol ! used in this manner causes the contents of the Op-code and address fields to be OR'd together resulting in an erroneous 15-bit address.



## HOW TO OBTAIN SOFTWARE INFORMATION

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

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

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

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

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

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

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

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

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

READER'S COMMENTS

VT-15 Graphics Software  
System, Programming Manual  
DEC-15-ZFSB-D

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

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

---

---

---

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

---

---

---

---

How can this manual be improved?

---

---

---

---

Other comments?

---

---

---

---

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

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

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

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

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

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

FIRST CLASS  
PERMIT NO. 33  
MAYNARD, MASS.

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

Postage will be paid by:

**digital**

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

