CONTENTS

CHAPTER 1

INTRODUCTION

CHAPTER 2

GRAPHIC SUBPROGRAM ROUTINES

CHAPTER 3

SYSTEM I/O DEVICE HANDLER

APPENDIX A

SAMPLE PROGRAMS

LIST OF TABLES

LIST OF ILLUSTRATIONS

Type 339 Buffered Display System

CHAPTER 1

INTRODUCTION

The PDP-9/339 Graphics Subprogram Package consists of a group of subroutines and functions which may be called by user programs written in FORTRAN IV or Macro-9. Its purpose is to provide a simplified means of using the 339 graphic display device without requiring detailed familiarity with the hardware. Its function is to compile display commands at the direction of the user and allow him to define display elements and direct the linking, displaying, deleting, etc., of those elements.

With the exception of Section 3, this manual is concerned with the 339 Graphics Subprogram Software Package and its use in developing 339 display source programs in FORTRAN IV.

Section 3 presents a description of the 339 System I/O Device Handler and the MACROs which comprise it.

The subprograms which make up the 339 Graphics package may be called using MACRO-9 .GLOBL pseudo-ops as well as FORTRAN IV CALL statements. Therefore, the MACRO equivalent of each CALL statement described is also given. All programming procedures described, however, are applicable to FORTRAN-level programs only.

Detailed descriptions of the 339 hardware and the development of assembly-language source programs for the 339 are given in the 339 User's Manual and associated PDP-9 ADVANCED Software System Manuals (refer to Table 1-1).

1.1 REFERENCE DOCUMENTS

The available documents which contain information useful in programming the 339 Programmed Buffered Display System are listed and described briefly in Table 1-1.

Table 1-1. Applicable Reference Documents

	DOCUMENT	DESCRIPTION
1.	339 Programmed Buffered Display User's Handbook DEC-09-16FA-D	Contains descriptions of the physical and functional characteristics of the 339 Display System and presents the Symbolic Codes unique to the 339 which are used in addition to the MACRO-9 codes for the preparation of 339 assembly language programs.
2.	PDP-9 ASSEMBLER DEC-9A-AMZA-D	Provides the information required for the preparation of MACRO-9 Assembly language source programs.
3.	PDP-9 FORTRAN IV DEC-9A-KFZA-D	Provides the reader with the information required to prepare FORTRAN IV programs for compilation and execution within the PDP-9 ADVANCED Software System.
4.	PDP-9 Keyboard Monitor Guide DEC-9A-MKFA-D	Provides keyboard operating instructions and procedures for PDP-9 ADVANCED Software System Programs.
5.	PDP-9 Utility Manual DEC-9A-GUAB-D	Contains individual detailed descriptions of the Utility programs supplied with the PDP-9 ADVANCED Software System.

1.2 REQUIRED 339 SYSTEM PROGRAMS

The programs which must be loaded into the system to provide the capabilities described in this manual are listed in Table 1-2. These programs are assembled by MACRO-9 or 9A into relocatable binary files which may be loaded by the Linking Loader. System program UPDATE-9 (refer to Utility Manual) may be used to incorporate these programs into either the System Library or a User Library; if they are to be included in the System Library, all unused device handlers and routines must be removed to provide space.

Table 1-2, Required 339 System Programs

Program Name	Capabilities	Length
DYA.4	All I/O Macros (.INIT, .READ, .WRITE, .WAIT, .WAITR, .CLOSE	1033_8 655_{10}
MAINFL	DINIT, DCLOSE, SETPT, PLOT, RSETPT, REPLOT, DELETE	650_8 424_{10}
SUBPIC (for VC38, or SUBPVA for VA39)	LINE, TEXT, COPY, PRAMTR, GRAPH	656_8 425_{10}
INTPT	LTPN, PBTN	272_8 186_{10}
TRCK2	TRACK	457_8 303_{10}
BLANK	BLANK, UNBLNK	46_8 38_{10}
DYLDR	DYSET, DYLINK	276_8 190_{10}
DYCHR	Character Table for VC38 (Loaded by the Monitor switch "VC38 ON")	677_8 447_{10}

1.3 SYSTEM DESCRIPTION

The Type 339 Programmed Buffered Display (Frontispiece) is an on-line system manufactured by Digital Equipment Corporation for operator observation, manipulation, and alteration of computer-stored data. The display system consists of a precision incremental CRT display, display control logic, light pen, pushbutton control box, and interface logic for communication with a PDP-9 computer. The computer processor and the display control logic share the computer memory, and are interdependent as a system.

The computer system furnishes rapid data storage and retrieval of data for display as dots, lines, curves, or characters (optional) on the face of the CRT. The operator can manipulate or alter the data by using the light pen, pushbutton control box, or the keys and switches on the computer console.

The Type 339 Programmed Buffered Display interfaces to the PDP-9 general purpose computer through the optional Direct Memory Access Channel Multiplexer Adapter, Type DM09A. The result is a multiprocessor system in which both the display processor and the computer's central processor operate out of the same core memory on a time-shared basis.

The direct memory access channel (DMA) interface permits the 339 Display to operate from the PDP-9 memory, obtaining its instructions on a cycle-stealing basis while allowing the PDP-9 to process a completely separate program. The PDP-9 can quickly interrupt this program, to service real-time interrupts generated by the operator, the display, or other external devices. The use of a common memory

for the display and the PDP-9 allows the computer to update the display files quickly, with a minimum of communication problems.

The 339 display system executes instructions contained in lists (groups of consecutive locations) in the shared memory. These lists are compiled from FORTRAN or symbolic language (MACRO-9) source programs or are loaded into memory from an external source.

Once a display program is loaded and compiled, the PDP-9 supplies the starting address of the list to the 339 display system and commands the display to begin. At this point, the PDP-9 is free to execute a completely separate program, pausing only to service interrupts caused by the display light pen, pushbutton control box, or other external source.

Display programs may consist of continuous loops of instruction or of chains comprised of several widely scattered lists. In either case the display is kept running continuously by the use of jump or jump-to-subroutine instructions which can be used to (1) direct the display processor, (2) proceed to the next list, or (3) return to the beginning of the current list. Skip instructions, used by the type 339 display, cause the display processor to test for specified conditions and to branch to different programs if the conditions are met.

1.3.1 Easy Man/Machine Interaction

The Type 339 standard input features, the light pen and pushbutton control box, provide the user with versatile means of communication

with the display system. The light pen allows the operator to "point" to a particular portion of the display and indicate that a change is desired. With tracking subroutines, the light pen can be used to "draw" lines and figures.

The pushbutton control box also provides a simple means of communication with the system. It features 12 pushbuttons, in two banks of 6 buttons each, a manual reset button for each bank, and a manual interrupt button that can be used to cause a PDP-9 program interrupt.

The activation of any pushbutton is sensed by the computer which then enters an interrupt mode. In the interrupt mode the processor will perform any operation specified as the response to the activated pushbutton.

1.4 339 SOFTWARE OPERATIONS

A functional block diagram illustrating the overall software operations performed in the 339 system is shown in Figure 1-1. The FORTRAN IV programs devised by the user will consist of standard FORTRAN IV statements and calls for routines unique to the 339 Graphics system:

1) MAIN DISPLAY FILE ROUTINES - Calls for these routines together with the standard FORTRAN IV statements will, when compiled, build a "Main Display File" in a portion of the PDP-9 memory assigned to the 339 Display system. The commands contained by this file will, during execution of the program, link together individual Subpicture Display Files into a CRT display command chain which will cause the desired image to be displayed.

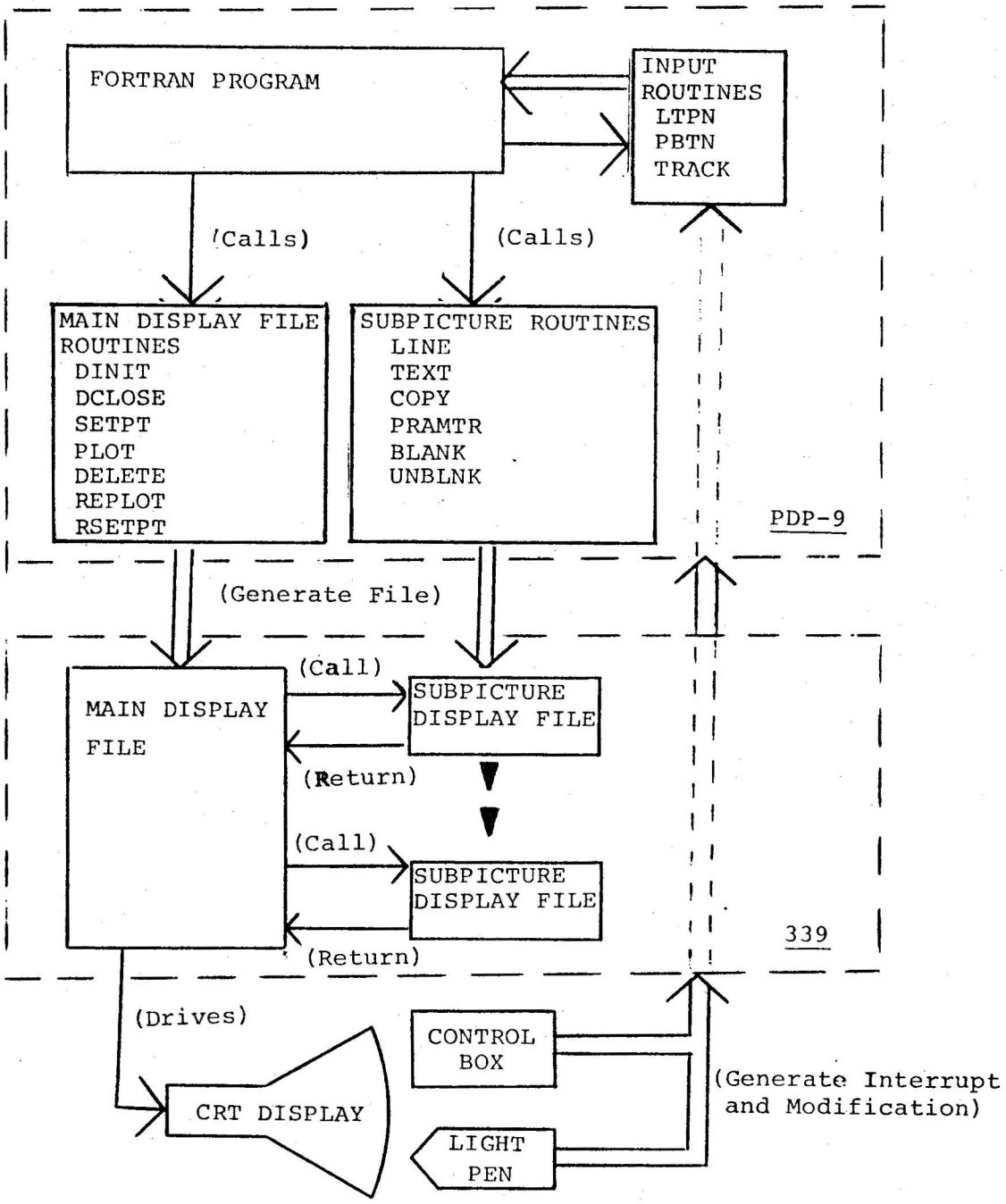


Figure 1-1, 339 Graphics Software Operations

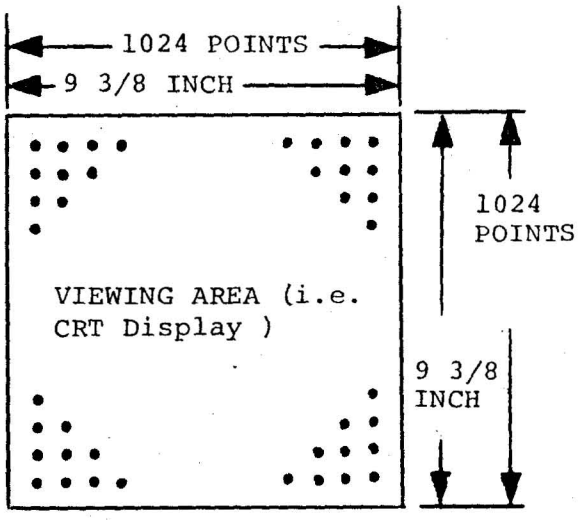
2) SUBPICTURE ROUTINES - Calls for these routines together with standard FORTRAN statements will, when compiled, form individual self-contained Subpicture Display Files. Each subpicture file contains all of the commands needed to generate a specific image on the 339 display CRT. These files are accessed by the Main Display File in any required order or sequence during the execution of the display program. The Subpicture files are normally programmed and compiled separately from the Main Display File program to build a library of accessible graphics (i.e. complete or partial pictorial images) from which complex images may be formed.

3) INPUT ROUTINES - Calls for 339 input subroutines which permit the use of the light pen and pushbuttons may be made in the FORTRAN program to direct display operations. These are used to control the flow of the program upon the occurrence of light pen or pushbutton interrupts. In this way program paths can be activated which modify the display command chain and, therefore, modify the picture.

1.5 339 DISPLAY DEVICE

The 339 system uses a 16-inch Cathode Ray Tube (CRT) which provides a usable display area 9 3/8 by 9 3/8 inches in size as its display device.

For programming purposes, the CRT display area (face) may be considered a dot matrix comprised of over a million (1024 by 1024) illuminable points (dots) spaced at about 0.01 inch intervals (see Figure 1-2). For display purposes, the matrix dots are illuminated,



This area consists of
a matrix of 1024 X 1024
illuminable points

Figure 1-2, 339 Graphics System Display Area

selectively, by an electron beam generated within the CRT. The illumination of a series of dots in any direction results in a visually continuous line of uniform resolution. Images are developed on the 339 CRT display by the controlled generation and positioning of the CRT electron beam in direct response to a programmed instruction list processed by the 339 system.

1.5.1 Programmable Display Area (Refer to Figure 1-3)

Although the CRT display area is physically limited to a 9 3/8 by 9 3/8 inch area, the hardware capabilities of the 339 system permit the user to develop images up to 8 times the display area in size. For programming purposes, the available (programmable) drawing area may be considered as a 75 by 75 inch sheet of paper on which one or more images may be developed.

Viewing any part of the programmable drawing area is accomplished by program features which, in effect, cause the "drawing sheet" to move under the viewing window (i.e. CRT display) to any desired position.

1.5.2 Program-controlled CRT Characteristics

Display Scale and Intensity are variable display characteristics which must be specified by the programmer in each display program. Two control functions, Light Pen on/off and Blink on/off are also programmable.

1.5.3 Scale

As previously described, images are generated on the display CRT by the controlled intensification of points within the display dot

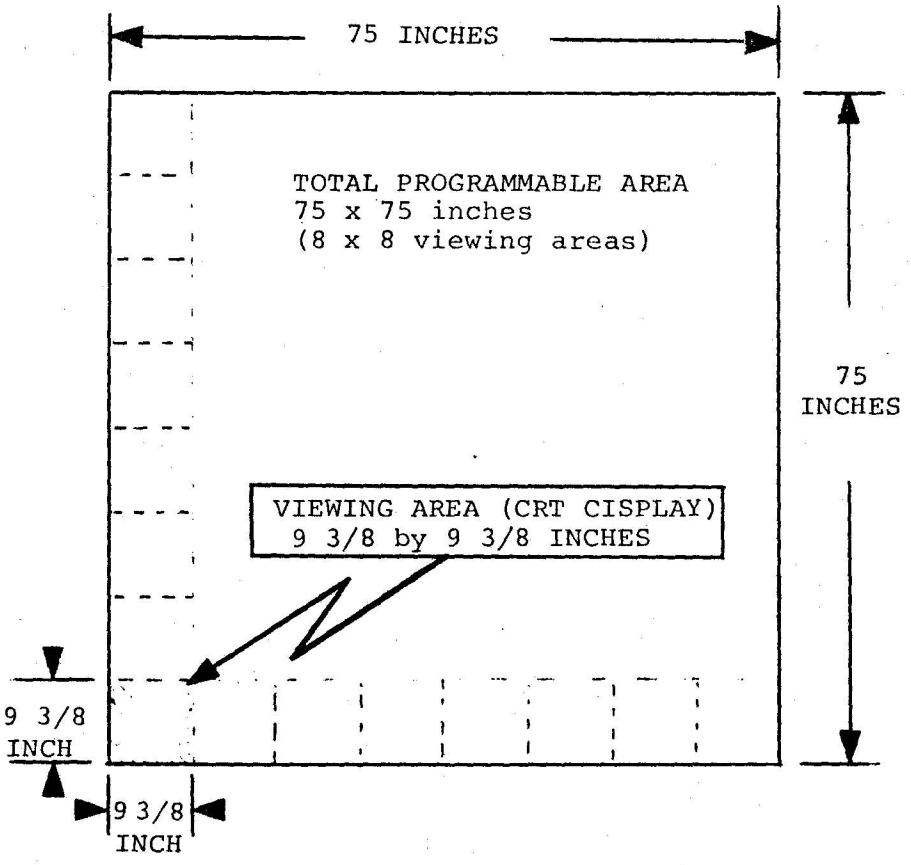


Figure 1-3, Programmable Display Area

matrix. Scale controls the amount of space to be left between the intensified dots during a display operation. There are 4 possible scale settings (0,1,2, and 3); the settings and the manner in which each affects the display are illustrated in Figure 1-4.

Figure 1-4, Scale Settings

Scale	Point Spacing	Intensify
0	● ● ● ● ● ● ● ● ● ●	Every
1	● o ● o ● o ● o ● o	2nd
2	● o o o ● o o o ● o	4th
3	● o o o o o o o ● o	8th

As illustrated in Figure 1-4, scale settings affect both the size and appearance of displayed lines or symbols. For example, at a setting of 3, each point will be distinguishable; at a setting of either 0 or 1, however, lines and symbols will appear to be continuous.

1.5.4 Intensity

The intensity at which selected matrix dots are to be illuminated during a display operation is specified by the user at any one of eight possible settings. The available settings range in increasing brightness from 0 to 7 with 0 setting being not visible, and 7 very bright.

Scale and intensity settings are interrelated and must be carefully selected to insure that the best possible display image is obtained. Scale setting affects the length of the line proportionately, since

the increments given are actually a count of intensified points rather than absolute points. Therefore, it affects the intensity in inverse proportion since fewer points are glowing. Thus if the intensity of a line must remain constant with a change in scale setting, the intensity setting must be changed by an equal amount. At a scale setting of 2, the points are clearly separated and the effect on intensity becomes secondary.

1.5.5 Blink Control Function

The ability to cause a displayed image or portion of the image to blink on and off at a visually discernible rate is a function which may be specified by the programmer.

The display "Blink" feature may be specified for any display file whether it represents an entire display or specific portion of a display. This feature is particularly useful when it is necessary to draw attention to specific displayed items.

The blink on/off control element is represented by an integer variable the value of which could, if desired, be made a function of the state of one of the control pushbuttons on the 339 control box.

1.5.6 Light Pen Control Function

The ability to use the light pen supplied with the 339 system is also a programmed function; the user must specify if the light pen is to be enabled or disabled throughout the display operation or is to be controlled from the 339 control box.

the increments given are actually a count of intensified points rather than absolute points. Therefore, it affects the intensity in inverse proportion since fewer points are glowing. Thus if the intensity of a line must remain constant with a change in scale setting, the intensity setting must be changed by an equal amount. At a scale setting of 2, the points are clearly separated and the effect on intensity becomes secondary.

1.5.5 Blink Control Function

The ability to cause a displayed image or portion of the image to blink on and off at a visually discernible rate is a function which may be specified by the programmer.

The display "Blink" feature may be specified for any display file whether it represents an entire display or specific portion of a display. This feature is particularly useful when it is necessary to draw attention to specific displayed items.

The blink on/off control element is represented by an integer variable the value of which could, if desired, be made a function of the state of one of the control pushbuttons on the 339 control box.

1.5.6 Light Pen Control Function

The ability to use the light pen supplied with the 339 system is also a programmed function; the user must specify if the light pen is to be enabled or disabled throughout the display operation or is to be controlled from the 339 control box.

1.5.7 Dimensioning Displays

In programming displays, the dimensions of the graphic elements used to form the picture are specified in terms of Display CRT Raster Units. The basic Raster Unit is defined as the distance between any two of the display CRT matrix illuminable points.

A Display Raster Unit, however, is seen as the distance between any two sequentially illuminated dots of a display. The size of a Display Raster Unit is directly dependent on the SCALE setting specified for the picture being developed.

In relation to display dimensions, SCALE settings are to be regarded as distance multipliers; the manner in which a specified dimension (e.g. 109 Raster Units) varies in actual displayed length according to each of the possible SCALE settings is illustrated in Table 1-3.

Table 1-3, Dimensional Effect of SCALE Settings.

SCALE SETTING	SPECIFIED DIMENSION	DISPLAYED LENGTH IN BASIC RASTER UNITS	VIEWED LENGTH IN INCHES
0	109	109	1-inch
1	109	219	2-inches
2	109	436	4-inches
3	109	872	8-inches

1.5.8 Character Generation

Characters and symbols may be generated by subroutines; however, the use of optional character generators VC38 or VA38 greatly simplifies this task.

The 339 Graphics Software package contains a subprogram which allows the use of either character generator, if present.

CHAPTER 2

GRAPHIC SUBPROGRAM ROUTINES

This section contains detailed descriptions of the FORTRAN IV CALL statements required for each 339 Graphic Subprogram. Each description also includes the equivalent statements needed to call the subprogram in MACRO-9.

All display file storage is supplied by the user in the form of dimensioned integer arrays. To facilitate storage management, the first location of each display file contains the length of the file. Limited reuse of storage is provided for in Main Display File routines REPLOT and DELETE. The storage requirements of each CALL which adds commands to a file are indicated in the description given of the CALL.

Some of the mnemonics used to represent arguments in the descriptions of the CALL statements are listed and defined in Table 2-1.

Table 2-1, Mnemonics Commonly Used in 339 CALL Statement Descriptions

	Mnemonics	Definition
1)	deltay	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 vertical direction. This quantity is signed to indicate the direction of displacement. i.e. + = move beam up - = move beam down
2)	deltax	Same as deltay except that the indicated displacement is made on the horizontal plane and the directions indicated by the sign are: + = move beam right - = move beam left.
3)	It	This variable is restricted to the INTEGER values 1 and 0 to indicate if the CRT beam movement is to be visible. (It = 1) to draw a line, or invisible (It = 0).
4)	pname	The subpicture display files generated by the compilation of the Graphic Subpicture CALLS are stored in dimension arrays specified by the user. An INTEGER variable, pname, represents the first element of the array specified to store the commands generated in

NOTE: The variable pname may be dropped from the statement argument lists; if dropped, the last given value for pname will be assumed.

the compilation of the [illegible] pname is located. 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.

5) featr — An INTEGER number which identifies a feature of the CRT display to be specified in the CALL (i.e. 1 = SCALE, 2 = Intensity, 3 = light pen, and 4 = Blink).

6) value — A single INTEGER digit which indicates what value or setting is specified for the selected display feature.

7) mainfl — Similar to pname, the value of mainfl represents the first array element of the dimensioned array specified by the user to store a Main Display File. 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.

8) cname — An INTEGER which identifies the location or first location which contains the display command(s) generated by the CALL in which cname is an argument.

2.1 SUBPICTURE ROUTINES

The 339 subpicture routines generate files of executable display commands which are identified and manipulated as separate graphical entities (i.e. Subpicture Display Files). The identity of a subpicture file is the address of its first location (pname) and is given as an argument in all calls to the subpicture routines. The calls are self-contained; they permit simultaneous generation of subpictures with each referenced subpicture left in displayable form so that it can be manipulated dynamically while displaying.

The first location of a subpicture (pname) contains its current length; this value must be Ø when the first reference to the Subpicture Display file is made. After the first reference, the value of pname is maintained by the subpicture routine.

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 control and the third (last in file) contains the 339 display command POP (subpicture return).

2.1.1 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.

a. Form

(1) FORTRAN: CALL LINE (deltay, deltax, It[,pname])

(2) Macro-9:

```
        .GLOBL      LINE
        JMS*        LINE
        JMP         .+5
        .DSA        deltay
        .DSA        deltax
        .DSA        It
       [.DSA        pname]
```

b. Input Variables

(1) deltay = Vertical component of beam displacement in raster units (integer).

(2) deltax = Horizontal component of beam displacement in raster units (integer).

(3) It = Line type (integer).

1 visible
0 invisible

(4) pname is the first location of this subpicture display file. It 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 ELEMNT, the calls to LINE building that subpicture are of the form CALL LINE (deltay, deltax, lt, ELEMNT(1)) (integer).

c. Output Variables

pname = Current length of this subpicture file. In the example above, the location ELEMNT (1) contains the length of the file.

d. Description

(1) LINE adds three commands to the display file:

```
VEC!EDS   (vector mode, enter data state)
deltay
deltax    (escape-to-control-state bit on)
```

(2) If it immediately follows another LINE call, LINE clears the preceding escape bit and adds 2 commands to the display file:

```
deltay
deltax   (escape bit on)
```

(3) The number of commands added to the display file is added to the contents of location pname.

e. Restrictions

(1) Deltay and deltax are signed integers, the magnitudes of which must not exceed 1023.

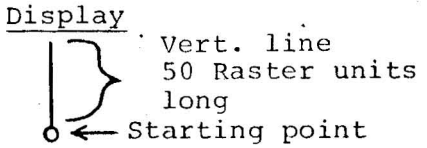
(2) Pname must contain zero before the first call referencing it.

(3) Sufficient space must follow pname to contain the subpicture file.

f. Example. The following two statements illustrate the use of the LINE subroutine to draw straight and sloped lines:

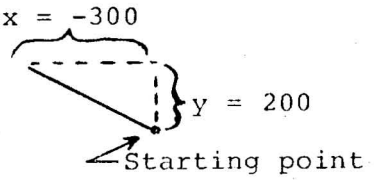
(1) Draw a vertical line:

```
CALL LINE(50,0,1,ILINE (1))
```



(2) Draw a sloped line:

```
CALL LINE(200,-300,1,ILINE(1))
```



2.1.2 TEXT Subroutine

This subroutine call may be used only in 339 systems which contain an optional character generator (VC38 or VA38).

The TEXT subroutine adds to the specified subpicture file the commands necessary to display the identified text string, starting at the current beam position. These are in the format determined by whichever of the optional character generators (VC 38 or VA38) is used.

The standard text font is drawn on a 5 x 7 dot matrix. Each character causes an increment of 7 raster units to the X position of the beam.

a. Form

(1) FORTRAN: CALL TEXT (str, n [,pname])

(2) Macro-9

```
        .GLOBL      TEXT
        JMS*        TEXT
        JMP         .+4
        .DSA        str
        .DSA        n
       [.DSA        pname]
```

b. Input Variables

(1) 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).

(2) n is an integer number representing the total number of characters to be displayed.

(3) pname is the first location of this subpicture file, as in LINE, above.

c. Output Variables

pname = Current length of this file.

d. Description

(1) TEXT adds to the assembled display file two commands

plus the character string at two characters per word:

```
CHAR!EDS (character mode, enter data state)
C1   C2
 .
 .
 .
 .
Cn
Ce   (escape-to-control-state character)
```

(2) If this TEXT call immediately follows a previous TEXT call, TEXT inserts only the new character string above the escape character.

(3) The number of commands added to the file is added to the contents of pname.

e. Restrictions.

(1) Pname must contain zero (Ø) before the first call referencing it.

(2) Sufficient space must follow pname to contain the subpicture file.

f. Example

The following statements illustrate the manner in which text to be displayed is setup and called:

(1) Setup for "339␣ASSABET␣ALLEY"

```
DIMENSION ADDR(4)

DATA ADDR(1)/5H339␣A/,ADDR(2)/5HSSABE/,ADDR(3)/5HT␣ALL/,
1 ADDR(4)/2HEY/
```

(2) CALL statement to display text

```
CALL TEXT (ADDR(1),17,IPIC(1))
```

2.1.3 COPY Subroutine

The COPY subroutine enables two or more subpicture display files to be linked together to generate a composite display image. This subroutine adds to one subpicture display file the commands necessary to link it to a second subpicture with the second display file starting at the last beam position specified by the first subpicture.

a. Form

(1) FORTRAN: CALL COPY (pname1 [,pname])

(2) Macro-9:

```
.GLOBL      COPY
 JMS*       COPY
 JMP        .+3
.DSA        pname1
[.DSA       pname]
```

b. Input Variables

(1) pname1 is the first location (name) of the subpicture to be copied (integer).

(2) pname is the first location (name) of this subpicture display file, as in LINE above.

c. Output Variables

pname = Current length of this file.

d. Description

(1) COPY adds two commands to the display file:

```
PJMP    (display subpicture jump)
pname1+2
```

(2) 2 is added to the contents of pname1.

e. Restrictions

(1) Pname1 need not be defined when COPY is called, but must be a defined subpicture when pname is displayed.

(2) Pname must contain zero (Ø) before the first call referencing it.

(3) Sufficient space must follow pname to contain the subpicture file.

f. Example

The statement

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

adds a call to the WINDOW subpicture file to the file identified as HOUSE.

2.1.4 PRAMTR Subroutine

The PRAMTR subroutine allows the user to add to the specified subpicture file the commands necessary to set scale and intensity for the display or to turn on or off light pen sensitivity and the blink circuit.

The scale setting determines the proximity of intensified points in a line:

0	Every point is intensified
1	Every other point is intensified
2	Every fourth point is intensified
3	Every eighth point is intensified

This setting affects the length of the line proportionately, since the increments given in the LINE call are actually a count of intensified points rather than absolute points. It also affects the intensity, in inverse proportion, since fewer points are glowing. Thus if the intensity of a line must remain constant with a change in scale setting, the intensity setting must be changed by an equal amount. At a scale setting of 2, the points are clearly separated and the effect on intensity becomes secondary.

More than one feature, each with its corresponding settings, may be specified in a PRAMTR CALL statement by using the following technique:

(1) Add together the integer code numbers which identify the

selected features and assign this value to the variable featr.

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 (#1) and of 6 for intensity (#2)).

a. Form

(1) FORTRAN:

A. (one feature) CALL PRAMTR (featr,value [,pname])

B. (more than one feature and setting)

CALL PRAMTR (featr(s),value1[,value2...][,pname])

(2) Macro-9:

```
.GLOBL    MODE
JMS*      MODE
JMP       .+N      Where: N = 2+ (number of
.DSA      featr             features specified) +1
.DSA      value             if pname is given
.DSA      pname
```

b. Input Variables

(1) featr = Feature of the display being set (integer):

1 set scale
2 set intensity
4 set light pen sensitivity (on or off)
8 set blink (on or off)

(2) value = The value to which featr is set (integer):

Featr	Possible Values
1 (scale)	0 (low) - 3 (high)
2 (intensity)	0 (low)- 7 (high)
4 (light pen)	1 = on
	0 = off
8 (blink)	1 = on
	0 = off

(3) Pname is the first location of this subpicture file, as in LINE, above.

c. Output Variables

pname = Current length of this file.

d. Description

(1) The PRAMTR subroutine adds one or more commands to the display file depending on the type of argument list used:

SC value or
INT value or
LPON if value = 1, LPOF if value = 0, or
BKON if value = 1,. BKOF if value = 0

(2) 1 is added to the contents of location pname.

e. Restrictions

(1) This subroutine must be used with care, since the setting given is in effect until explicitly changed in this or some other part of the display. Thus, if the blink is turned on at the beginning of a subpicture, it must be turned off at the end in order that only that subpicture blinks. Otherwise the whole display will blink.

(2) Pname must contain zero (Ø) before the first call referencing it.

(3) Sufficient space must follow pname to contain the subpicture file.

f. Examples:

(1) The following single feature statement:

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

specifies an intensity (2) level of 7 for the subpicture display file starting at the first location of array HOUSE.

(2) The following multiple-feature statement:

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

specifies the values Ø and 4 for display SCALE and INT (intensity) features, turns the optional light pen on (LPEN = 4) for the execution of the subpicture display file which starts at location 1 of array IN.

2.1.5 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 at the value of each data point. The other coordinate is then automatically incremented by one (in the current scale), leaving the beam positioned one increment past the end of the graph. Note that axis and labelling must be provided separately.

a. Form:

(1) FORTRAN: CALL GRAPH (dta,n,a [,pname])

(2) Macro-9:

```
        .GLOBL    GRAPH
        JMS*      GRAPH
        JMP       .+5
        .DSA      dta
        .DSA      n
        .DSA      a
       [.DSA      pname]
```

b. Input Variables

(1) dta contains the set of data points, one per word, in the range 0 to 1023 (integer).

(2) n = number of data points to be displayed (integer).

(3) a indicates which axis to imcrement.
Ø = increment x, set y to data values
1 = increment y, set x to data values

(4) pname is the first location of this subpicture file, as in LINE, above.

c. Output variables

pname = current length of this file.

d. Description

(1) GRAPH adds to the display file one command plus the data set, at one value per word:

```
GRAPH!EDS   (graph mode, enter data state)
VAL1
VAL2
 .
 .
 .
VALn        (escape bit on)
```

(2) If this GRAPH call immediately follows a previous GRAPH call, the escape bit is cleared and the new data points only are added to the file. The escape bit is then set in the last of the new data points.

(3) The number of commands added to the file is added to the contents of pname.

e. Restrictions

General restrictions only (see LINE).

f. Example
The following program illustrates the use of the GRAPH and other subroutines.

```
C TEST PROGRAM FOR THE GRAPH SUBROUTINE
C FTST10
COPYRIGHT 1969, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
          INTEGER SINWV(300),Y(200)
          DIMENSION TITL(10),MAINFL(20)
          DATA TITL(1),TITL(2),TITL(3),TITL(4)/5HTHIS ,
          1 5HIS A ,5HSINE ,5HWAVE./
C SET UP INTEGER ARRAY OF VALUES TO BE PLOTTED
10        X=0.
          DO 20 I=1,200
          Y(I)=IFIX(SIN(X)*256.)+512
          X=X+.0628
20        CONTINUE
C SET UP SUB-PICTURE TO PLOT THOSE VALUES
          SINWV(1)=0
          CALL PRAMTR(3,0,7,SINWV(1))
          CALL LINE(0,450,1)
          CALL LINE(250,-450,0)
          CALL LINE(-500,0,1)
          CALL LINE(250,0,0)
          CALL PRAMTR(1,1)
          CALL GRAPH(Y(1),100,0)
          CALL GRAPH(Y(101),100,0,SINWV(1))
          CALL LINE(-20,-200,0)
          CALL TEXT(TITL(1),20)
```

```
C SET UP MAIN DISPLAY FILE TO DISPLAY THE GRAPH
      MAINFL(1)=0
      CALL DINIT(MAINFL(1))
      CALL SETPT(512,200)
      CALL PLOT(0,SINWV(1))
      PAUSE 1
      CALL DCLOSE(MAINFL(1))
      PAUSE 2
      GO TO 10
      END
```

2.1.6 BLANK Subroutine

The BLANK subroutine is used to prevent the displaying of any copy of this subpicture. The display file length is not changed.

a. Form

(1) FORTRAN: CALL BLANK (pname)

(2) Macro-9:

```
.GLOBL      BLANK
JMS*        BLANK
JMP         .+2
.DSA        pname
```

b. Input Variables

pname is the subpicture to be BLANKed.

c. Output Variables

None.

d. Description

(1) The command in location pname+2 (the first executable command in the display file) is moved to location pname+1.

(2) The command POP (display subpicture return) is inserted at location pname+2.

e. Restrictions

(1) Pname must be a defined subpicture file.

(2) BLANK has no meaning as the first call referring to pname.

f. Example

The following statement prevents the subpicture display file starting at the first location of array IPIC from being displayed.

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

2.1.7 UNBLNK Subroutine

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

a. Form

(1) FORTRAN: CALL UNBLNK (pname)

(2) Macro-9:

```
.GLOBL    UNBLNK
JMS*      UNBLNK
JMP       .+2
.DSA      pname
```

b. Input Variables

pname is the subpicture to be UNBLNKed.

c. Output Variables

None.

d. Description

The command in location pname+1 (placed there by a call to BLANK) is moved to location pname+2, overlaying the POP that was there.

e. Restrictions

(1) Pname must be a defined subpicture file.

(2) A call to UNBLNK must have been preceded by a call to BLANK.

f. Example

The following statement will enable the previously blanked subpicture IPIC (refer to "f." of 2.1.6) to be displayed.

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

2.2 MAIN DISPLAY FILE ROUTINES

These routines are used to generate the display main file, to which the display controller is directed when initiating display, and which is presumed to be calling upon the subpicture files generated with the routines described above. As is the case with subpicture files, storage used for the main file is supplied by the calling program as a dimensioned array. That array is identified by only one call to the initializer (DINIT) and is implicit in all other calls. These calls assume that reference is made to the storage identified by DINIT. They are concerned, however, with the identification of each entry to the main display file. Thus each call has as an optional argument the location of the display code generated by that particular call which provides a "handle" to that particular graphic entity. This provides the flexibility required to build and modify a display file in an interactive environment. It also enables the user to perform limited storage management. Each routine that adds commands to the file inserts a word count corresponding to this call, and each change or deletion of commands updates this local word count. Thus, small blocks of storage containing unneeded commands can be reused.

As with subpicture files, the name of a main display file is its first location, which contains the length of the file and is updated by each call that changes that length. If a new file is to be generated, the first location must contain zero before any display routines refer

to it. The remaining three words of storage overhead in a main file are the second word, reserved for control, and the last two words, which contain a JUMP to the beginning of the file. The local word counts keeping track of groups of commands are kept in the high-order 6 bits of the PDP-9 word, since the display uses only the low-order 12 bits.

2.2.1 DINIT (Display Initialize) Subroutine

The DINIT subroutine initializes the display via device number (.DAT slot) 10. The 339 Device Handler must be associated with .DAT slot 10 as DINIT contains .IODEV 10, which causes the device handler tied to .DAT slot 10 to be loaded. It 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.

a. Form

(1) FORTRAN: CALL DINIT (mainfl)

(2) Macro-9:

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

b. Input Variables

mainfl is the first location of the main display file. Like pname, it is an element of a dimensioned array (integer).

c. Output Variables

Location mainfl contains the length of the display file. This is updated by all Main File Routines.

d. Description

(1) Determine display length from contents of mainfl (new display file if mainfl = 0).

(2) Insert

```
JUMP
mainfl+2
```

at end of file.

(3) Initialize display.

(4) Start the display running at mainfl+2.

e. Restrictions

(1) If a new display file is being formed, location mainfl must contain zero. If this is a previously defined file, mainfl contains the file length and must not be altered.

(2) Sufficient storage must follow mainfl to accommodate the display file to be generated at any one time.

(3) Only one main display file can be running at any one time.

f. Example

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

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

2.2.2 DCLOSE (Display Terminate) Subroutine

The DCLOSE subroutine is used to end a display. It also converts the main file specified into a subpicture file by replacing the JUMP to the starting location by a POP command. This allows it to be used by any subsequently generated main file, as subpicture mainfl.

a. Form

(1) FORTRAN: CALL DCLOSE (mainfl)

(2) Macro-9:

```
.GLOBL      DCLOSE
JMS*        DCLOSE
JMP         .+2
.DSA        mainfl
```

b. Input Variables

mainfl is the starting location of the main display file being closed.

c. Output Variables

None.

d. Description

(1) Stop running the display device.

(2) Insert POP at the last location of the display file.

e. Restrictions

None.

f. Example

The following statement will end the display initiated by the statement given in the example "f." of 2.2.1.

```
CALL DCLOSE (MAINFL (1))
```

2.2.3 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.

a. Form

(1) FORTRAN: CALL SETPT (y,x[,cname])

(2) Macro-9:

```
.GLOBL      SETPT
JMS*        SETPT
JMP         .+4
.DSA        y
.DSA        x
[.DSA       cname]
```

b. Input Variables

(1) y = Vertical coordinate of beam location.

(2) x = Horizontal coordinate of beam location.

c. Output Variables

(1) cname = Location of the group of display commands generated by this call (Integer Variable).

d. Description

(1) SETPT adds three commands to the main file:

```
POINT!EDS    (point mode, enter data state)
y
x
```

(2) The number 03 is entered into the high-order six bits of the control state command (POINT!EDS).

(3) The value 3 is added to the contents of location mainfl.

(4) The location of the POINT!EDS is stored in cname (if given).

e. Restrictions

(1) The values x and y must be positive integers, and their values must not exceed 1023.

(2) This call 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.

(3) Cname is an optional output of this subroutine. Using the same variable name as one used in a previous call will destroy the previous contents.

f. Example

The following statement establishes an absolute beam position at matrix coordinates y=10, x=10 of the display.

```
CALL SETPT (1Ø,1Ø)
```

2.2.4 PLOT Subroutine

The PLOT subroutine is the prime active agent in the generation of the main display file. There are three forms of calls corresponding to the three subpicture routines, COPY, LINE, and PRAMTR. They are used to cause the displaying of predefined (but not necessarily complete) subpictures and individual lines, 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.

A. PLOT a Subpicture

a. Form

(1) FORTRAN: CALL PLOT (Ø,pname [,cname])

(2) Macro-9:

```
.GLOBL    PLOT
JMS*      PLOT
JMP       .+4
.DSA      (Ø
.DSA      pname
[.DSA     cname]
```

b. Input Variables

pname is the name (first location) of the subpicture to be displayed.

c. Output Variables

cname = Location of the group of display commands generated by this call.

d. Description

(1) Two commands are added to the main display file:

```
PJMP (display subpicture jump)
pname+2
```

(2) The number 02 is entered into the high-order six bits of the PJMP.

(3) The value 02 is added to the contents of mainfl.

(4) The location of the PJMP is stored in cname (if given).

e. Restriction

As in SETPT, cname is an output and multiple use of the same variable name will destroy previous contents.

f. Example

CALL PLOT (COPI,HOUSE(1),MAIN) where:

COPI has been assigned the integer value Ø by a DATA statement; HOUSE identifies the subpicture file to be displayed and MAIN is an optional variable by which this display may be referenced.

B. PLOT a Line (or reposition the beam)

a. Form

(1) FORTRAN: CALL PLOT (1,deltay,deltax,It[,cname])

(2) Macro-9:

```
.GLOBL      PLOT
JMS*        PLOT
JMP         .+6
.DSA        (I
.DSA        deltay
.DSA        deltax
.DSA        It
[.DSA       cname]
```

b. Input Variables

(1) deltay = Vertical displacement through which the beam is to move.

(2) deltax = Horizontal displacement through which the beam is to move.

(3) It = Line type.

1 visible

0 invisible

c. Output Variables

cname = Location of the group of display commands generated by this call.

d. Description

(1) Three commands are added to the main display file:

```
VEC!EDS     (vector mode, enter data state)
deltay      (intensify bit = LT)
deltax      (escape bit on)
```

(2) The number 03 is entered into the high-order six bits of the VEC!EDS.

(3) The value 03 is added to the contents of mainfl.

(4) The location of the VEC!EDS is stored in cname (if given).

e. Restrictions

(1) Deltay and deltax are signed decimal integers not to exceed 1023.

(2) As in SETPT, cname is an output and multiple use of the same variable name will destroy previous contents.

f. Example

```
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.

C. PLOT a Control Command

a. Form

(1) FORTRAN: CALL PLOT (2,featr,value[,cname])

(2) Macro-9:

```
.GLOBL    PLOT
JMS*      PLOT
JMP       .+5
.DSA      (2
.DSA      featr
.DSA      value
[.DSA     cname]
```

b. Input Variables

featr , value.

NOTE
The variables featr and value must be specified in the same manner as for PRAMTR statements. Refer to paragraph 2.1.4 for detailed information.

c. Output Variables

cname = Location of the group of display commands generated by this call.

d. Description

(1) This subroutine adds one or more commands to the display file depending on the type of argument list used:

SC value or
INT value or
LPON if value=1, LPOF if value=0 or
BKON if value=1, BKOF if value-0

(2) The number 01 or more is entered into the high-order six bits of the command.

(3) The value 01 or more is added to the contents of mainfl.

(4) The location of the command is stored in cname (if given).

e. Restrictions

(1) This call must be used with care, since the setting given is in effect until explicitly changed in this or some other part of the display. Thus if the blink is turned on before calling a subpicture, it must be turned off after the call in order that only that subpicture blinks. Otherwise, the whole display will blink.

(2) As in SETPT, cname is an output and multiple use of the same variable name will destroy previous contents.

f. Example

(1) Single-feature statement

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

(2) Multiple-feature statement

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

establishes the values Ø and 4 for display feature SCALE and INT, turns the light pen option (LPEN) on, and specifies the location IN as the first location of the array containing the display code generated by this statement (PRAM = 2).

2.2.5 DELETE Function

The function DELETE is used to delete from the main display file any display entity formed by a single call to a main file subprogram and assigned to cname. If cname does not point to the beginning of such a display entity, the DELETE fails and has no effect on the display file.

a. Form

(1) FORTRAN: i = DELETE (cname)

or CALL DELETE (cname)

(2) Macro-9:

```
.GLOBL      DELETE
JMS*        DELETE
JMP         .+2
.DSA        cname
DAC         i /if desired
```

b. Input Variables

cname = Location of the group of display commands to be deleted.

c. Output Variables

i = Boolean success indicator.

```
.TRUE.    successful deletion.
.FALSE.   unsuccessful deletion.
```

d. Description

(1) Check if cname points to a display entity; do nothing if not.

(2) Replace all commands in this group by display no-operation (NOP) commands.

(3) Insert sequential numbers in the high-order six bits of immediately adjacent free storage locations, starting from the bottom of this group and moving up in the display file.

e. Restrictions

None.

f. 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.

2.2.6 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, 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 LINE group. There are three forms of calls to REPLOT, each of which is the same as the corresponding call to PLOT.

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.

A. REPLOT a subpicture

a. Form

(1) FORTRAN: i = REPLOT (Ø,pname,cname)

or CALL REPLOT (copy,pname,cname)

(2) Macro-9:

```
.GLOBL   REPLOT
JMS*     REPLOT
JMP      .+4
.DSA     (Ø
.DSA     pname
.DSA     cname
DAC      i /if desired
```

b. Input Variables

(1) pname is the name (first location) of the subpicture to be displayed.

(2) cname = Location in which to store the display commands generated.

c. Output Variables

i = Logical success indicator.

.TRUE.	REPLOT successful.
.FALSE.	not enough room at location pointed to by cname, no action.

NOTE: If the above form is used, both i and REPLOT must be declared as LOGICAL in a type statement.

d. Description

(1) Check if cname points to a large enough block; do nothing if not.

(2) Insert two commands at the location pointed to by cname:

```
PJMP
pname+2
```

(3) The number 02 is entered into the high-order six bits of the PJMP.

(4) Subtract 2 from the sequential numbers in the high-order six bits of any adjacent free storage locations above this.

(5) Insert NOP's in any remaining locations belonging to a former command group at this address.

e. Restrictions

None.

f. Example

```
CALL REPLOT (Ø,SLIDE(M),NAME)
```

where Ø indicates the function (i.e. subpicture), M represents the first location of the subpicture display file (in array SLIDE) and NAME identifies the location in the display file into which this line is to be inserted.

B. REPLOT a line (or reposition the beam)

a. Form

(1) FORTRAN: `i = REPLOT (l,deltay,deltax,It,cname)`

or `CALL REPLOT (line,deltay,deltax,It,cname)`

(2) Macro-9:

```
.GLOBL    REPLOT
JMS*      REPLOT
JMP       .+6
.DSA      (l
.DSA      deltay
.DSA      deltax
.DSA      It
.DSA      cname
DAC       i /if desired
```

b. Input Variables

(1) deltay = Vertical displacement through which the beam is to move.

(2) deltax = Horizontal displacement through which the beam is to move.

(3) It = Line type.

1 visible
0 invisible

(4) cname = Location in which to store the display commands generated.

c. Output Variables.

i = Logical success indicator.

.TRUE.	REPLOT successful.
.FALSE.	not enough room at location pointed at by cname, no action.

NOTE: If the above form is used, both i and REPLOT must be declared as LOGICAL in a type statement.

d. Description

(1) Check if cname points to a large enough block, do nothing if it does not.

(2) Insert three commands at the location pointed to by cname:

VEC!EDS	(vector mode, enter data state)
deltay	(intensify bit = It)
deltax	(escape bit on)

(3) The number 03 is entered into the high-order six bits of the VEC!EDS.

(4) Subtract the value 3 from the sequential numbers in the high order six bits of any adjacent free storage locations above this.

(5) Insert NOPs in any remaining locations belonging to a former command group at this address.

e. Restrictions

Deltay and deltax are signed decimal integers not to exceed 1023.

f. Example

```
CALL REPLOT (LYNE,Y,X,OFF,DISPL)
```

where the first four arguments are assigned the values 1, 50, 50 and 0 respectively, causing the beam to be moved, not intensified, up 50 raster units and to the right 50 raster units. DISPL identifies the location in the display file into which this line is to be inserted.

C. REPLOT a Control Command

a. Form

(1) FORTRAN: i = REPLOT (2,featr,value,cname) or

CALL REPLOT (mode,featr,value,cname)

(2) Macro-9:

```
.GLOBL      REPLOT
JMS*        REPLOT
JMP         .+5
.DSA        (2
.DSA        featr
.DSA        value
.DSA        cname
DAC         i  /if desired
```

b. Input Variables

featr, value.

NOTE: The variables featr and value must be specified in the same manner as for PRAMTR statement. Refer to paragraph 2.1.4 for detailed information.

c. Output Variable

i = Logical success indicator.

.TRUE.	REPLOT successful.
.FALSE.	not enough room at location pointed to by cname, no action.

NOTE: If the above form is used, both i and REPLOT must be declared LOGICAL in a type statement.

d. Description

(1) Check whether cname points to a large enough block (i.e., to a free location); do nothing if it does not.

(2) The REPLOT subroutine adds one or more commands to the display file depending on the type of argument list used:

SC value or
INT value or
LPON if value = 1, LOPF if value = 0 or
BKON if value = 1, BKOF if value = 0

(3) The number 01 or more is entered in the high-order six bits of that command.

(4) Subtract 1 or more from the sequential numbers in the high-order six bits of any adjacent free storage locations above this.

e. Restrictions

This call must be used with care, since the setting given is in effect until explicitly changed in this or some other part of the display. Thus if the blink is turned on before calling a subpicture, it must be turned off after the call in order that only that subpicture blinks. Otherwise the whole display will blink.

f. Example

```
CALL REPLOT (PRAM,BLINK,I,IBL1)
```

where the first two arguments are assigned the values 2 and 8 respectively, I may represent either ON or OFF, and IBL1 identifies the location in the display file into which this line is to be inserted.

2.2.7 RSETPT (Set Point) Subroutine

The function RSETPT permits absolute beam locations previously defined by SETPT to be redefined. This function 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 as is done by REPLOT takes place in RSETPT.

a. Form

(1) FORTRAN: `i = RSETPT (y,x,cname)` or

`CALL RSETPT (y,x,cname)`

(2) Macro-9:

```
.GLOBL    RSETPT
JMS*      RSETPT
JMP       .+4
.DSA      y
.DSA      x
.DSA      cname
DAC       i    /if desired
```

b. Input Variables

(1) y = Vertical coordinate of beam location.

(2) x = Horizontal coordinate of beam location.

(3) cname = Location in which to put the display commands that are generated.

c. Output Variables

i = Logical success indicator

```
.TRUE.    RSETPT successful.
.FALSE.   Not enough room at location
          pointed to by cname, no action.
```

NOTE: If the above form is used, both i and REPLOT must be declared LOGICAL in a type statement.

d. Description

(1) Check if cname points to a large enough block, do nothing if it does not.

(2) Insert three commands at the location pointed to by cname:

```
POINT!EDS
  y
  x
```

(3) The number 03 is entered into the high-order six bits of the POINT!EDS.

(4) Subtract 3 from the sequential numbers in the high-order six bits of any adjacent free storage locations above this.

(5) Insert NOP's in any remaining locations belonging to a former command group at this address.

e. Restrictions

(1) x and y must be positive integers, the value of which must not exceed 1023.

(2) This call causes the beam to be given an absolute location, as opposed to a relative displacement. This 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.

f. Example

```
CALL RSEPT(1Ø,1Ø,NAME)
```

where the value of 1Ø is assigned to the x and y coordinates and NAME identifies the location into the display file into which this line is to be inserted.

2.3 INPUT SUBPROGRAMS

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

2.3.1 LTPN Functions

The function LTPN is used to determine whether a light pen hit has occurred. If it has not, the function returns an indicator to this

effect. If it has, the logical (name of the display element hit) 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 item "f"). The IF could branch to itself if no hit has occurred, or to the user's light pen hit processing code if a hit has occurred.

a. Form

(1) FORTRAN: `i = LTPN(y,x,cname,pb)`

NOTE: The variable i, LTPN, and pb must be declared LOGICAL in a type statement.

(2) Macro-9:

```
.GLOBL    LTPN
JMS*      LTPN
JMP       .+5
.DSA      y
.DSA      x
.DSA      cname
.DSA      pb
DAC       i
```

b. Input Variables

None

c. Output Variables

(1) i = Logical indicator.

.TRUE. a light pen hit has occurred.
.FALSE.[1] no light pen hit has occurred.

(2) y = Vertical coordinate of light pen position. ($0 \leq y \leq 1023$ raster units).

(3) x = Horizontal coordinate of light pen position. ($0 \leq x \leq 1023$ raster units).

(4) cname = Location in the main display file of the group of commands causing this entity to be displayed.

[1] If i = .FALSE. items (2) through (5) have no meaning.

(5) pb represents an array each element of which will contain either the logical .T. or .F. corresponding to ON or OFF for each of the 12 control box push-buttons.

d. Description

(1) Issue a read on light pen interrupt to the display device handler.

(2) Return if no interrupt was posted.

(3) Read display registers and translate into display file locations.

e. Restrictions

Care must be taken when using this routine in conjunction with the Main Display File routines. Because of conflicting requests to the 339 Device Handler, the Main File routines clear the request for pending light pen interrupts. Thereafter, until LTPN has been called again, a light pen interrupt is not recognized. Thus, in the sequence

```
          .
          .
          .
A         I = LTPN(...)
          .
          .
          .
B         CALL REPLOT(...)
          .
          .
          .
C         I = LTPN(...)
```

a light pen interrupt occurring between A and B will set I = .TRUE., but will have no effect on the segment between B and C.

f. Example

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

```
IF(LTPN(LPY,LPX,NAME,PB)) GO TO 1ØØ
```

In the above statement, if a hit has occurred (LTPN is .TRUE.) the y and x coordinates of the hit, the name of the display element detected and the status of the pushbutton control box are indicated and the program is directed to execute statement 1ØØ.

2.3.2 PBTN Function

The function PBTN is used to determine whether a pushbutton or the manual interrupt button has been depressed. This function works like LTPN, in that it answers the question, "Has the expected action taken place?" The user program then decides what to do, based on the answer.

a. Form

(1) FORTRAN: i = PBTN (pb)

(2) Macro-9:

```
.GLOBL    PBTN
JMS*      PBTN
JMP       .+2
.DSA      pb
DAC       i
```

b. Input Variables

None.

c. Output Variables

(1) i = Logical indicator.

.TRUE. = a button was pushed.
.FALSE. = no button was pushed.

(2) pb represents an array each element of which will contain either the logical .T. or .F. corresponding to ON or OFF for each of the 12 control box push-buttons.

d. Description

(1) Issue a read on pushbutton or manual interrupt to the display device handler.

(2) Return if no interrupt was posted.

(3) Read pushbutton status register.

e. Restrictions

Care must be taken when using this routine in conjunction with the Main Display File routines. Because of conflicting requests to the 339 Device Handler, the Main File routines clear the request for pending pushbutton interrupts. Thereafter, until PBTN has been called again, a pushbutton interrupt is not recognized. Thus in the sequence

```
A       I = PBTN(PB)
          .
          .
          .
B       CALL REPLOT(...)
          .
          .
          .
C       I = PBTN(PB)
```

a pushbutton interrupt occurring between A and B will set I = .TRUE., but will have no effect on the segment between B and C.

2.3.3 TRACK Subroutine

The TRACK subroutine is used for light pen tracking. Calling TRACK causes the proper linkage to be set up to the main display file, and a tracking pattern is displayed which will follow the motion of the light pen. This routine remains in control, displaying and moving the tracking pattern, until any pushbutton or the manual interrupt button is depressed. At this point, control is returned to the user program with the physical location of the tracking pattern indicator point and the status of the pushbutton box. The tracking pattern

will continue to be displayed, but it will not track the light pen until TRACK is called again. If desired, the user can DELETE the tracking pattern; he can also BLANK or UNBLNK the display file TRCK if he has declared EXTERNAL or .GLOBL TRCK. When calling TRACK, the starting location of the tracking pattern can be specified, and the indicator point can be constrained against horizontal movement, vertical movement, or any movement (to move the tracking pattern away from the point), by setting the appropriate starting coordinate negative.

a. Form

(1) FORTRAN: CALL TRACK (±y,±x,cname,pb)

I = TRACK(±y,±x,CNAME,pb)

(2) Macro-9:

```
.GLOBL      TRACK
.JMS        TRACK
JMP         .+5
.DSA        y
.DSA        x
,DSA        cname
.DSA        pb
.DAC        I
```

b. Input Variables

(1) y = Vertical coordinate in raster units of the tracking pattern initial position. If negative, the indicator point is constrained to horizontal motion at this vertical coordinate.

(2) x = Horizontal coordinate in raster units of the tracking pattern initial position. If negative, the indicator point is constrained to vertical motion at this horizontal coordinate.

(3) cname = Location in the main display file of the group of commands causing the tracking pattern to be displayed.

0 Add those commands to the display file.

non-0 Those commands are in the file and pointed to by cname (if cname contains any other value the tracking pattern commands are not inserted and I is set to .FALSE.)

c. Output Variables

(1) I = Logical success indicator.

.TRUE. = TRACK was inserted.
.FALSE. = cname points to some unknown location.

(2) y = Present position vertical coordinate in raster units of the tracking pattern indicator point. If input y was negative (constrained), neither sign nor magnitude has changed.

(3) x = Present position horizontal coordinate in raster units of the indicator point. If input x was negative (constrained), neither sign nor magnitude has changed.

(4) cname = Location in the main display file of the group of commands causing the tracking pattern to be displayed.

(5) pb = represents an array each element of which will contain either the logical .T or .F corresponding to ON or OFF for each of the 12 control box pushbuttons.

d. Description

(1) Insert starting position and constraints in tracking pattern display file.

(2) If input cname = 0

(a) Five commands are added to the main display file:

```
PJMP
TRCK
POINT!EDS        These commands return the
Y                beam to the main part of
X                the display.
```

(b) The number 05 is entered into the high-order six bits of the PJMP.

(c) The value 5 is added to the contents of the first location of the main display file, mainfl.

(d) The location of the PJMP is stored in cname.

If input cname points to a PJMP TRCK, go to item (3).

If input cname points to anything else, set I = .FALSE. and return to user.

(3) Issue a read on light pen, pushbutton, or manual interrupt to the display device handler.

(4) If a light pen hit occurs on the tracking pattern, move it in the appropriate direction and return to item (3).

(5) If a pushbutton or manual interrupt occurs, read the indicator point position and the pb values and return to the user program.

e. Restrictions

The values y and x are decimal integers not to exceed 1023 in magnitude.

f. Example

```
CALL TRACK (Y2,X2,NAME(1),PB)
```

2.4 RELOCATABLE DISPLAY FILES

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. 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. Note that a display file cannot be displayed after having been processed by DYSET; DYLINK must be used to return it to displayable form.

2.4.1 DYSET Subroutine

The DYSET Subroutine converts subpicture calls to a symbolic form independent of core memory location, using specified ASCII strings.

a. Form

(1) FORTRAN: CALL DYSET (PNAME1,ASCII1,...,PNAMEN,ASCIIN)

(2) Macro-9:

```
.GLOBL    DYSET
JMS*      DYSET
JMP       2*N+.+1   } Where N is the
.DSA      PNAME.    } number of display
.DSA      ASCII1    } files in the call.
    .
    .
    .
.DSA      PNAMEN
.DSA      ASCIIN
```

b. Input Variables:

(1) The PNAMEs are the first locations of the inter-dependent display files, both calling and called.

(2) The ASCIIs are the names of real arrays containing 9 characters of 5/7 IOPS ASCII (which may be used for file names on output).

c. Output Variables:

None.

d. Description:

(1) DYSET searches each listed display file (PNAME) for PJMPs. When it finds one, it appends the ASCII name of the file being called to the file being searched, if that name is not already there.

(2) The operand of the PJMP is made a relative pointer to the ASCII name of the called display file.

(3) The contents of PNAME (the display file length) are increased by 4 each time an ASCII name is appended to the file.

e. Restrictions:

(1) Space provided for a display file must include 4 locations for each subpicture call it makes, if it is to be set up for relocation.

(2) Display commands must not be added to a display file nor can it 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.)

(3) 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.

2.4.2 DYLINK Subroutine

The DYLINK Subroutine converts ASCII display file names to subpicture calls on the corresponding display files.

a. Form

(1) FORTRAN: CALL DYLINK (PNAME1,ASCII1,...,PNAMEN,ASCIIN)

(2) Macro-9:

```
.GLOBL      DYLINK
JMS*        DYLINK
JMP         2*N+.+1
.DSA        PNAME1
.DSA        ASCII1
  .
  .
  .
.DSA        PNAMEN
.DSA        ASCIIN
```

Where N is the number of display files in the call.

b. Input Variables

(1) The PNAMEs are the first locations of the interdependent display files, both calling and called.

(2) The ASCIIs are the names of real arrays containing 9 characters of 5/7 IOPS ASCII (which may be used for file names on output).

c. Output Variables

None.

d. Description

(1) DYLINK searches each listed display file for PJMPs. 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 PJMP.

(2) The operand of the PJMP is replaced by the address of the corresponding subpicture file, obtained from the argument list.

(3) The contents of PNAME (the display file length) are reduced to the actual number of display commands in the file (excluding the ASCII blocks).

e. Restrictions

(1) 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.

f. Example

(1) Display files have been created in integer arrays ID1 and ID2. ID1 uses files starting at ID2(1) and ID2(47) as subpictures (via calls to COPY). It is required to write the files out relocatably, on logical unit number 5.

```
DIMENSION TITL1(2), TITL2(2), TITL3(2)
DATA TITL1(1), TITL1(2)/5HPICT1, 4H BIN/,
1 TITL2(1), TITL2(2)/5HPICT2, 4H BIN/,
2 TITL3(1), TITL3(2)/5HPICT3, 4H BIN/
        .
        .
        .
CALL DCLOSE (ID1(1))
CALL DYSET (ID1(1),TITL1,ID2(1),TITL2,ID2(47),TITL3)
CALL ENTER (5,TITL1)
J=ID1(1)+1
WRITE (5) (ID1(1), I=1,J)
CALL CLOSE (5,TITL1)
CALL ENTER (5,TITL2)
WRITE (5) (ID2(1), I=1,46
CALL CLOSE (5,TITL2)
CALL ENTER (5,TITL3)
J=ID2(47)+1
WRITE (5) (ID2(1), I=47,J)
CALL CLOSE (5,TITL3)
        .
        .
        .
```

(2) These display files are now to be input from unit 5 into array IA. The same DIMENSION and DATA statements as shown in (1) are required.

```
        .
        .
        .
CALL SEEK (5,TITL1)
READ (5)J,(IA(I+1), I=1,J)
ID (1)=J
CALL CLOSE (5,TITL1)
CALL SEEK (5,TITL2)
K=J+2
READ (5)J,(IA(I+K), I=1,J)
IA(K)+J
CALL CLOSE (5,TITL2)
CALL SEEK (5,TITL3)
L=J+2
READ (5)J,(IA(I+L),I+1,J)
IA(L)=J
CALL CLOSE (5,TITL3)
CALL DYLINK (IA(1),TITL1,IA(K),TITL2,IA(L),TITL3)
CALL DINIT (IA(1))
```

CHAPTER 3

SYSTEM I/O DEVICE HANDLER

The 339 Graphic Display Device Handler provides an interface between the user and the hardware which conforms to the conventions of the Input/Output and Keyboard Monitor Systems, as described in DEC manual Monitors, Advanced Software System, Doc. No. DEC-9A-MADØ-D. 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-9 and the display controller. To start up a display, the user generates a display file consisting of display commands as described in the Programmed Buffered Display Type 339 User's Handbook, 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.

3.1 LEGAL FUNCTIONS

3.1.1 .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 five words of standard settings contained in the handler. The user may optionally substitute his own settings for the latter four of these. The first of the five

words contains the location of the pushdown pointer list, which is supplied by the Monitor. This location is returned to the user by the .INIT call.

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

a. Form

.INIT a,f,r

b. Variables

a = Device Assignment Table (.DAT) slot number (in octal).

f = Initialization flag:

0 use standard display initialization.
1 user's initialization is pointed to by r.

r[1] = Optional pointer to user's initialization settings.

c. Expansion

LOC	CAL	$=f_{7-8}+a_{9-17}$
LOC+1		1
LOC+2		r
LOC+3		n /location of pushdown pointer list.

[1] If f=1, r points to a block of four words containing initial settings for (a) Display status, (b) Pushbuttons 0 through 5, (c) Pushbuttons 6 through 11, (d) Character generator. If the character generator word is Ø, the standard setting described in sub-paragraph d. (1) (e) and the system character tables are used. For a detailed description of these settings refer to 3.1.1.1, items a, b, and c.

d. Description

(1) Initialize display (as described for the IOT's presented in a. through c. of 3.1.1.1); normal settings are:

(a) Pushdown Pointer points to the location of the Pushdown Pointer List in the Monitor area.
(b) Display status is set to
1. disable edge flag interrupts
2. enable light pen interrupts
3. reenable light pen one data request after hit
4. allow full 13 bit x and y beam position registers
5. enable the intensification bit in data state
6. inhibit edge flags
7. enable pushbutton interrupts
8. enable internal stop interrupts
(c) Pushbuttons 0 through 5 are set to 0 (off).
(d) Pushbuttons 6 through 11 are set to 0 (off).
(e) Character generator is set to
1. lower case
2. 6-bit code
3. high-order six bits of the character routines' starting address, within the Monitor. This assumes the use of the optional VC38 Character Generator. If the VA38 Character Generator is used, no special settings are required and Step (e) is omitted.

(2) Return Pushdown Pointer list location to LOC+3.

(3) Connect handler to PIC or API.

(4) Clear the read-busy switch.

NOTE: If f=Ø and r=1, clearing the read-busy switch is the only action taken by the handler.

3.1.1.1 INITIALIZATION SETTINGS - The standard settings and character tables for IOTs , SIC, LBF, and SCG are those used for the initialization of the display. A complete description of each of these IOTs is given in the following:

a. SIC 700665 Set Initial Conditions - SIC sets up a number of status registers in the display. The instruction enables four display flags onto the interrupt line, sets

the page size to 10, 11, 12, or 13 bits in x and y and light pen conditions. One of three options is available in the event the display is resumed after a light pen hit. The light pen can be left on; it can be turned off completely; or it can be turned off until the completion of the present command, then automatically turned back on at the next data request. A register tells the display to ignore all edge flags; therefore, when the position register overflows, the edge flag is inhibited and the display continues in a normal fashion. Another register overrides the intensification bit in data state, causing all beam movements to be intensified. This feature is used principally for diagnostic purposes.

SIC

Edge Interrupt	L.P. Interrupt	L.P. Resume Options		Y Dimension		X Dimension		Intensify All Points	Inhibit Edge Flags	P.B. Interrupt	Internal Stop Interrupt
6	7	8	9	10	11	12	13	14	15	16	17

Bits	Interpretation
6	Enable edge flag interrupt
7	Enable light pen flag interrupt
8	If bit is a 0, do not disable light pen after the resume; if bit is a 1, bit 9 indicates when to reenable the light pen
9	If bit is a 0, reenable light pen on the first data request after the display is resumed. If bit is a 1, the light pen hit is equivalent to a LPOF command.
10,11	Set y dimension 00: 9.375 in. (10 bits) 01: 18.75 in. (11 bits) 10: 37.5 in. (12 bits) 11: 75.0 in. (13 bits)
12,13	Set x dimension, same as y
14	Intensify all points
15	Inhibit edge flags
16	Enable interrupt on pushbutton hit
17	Enable interrupt on internal stop flag

b. LBF 700705 Load Break Field - This instruction has two functions. First, it loads the break field register when initializing the display; second, it sets the pushbuttons. Both functions have enable bits so that one may be executed without the other. If neither enable bit is up, the IOT pulses have other meanings (STPD-700704 and SPES-700701).

Break Field			Pushbuttons								
6	7	8	9	10	11	12	13	14	15	16	17

Bits	Interpretation
6	Enable change of break field
7,8,9	New break field
10	Enable change of pushbuttons
11	If bit is a 0, set pushbuttons 0-5 according to AC bits 12-17; if bit is a 1, set pushbuttons 6-11 according to AC bits 12-17.
12-17	New pushbutton states.

b. SCG 700743 Set Character Generator -SCG sets the SAR, Case and CHSZ.

Spare			Case	CHSZ	Spare	SAR					
6	7	8	9	10	11	12	13	14	15	16	17

Bit(s)	Interpretation
6,7,8	Spare
9	Set case 0-lower 64 1-upper 64
10	Set code size 0-6 bit character format 1-7 bit character format
11	Spare
12-17	Starting address register

3.1.2 .READ Macro

The .READ macro is used for input to the user program from the hardware registers of the display controller. It provides the capability to read specified display registers at any time, or in response to specified interrupts. 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 grouping of flags and registers. This is done with an optional descriptive word following the .READ macro. The first six bits of that word indicate which interrupts are of interest, and the next 10 indicate the registers to read if any of those interrupts are set.

.READ results in loading into the user-specified buffer a descriptive word of the same form as the optional argument, with only one of the first six bits turned on to indicate which interrupt flag actually was set. The next 10 bits indicate which registers were read, and the contents of those registers are stored in the following words of the buffer. Only one .READ is serviced at a time. If no interrupts are specified, results are returned immediately and another .READ can be honored. If any interrupts are specified, results are not returned until an interrupt has occurred. The user specifies an interrupt flag count (i.e. the number of set flags specified to be of interest); the .READ will accumulate this flag count before it finishes entering the specified number of descriptive-word data groups into the buffer. Note that more than one flag may be set at interrupt time, depending on which flags have been enabled onto the interrupt line. A .READ given in the interim has the effect of a .WAIT, that is, the user program waits until the first .READ is completed, then accepts the second one and proceeds.

a. Form

.READ a, m, l, w

nstd /optional word describing non-standard groups

b. Variables

a = .DAT slot number.

m = Type of read

0 Read PDP, S1, S2, PB now (no interrupt).
1 Read PDP, XP, YP, DAC, PB if light pen interrupt flag is set.
2 Read PDP, XP, YP, DAC, PB if manual interrupt flag is set.
3 Read PDP, XP, YP DAC, PB if pushbutton interrupt flag is set.
4 Read PDP, XP, YP, DAC, PB if edge flag interrupt flag is set.
5 Read PDP, XP, YP, DAC, PB if internal stop interrupt flag is set.
6 Read PDP, XP, YP, DAC, PB if external stop interrupt flag is set.
7 nstd specifies registers and interrupt flags as follows:

bit 0 on service light pen interrupt.
bit 1 on service manual interrupt.
bit 2 on service pushbutton interrupt.
bit 3 on service edge flag interrupt.
bit 4 on service internal stop interrupt.
bit 5 on service external stop interrupt.
bit 6 on read Pushdown Pointer (PDP).
bit 7 on read X-position register (XP).
bit 8 on read Y-position register (YP.
bit 9 on read Display Address Counter (DAC).
bit 10 on read Status 1 (S1).
bit 11 on read Status 2 (S2).
bit 12 on read Pushbuttons (PB).
bit 13 on read Slave Group 1 (SG1).
bit 14 on read Slave Group 2 (SG2).
bit 15 on read Character Generator (CG).
bit 16 on no registers are read.

l = Return buffer address

C(l) = 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 = Interrupt count, number of interrupt occurrences this .READ will service before finishing.

c. Expansion

```
LOC     CAL+m6-8 +a9-17
LOC+1       10
LOC+2       1
            .DEC
LOC+3       -w   /decimal
LOC+4       nstd
```

d. Description

(1) Wait for previously requested interrupt(s) to occur.

(2) Load user's previous .READ buffer with descriptive word and data.

(3) Determine interrupts to service and registers for current .READ.

(4) Turn on Input Busy flag.

3.1.3 .WRITE Macro

The macro .WRITE 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 restart the display if it has been stopped. A .WRITE to the display is completed immediately and requires no waiting, since this only involves issuing IOTs to the display. Executing a list of display commands is a continuing process and does not end until the user specifically ends it.

a. Form

```
.WRITE  a, m, l, w
```

b. Variables

a = .DAT slot number (octal).

m = Type of write.

0 restart display (L not required).
1 resume display after internal stop.

NOTE: The display is automatically resumed after light pen or edge violation interrupts.

2 stop display (issue external stop).
4 start display pointed at by l.

l = Display file starting address.

w = Not used by this macro.

c. Expansion

```
LOC     CAL+m6-8 +a9-17
LOC+1       11
LOC+2       l
                .DEC
LOC+3       -w    /decimal
```

d. Description

Determine function requested and issue corresponding IOTs.

It should be noted that there are no restrictions regarding the manner in which the user may generate a display file. He is responsible for its contents and meaning, as well as such details as ending the file with a JUMP to its beginning to keep the image refreshed. One problem that frequently occurs is that of storage for display files, which tend to become large very rapidly. A solution to this problem is the use of a bulk storage device for temporary storage of these files. For example, a file-oriented device, such as DECtape, could be used very conveniently to store

display files by name; these files would then be read into core storage immediately before displaying. Each such file could be generated and then written on a DECtape by the sequence

```
.INIT DECtape
.ENTER filename
.WRITE (in Dump Mode)
.CLOSE DECtape
```

This procedure uses Dump mode in order to leave the display file in the same format in which it was generated. Retrieval of the file would be accomplished by the sequence

```
.INIT DECtape
.SEEK filename
.READ (in Dump Mode)
.CLOSE DECtape
```

A .WRITE to the display device handler, using the same buffer pointer, would then start the display running.

Another approach would be to store several display files as one storage file. In this case, the .INIT DECtape and ENTER filename would refer to the storage file rather than an individual display file. A .WRITE, in Dump mode, would be given for each display file to be output. Such an approach requires the user to keep his own word counts (which could be done by creating a word, containing the length of the display file, at the beginning of each file). Retrieval, in this case, is performed by an .INIT DECtape and a .SEEK filename, followed by a sequence of

```
.READ count word
.READ number of words specified by count
      word into display buffer
```

for each display file making up the storage file.

3.1.4 .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 specified number of interrupts 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.

a. Form

.WAIT a

b. Variables

a = .DAT slot number (octal).

c. Expansion

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

d. Description

(1) Allow previous .READ to complete.

(2) Load user's previous .READ buffer with descriptive word and data.

(3) Turn off Input Busy flag.

3.1.5 .WAITR Macro

The .WAITR macro is used for the same purpose as the .WAIT macro, also allowing the user program to proceed in line if the previous .READ is complete. If it 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.

a. Form

.WAITR a, addr

b. Variables

a = .DAT slot number (octal).

addr = location in the user program to branch to if input is not complete.

c. Expansion

LOC	CAL	$+1000+a_{9-17}$
LOC+1		12
LOC+2		addr

d. Description

(1) Branch on .READ incomplete to addr.

(2) Load user's previous .READ buffer with descriptive word and data.

(3) Turn Input Busy flag off.

3.1.6 .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.

a. Form

.CLOSE a

b. Variables

a = .DAT slot number (octal).

c. Expansion

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

d. Description

End current display.

3.2 IGNORED FUNCTIONS

The following System I/O macros are ignored by the 339 Display Device Handler:

a. .DLETE

b. .RENAM

c. .FSTAT

d. .ENTER

e. .CLEAR

f. .MTAPE

g. .SEEK

h. .TRAN

3.3 LEGAL DATA MODES

a. Output - The data output to the display must be a valid display file, consisting of display commands in both control state and data state, as described in the Programmed Buffer Display Type 339 User's Handbook. The handler does no data conversion.

b. Input - Input from the display consists of the contents of the various hardware registers in the display controller. These registers are described in the Programmed Buffer Display Type 339 User's Handbook.

APPENDIX A

SAMPLE PROGRAMS

This appendix contains three self-annotated listings of 339 Graphics display programs written in FORTRAN IV using the routines described in this manual.

HOUSE DISPLAY PROGRAM

The following program listing contains the Subpicture and Main Display File programs required to generate the "house", image shown in Figure A-1.

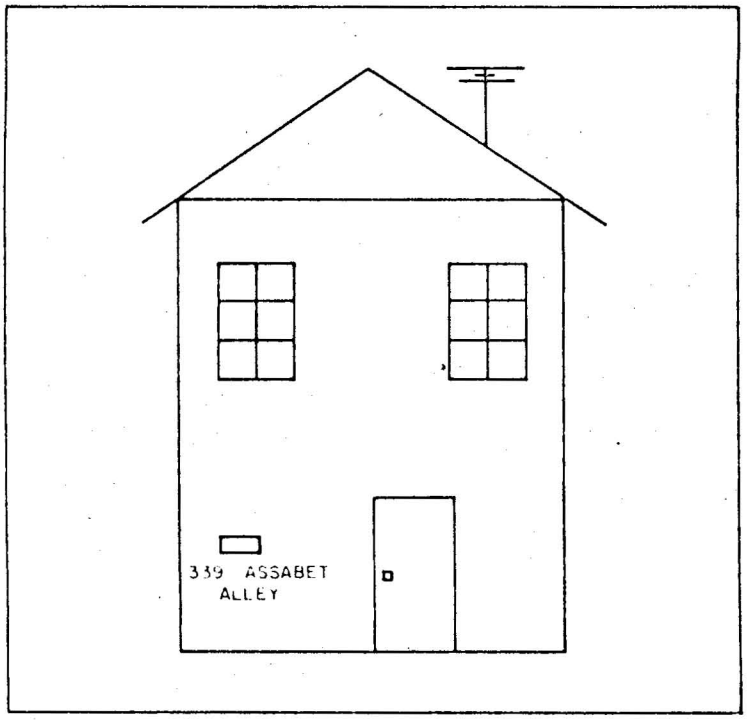


Figure A-1, House Display Image

```
C SAMPLE FORTRAN PROGRAM USING SUBPICTURE,MAIN FILE AND INPUT ROUTINES
C FNTSTG  - TO TEST DISPLAY FILE LOAD/UNLOAD -
COPYRIGHT 1968 DIGITAL EQUIPMENT CORP.,MAYNARD, MASS.
      INTEGER WINDOW,HOUSE
      INTEGER ON,OFF,SCALE,BLINK,COPI,Y,X
      INTEGER MAINFL,EDGE,DISPL,BL1,PRAM
      LOGICAL PB,PRTN,LTPN
      INTEGER A (40),B(120),C(40)
      DIMENSION SAM1(2), SAM2(2), SAM3(2)
      DIMENSION WINDOW(40),HOUSE(120),ADDR(4),HOUPT(2),WINPT(2)
      DIMENSION MAINFL(40),EDGE(4),PB(12)
      DATA SAM1(1), SAM1(2)/5HDISPL,4H1BIN/,
     1 SAM2(1),SAM2(2)/5HDISPL,4H2BIN/,
     2 SAM3(1), SAM3(2)/5HDISPL,4H3BIN/
      DATA ADDR(1)/5H339 A/,ADDR(2)/5HSSAGE/,ADDR(3)/5HT ALL/,
     1ADDR(4)/2HEY/,
     2 HOUPT(1),HOUPT(2)/5HHOUSE,4H BIN/,WINPT(1),WINPT(2)/
     3 5HWINDO,4HWBIN/
      DATA LYNE,ON,SCALE/3*1/,PRAM,INT/2*2/,OFF,COPI/2*0/,BLINK
     1 /0/,Y,X/2*50/,LPEN/4/
C CLEAR FIRST LOCATIONS OF WINDOW AND HOUSE TO INDICATE NEW FILES.
      WINDOW(1)=0
      HOUSE(1)=0
C DRAW VERTICAL BARS OF WINDOW (ORIGIN=UPPER LEFT CORNER)
5     DO 10 I = 1,3
      CALL LINE (-150,0,1,WINDOW(1))
      IF (I.EQ.3) GO TO 20
      CALL LINE (150,50,0)
10    CONTINUE
C DRAW HORIZONTAL BARS OF WINDOW (RETURNS TO ORIGIN)
20    DO 30 I = 1,4
      CALL LINE (0,-100,1)
      IF (I.EQ.4) GO TO 40
      CALL LINE (50,100,0)
30    CONTINUE
C ORIGIN OF HOUSE=LOWER LEFT CORNER
40    CALL LINE (0,500,1,HOUSE(1))
      CALL LINE (583,0,1)
      CALL LINE (0,-500,1)
      CALL LINE (-583,0,1,HOUSE(1))
C POSITION BEAM FOR LETTERS
      CALL LINE (100,25,0)
C DRAW 17 CHARACTERS FROM ADDR. CHARACTERS ARE PLOTTED ON
C     A 5*7 SPOT MATRIX WITH 2 RASTER UNITS BETWEEN. 17*7=
C     119 RASTER UNITS OF BEAM DISPLACEMENT IN X.(0 IN Y)
      CALL TEXT (ADDR(1),10)
      CALL TEXT(ADDR(3),7,HOUSE(1))
C CENTER MAILBOX OVER ADDRESS
      CALL LINE (50,-25,0)
      CALL LINE (0,-50,1)
      CALL LINE (25,0,1)
      CALL LINE (0,50,1)
      CALL LINE (-25,0,1)
C DRAW DOOR
      CALL LINE (-150,117,0)
      CALL LINE (200,0,1)
```

```
            CALL LINE (0,100,1)
            CALL LINE (-200,0,1)
            CALL LINE (95,-95,0)
            CALL LINE (10,0,1,HOUSE(1))
            CALL LINE (0,10,1)
            CALL LINE (-10,0,1)
            CALL LINE (0,-10,1)
C DRAW WINDOWS
            CALL PRAMTR(INT,7,HOUSE(1))
            CALL LINE (405,-205,0)
            CALL COPY (WINDOW(1))
            CALL LINE (0,300,0)
            CALL COPY (WINDOW(1))
            CALL PRAMTR(INT,5)
C DRAW REAR
            CALL LINE (50,185,0)
            CALL LINE (200,-275,1)
            CALL LINE (-200,-275,1)
C DRAW ANTENNA
            CALL LINE (100,450,0)
            CALL LINE (100,0,1)
            CALL LINE (0,-50,0)
            CALL LINE (0,100,1)
            CALL LINE (-10,-40,0)
            CALL LINE (0,-20,1)
            CALL LINE (-10,-15,0)
            CALL LINE (0,50,1)
C CLOSE FIGURE (BEAM BACK TO ORIGIN)
            CALL LINE (-730,-410,0)
C USE MAIN FILE AND INPUT ROUTINES WITH SUBPICTURES MADE ABOVE
C INITIALIZE DISPLAY, SET SCALE=0, INTENSITY=4
            MAINFL(1)=0
            CALL DINIT(MAINFL(1))
            CALL PLOT (PRAM,SCALE+INT+LPEN,0,4,1,IN)
C DRAW BACKGROUND
            CALL SETPT(10,10)
            CALL PLOT(LYNE,1000,100,ON,EDGE(1))
            CALL PLOT(LYNE,0,800,ON,EDGE(2))
            CALL PLOT(LYNE,-1000,100,ON,EDGE(3))
            CALL PLOT(LYNE,0,-1000,ON,EDGE(4))
C LOCATE AND DRAW HOUSE
            CALL PLOT(LYNE,50,50,OFF,DISPL)
            CALL PLOT(PRAM,BLINK,ON,BL1)
            CALL PLOT(COPI,HOUSE(1),MAIN)
            CALL PLOT(PRAM,BLINK,OFF)
C WAIT HERE FOR PUSHBUTTON HIT
200         IF (PBTN(PB)) GO TO 400
            IF (LTPN(LPY,LPX,NAME,PB)) GO TO 100
            GO TO 200
C PB5 WRITES AND READ WINDOW AND HOUSE AND RESTARTS
400         IF (PB(6)) GO TO 410
C PB4 WRITES AND READS MAINFL,WINDOW AND HOUSE ALL TOGETHER.
            IF (PB(5)) GO TO 415
C PB3 WRITES & READS MAINFL & HOUSE,THEN HOUSE AND WINDOW.
            IF (PB(4)) GO TO 416
C PB2 ADDS ANOTHER CALL TO THE WINDOW, THEN WRITES & READS ALL FILES.
            IF (PB(3)) GO TO 417
```

```
C PB1 RESTARTS FROM THE BEGINNING, TO BE DONE BEFORE EACH TEST.
      IF (PB(2)) GO TO 418
      GO TO 42%
410   CALL DCLOSE(MAINFL(1))
      CALL DYSET (HOUSE(1), SAM1, WINDOW(1), SAM2)
      CALL ENTER (5,SAM1)
      I=HOUSE(1)+1
      WRITE (5) (HOUSE(J),J=1,I)
      I=WINDOW(1)+1
      WRITE (5) (WINDOW(J),J=1,I)
      CALL CLOSE (5, SAM1)
      CALL SEEK (5, SAM1)
      READ (5) I, (B(J+1), J=1,I)
      B(1)=I
      READ (5) J,(C(K+1),K=1,J)
      C(1)=J
      CALL CLOSE (5, SAM1)
      CALL DYLINK (B(1), SAM1, C(1), SAM2)
      CALL DINIT(MAINFL(1))
      CALL REPLOT (0,B(1),MAIN)
      GO TO 20%
415   CALL DCLOSE(MAINFL(1))
      CALL DYSET(MAINFL(1),SAM1,HOUSE(1),SAM2,WINDOW(1),SAM3)
      CALL ENTER(5,SAM1)
      I=MAINFL(1)+1
      WRITE(5)(MAINFL(J),J=1,I)
      I=HOUSE(1)+1
      WRITE(5)(HOUSE(J),J=1,I)
      I=WINDOW(1)+1
      WRITE (5) (WINDOW(J),J=1,I)
      CALL CLOSE(5,SAM1)
      CALL SEEK(5,SAM1)
      READ(5)I1,(A(J+1),J=1,I1)
      A(1)=I1
      READ (5) I1, (B(J+1), J=1,I1)
      B(1)=I1
      READ (5) I1, (C(J+1), J=1,I1)
      C(1)=I1
      CALL CLOSE(5,SAM1)
      CALL DYLINK(A(1),SAM1,B(1),SAM2,C(1),SAM3)
      CALL DINIT(A(1))
      GO TO 20%
416   CALL DCLOSE(MAINFL(1))
      CALL DYSET(MAINFL(1),SAM1,HOUSE(1),SAM2)
      CALL DYSET(HOUSE(1),SAM2,WINDOW(1),SAM3)
      CALL ENTER(5,SAM1)
      I=MAINFL(1)+1
      WRITE(5)(MAINFL(J),J=1,I)
      I=HOUSE(1)+1
      WRITE(5)(HOUSE(J),J=1,I)
      I=WINDOW(1)+1
      WRITE (5) (WINDOW(J),J=1,I)
      CALL CLOSE(5,SAM1)
      CALL SEEK(5,SAM1)
      READ(5)I1,(A(J+1),J=1,I1)
      A(1)=I1
      READ (5) I1, (B(J+1), J=1,I1)
```

```
            B(1)=I1
            READ (5) I1, (C(J+1), J=1,I1)
            C(1)=I1
            CALL CLOSE(5,SAM1)
            CALL DYLINK(A(1),SAM1,B(1),SAM2)
            CALL DYLINK(B(1),SAM2,C(1),SAM3)
            CALL DINIT(A(1))
            GO TO 200
417         CALL SETPT(460,245)
            CALL PLOT(0,WINDOW(1))
            PAUSE
            GO TO 415
418         CALL DCLOSE
            GO TO 200
C PB10 ON MOVES THE HOUSE TO THE RIGHT 10 RASTER UNITS UP TO THE EDGE
420         IF (PB(11)) X=MIN0(400,X+10)
C PB9 ON MOVES IT LEFT 10 UNITS
            IF (PB(10)) X=MAX0(25,X-10)
C PB7 & PB8 MOVE IT LEFT AND RIGHT TEN UNITS
            IF (PB(9)) Y=MIN0(260,Y+10)
            IF (PB(8)) Y=MAX0(0,Y-10)
C PB6 ON TURNS OFF THE BLINKING
            IF (PB(7)) GO TO 60
            I=ON
            GO TO 70
60          I=OFF
70          CALL REPLOT(LYNE,Y,X,OFF,DISPL)
            CALL REPLOT(PRAM,BLINK,I,BL1)
C GO BACK AND WAIT FOR THE NEXT BUTTON PUSH
            GO TO 200
C PROCESS LIGHT PEN HIT
100         IF (NAME .EQ. EDGE(1)) Y=MIN0(260,LPY)
            IF (NAME .EQ. EDGE(4)) X=MIN0(400,LPX)
            GO TO 70
            END
```

TRACKING PROGRAM

The following program is an example of the manner in which the FORTRAN Tracking subroutines may be employed in the operation of the 339 Graphics System.

```
C SAMPLE FORTRAN PROGRAM USING THE TRACKING SUBROUTINE.
C ENTST7
COPYRIGHT 1968, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
      INTEGER X1,X2,Y1,Y2,Y3,X3
      LOGICAL PB,LTPN,REPLOT
      DIMENSION MAINFL(2010),PB(12),NAME(5)
C SET UP START OF DISPLAY
90    X1=100.
      Y1=100.
      MAINFL(1)=0
      Y2=Y1
      X2=X1
      CALL DINIT (MAINFL(1))
C SET SCALE =0, INTENSITY=5
      CALL PLOT(2,7,0,5,1)
      CALL SETPT(Y1,X1)
      DO 80 I=1,5
      NAME(I)=0
80    CONTINUE
      IN=1
100   LT=1
C DISPLAY TRACKING PATTERN WITH GIVEN COORDINATES.
      CALL TRACK(Y2,X2,NAME(1),PB)
C CONTROL COMES HERE WHEN A PUSHBUTTON OR MANUAL INTERRUPT IS HIT.
105   IF (PB(9)) GO TO 8
      X2=IABS(X2)
      GO TO 110
C PB8 CONSTRAINS X (MAKES THE DOT MOVE VERTICALLY
8     X2=-IABS(X2)
110   IF (PB(10)) GO TO 9
      Y2=IABS(Y2)
      GO TO 111
C PB9 CONSTRAINS Y (MAKES THE DOT MOVE HORIZONTALLY)
9     Y2=-IABS(Y2)
111   IF (PB(11)) GO TO 10
      IF (PB(12)) GO TO 11
      IF (PB(1)) GO TO 1
      IF (PB(5)) GO TO 4
```

```
C PB7 RESTARTS THE DISPLAY FILE.
          IF (PB(8)) GO TO 90
          GO TO 100
C PB10 MAKES INVISIBLE LINES.
10        LT=0
C PB11 MAKES VISIBLE LINES.
11        CALL PLOT(1,IABS(Y2)-Y1,IABS(X2)-X1,LT,NAME3)
          IF (MAINFL(1) .GE. 2000) PAUSE 123
          Y3=Y1
          X3=X1
          Y1=IABS(Y2)
          X1=IABS(X2)
          GO TO 104
C PB9 CAUSES LINES TO BE DELETED. PB1 RETURNS TO TRACKING.
1         IF (.NOT. (LTPN(NY,NX,NAME2,PB))) GO TO 1
          IF (PB(2)) GO TO 100
          IF (NAME2 .NE. NAME3) GO TO 120
          CALL DELETF(NAME2)
          Y1=Y3
          X1=X3
          GO TO 1
C 2048 REMOVES THE INTENSITY BIT FROM Y AND THE ESCAPE BIT FROM X
120       I=NAME2-MAINFL(104)+1
          Y3=MAINFL(I+1)-2048
          X3=MAINFL(I+2)-2048
C CONVERT SIGN-MAGNITUDE TO 2'S COMPLEMENT
          IF (Y3 .GE. 1024) Y3=-(Y3-1024)
          IF (X3 .GE. 1024) X3=-(X3-1024)
          IF (REPLOT(1,Y3,X3,0,NAME2)) GO TO 1
          PAUSE
C PB4 INSERTS ANOTHER CALL TO TRACKING.
4         IN=IN+1
          IF (IN-5) 5,5,100
5         CALL TRACK(Y2,X2,NAME(IN),PB)
          LT=1
          GO TO 105
          END
```

READ AND DISPLAY SLIDES

The following example program demonstrates the manner in which a series of Subpicture Display Files may be selected and displayed using the 339 System as a "slide projector".

```
C FOCC 1966 DEMO - 339 SEMINAR SLIDES.
C SLIDES
C 339 SLIDE PROJECTOR
COPYRIGHT 1966, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
      INTEGER SLIDE,DUMMY
      LOGICAL *8TN,PB
      DIMENSION PB(12),SLIDE(2500),SLNM(2),MAINFL(50),DUMMY(5),
     1 SLNM1(20),DUMNM(2),LNG(15)
      EXTERNAL TRCK
      DATA SLNM1(1),SLNM1(2),SLNM1(3),SLNM1(4),SLNM1(5),SLNM1(6)
     1 /5HSLID0,4H BIN,5HSLID1,4H BIN,5HSLID2,4H BIN/,
     2 SLNM1(7),SLNM1(8),SLNM1(9),SLNM1(10),SLNM1(11),SLNM1(12)
     3 /5HSLID3,4H BIN,5HSLID4,4H BIN,5HSLID5,4H BIN/,
     4 SLNM1(13),SLNM1(14),SLNM1(15),SLNM1(16),SLNM1(17),SLNM1(18)
     5 /5HSLID6,4H BIN,5HSLID7,4H BIN,5HSLID8,4H BIN/,
     6 SLNM1(19),SLNM1(20) /5HSLID9,4H BIN/
      DATA DUMNM(1),DUMNM(2)/5HDUMMY,4H BIN/
C SET UP A POINTER TO EACH SLIDE NAME
      NSLIDE=10
      NN=1
      DO 110 M=1,NSLIDE
      SLNM(1)=SLNM1(2*M-1)
      SLNM(2)=SLNM1(2*M)
      CALL SEEK(5,SLNM)
      LNG(M)=NN
      READ (5) L
      LP=L+NN-1
      READ (5) (SLIDE(J),J=NN,LP)
      NN=LP+1
      CALL CLOSE(5,SLNM)
110   CONTINUE
C READ DUMMY SUB-PICTURE
      CALL SEEK(5,DUMNM)
      READ (5) L
      READ (5) (DUMMY(I),I=1,L)
      CALL CLOSE(5,DUMNM)
```

```
C START UP MAIN FILE AND GET SET TO PICK SLIDES FROM PUSHBUTTONS.
          MAINFL(1)=0
          CALL OINIT(MAINFL(1))
          CALL SETPT(1000,15)
          CALL PLOT(2,3,1,7)
          CALL PLOT(0,SLIDE(1),NAME)
          CALL PLOT(0,DUMMY(1),NAME2)
          NAME3=0
10        IF (.NOT.(PBTN(PB))) GO TO 10
15        K=0
          DO 20 I=6,12
          L=12-I
          IF (PB(I)) K=K+2**L
20        CONTINUE
          IF (K .EQ. 0) GO TO 10
30        IF (K .GT. NSLIDE) GO TO 10
          M=LNG(K)
          IF (K .EQ. 7) GO TO 600
          IF (K .EQ. 10) GO TO 900
          CALL REPLOT(0,SLIDE(M),NAME)
          GO TO 10
C SHOW SLIDE AND SLIDE TOGETHER WHEN SLIDE IS REQUESTED
600       N=LNG(5)
          CALL REPLOT(0,SLIDE(M),NAME)
          CALL REPLOT(0,SLIDE(N),NAME2)
610       IF (.NOT.(PBTN(PB))) GO TO 610
          CALL DELETE(NAME2)
          GO TO 15
C SHOW TRACKING PATTERN WITH SLIDE
900       CALL REPLOT(0,SLIDE(M),NAME)
          IF (NAME3) 910,920,910
910       CALL UNBLNK(TRCK)
920       CALL TRACK(300,500,NAME3,PB)
          CALL BLANK(TRCK)
          GO TO 15
          END
```