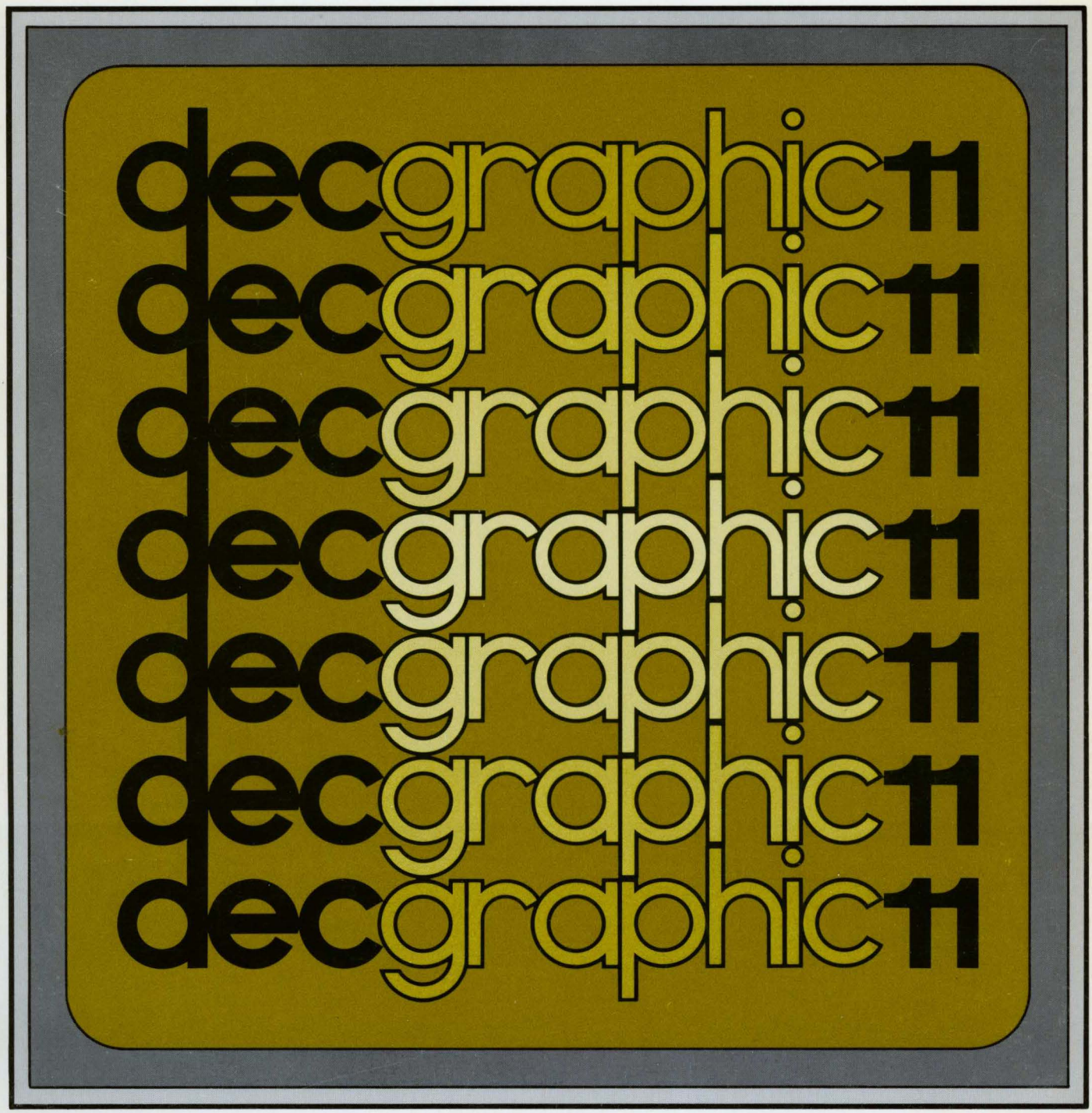
# PICTURE BOOK
# REFERENCE MANUAL




digital

# PICTURE BOOK
# REFERENCE MANUAL

For additional copies, order No. DEC-11-GPBMA-B-D
from Software Distribution Center, Digital Equipment
Corporation, Maynard, Mass.

digital equipment corporation • maynard. massachusetts

First Printing July 1973
Second Printing December 1973

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

The software described in this document is furnished to the purchaser under a license for use on a single computer system and can be copied (with inclusion of DIGITAL's copyright notice) only for use in such system, except as may otherwise be provided in writing by DIGITAL.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.



The HOW TO OBTAIN SOFTWARE INFORMATION page, located at the back of this document, explains the various services available to DIGITAL software users.

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

| | | | |
|---|---|---|---|
| CDP | DIGITAL | INDAC | PS/8 |
| COMPUTER LAB | DNC | KA10 | QUICKPOINT |
| COMSYST | EDGRIN | LAB-8 | RAD-8 |
| COMTEX | EDUSYSTEM | LAB-8/e | RSTS |
| DDT | FLIP CHIP | LAB-K | RSX |
| DEC | FOCAL | OMNIBUS | RTM |
| DECCOMM | GLC-8 | OS/8 | RT-11 |
| DECTAPE | IDAC | PDP | SABR |
| DIBOL | IDACS | PHA | TYPESET 8 |
| | | | UNIBUS |

TABLE OF CONTENTS

# PREFACE

Picture Book is a picture-producing controller for the GT40 terminal. This manual provides the programmer with the information needed to write an application program that uses Picture Book subroutines to draw graphic illustrations. The operator may then manipulate the simplified graphics functions; for example, a program that uses Picture Book subroutines could allow an architect to display the layout of a room on the GT40 screen and then move the room or parts of the room across the screen with the light pen. An electronic circuit designer could recall a specific circuit board and then alter the design.

To produce pictures using the controller, the user must write an application program that uses Picture Book or other language subroutines. The subroutines draw a picture on the GT40 screen and store the picture data in the GT40 memory. The application program, with operator interaction, calls any datum stored in the GT40 memory to the screen using the FORTRAN subroutines, and alters a picture that is already drawn.

The Picture Book package consists of FORTRAN subroutines that simulate alphanumeric communications similar to the VT06 terminal and perform Graphics functions on the GT40 terminal.

This manual assumes the reader is familiar with the following manuals:

GT40 Users Manual
DEC-11-GGTGA-A-D

GT40 Users Guide
DEC-11-HGTGA-A-D

For additional information about the GT40 terminal, refer to the following manual:

PDP-11/05 Processor Handbook

PDP-11 Peripherals and Interfacing Handbook

GT40 Graphics Display Terminal Maintenance Manual
DEC-11-HGTVA-A-D

## 1.0 INTRODUCTION

Picture Book, the software controller for the GT40 terminal, enables the user to interact with a host computer through VT06 simulation and produce a graphics display on the GT40 screen. The user can write an application program that uses this controller to draw and alter a design or picture on the GT40. The Picture Book controller operates in a GT40 terminal and requires a host computer such as the PDP-10 to run the application program.

The controller makes an accurate note of each mark the user draws. When the user first loads the controller into the GT40 memory it arranges memory into a Book with a chapter (an area in memory) for each kind of mark he can enter into a drawing; then, as the application program or the user makes entries to a graphic display, the display appears simultaneously in the GT40 Book.

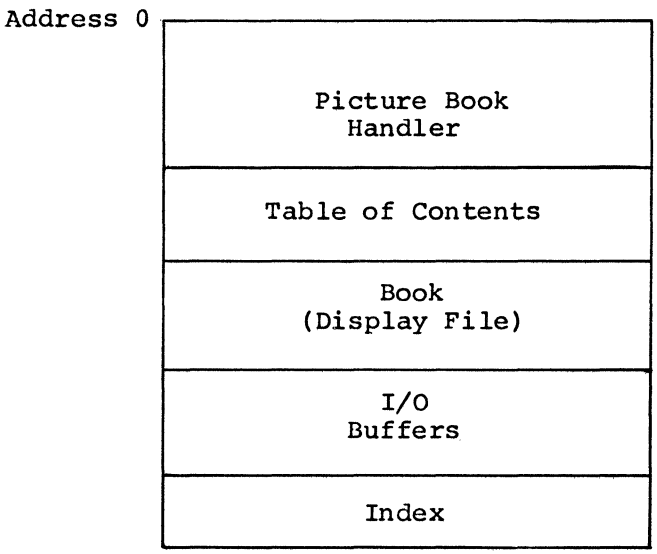
Picture Book has broad capabilities for graphic illustration; the user may be, for example, a clothing designer, an electronic circuit designer, an architect or a surveyor. In addition, Picture Book can be used to build Graphs and Tables of text that may appear in conjunction with a drawing.

Picture Book provides the user with two methods of producing a picture on the GT40; (1) creation of an application program that uses the FORTRAN callable subroutines to draw a picture which then can be changed by making keyboard entries, or (2) generation of a picture directly through keyboard entries, using the PICLET language to create or alter the display entries individually.

A user application program draws a picture on the GT40 screen by calling FORTRAN subroutines. Each FORTRAN subroutine enters a particular kind of mark in the book, and then returns control to the user program. The FORTRAN subroutines provide a variety of capabilities. The subroutines place the actual display data into the GT40 Book to access and re-display or alter anything that is drawn. The actual display can consist of 8 levels of brightness on the screen, and any part of the display may be blinking or non-blinking. The FORTRAN subroutines also provide various types of lines (dot-dash, long dash, etc.) and can make any part of the display light pen sensitive.

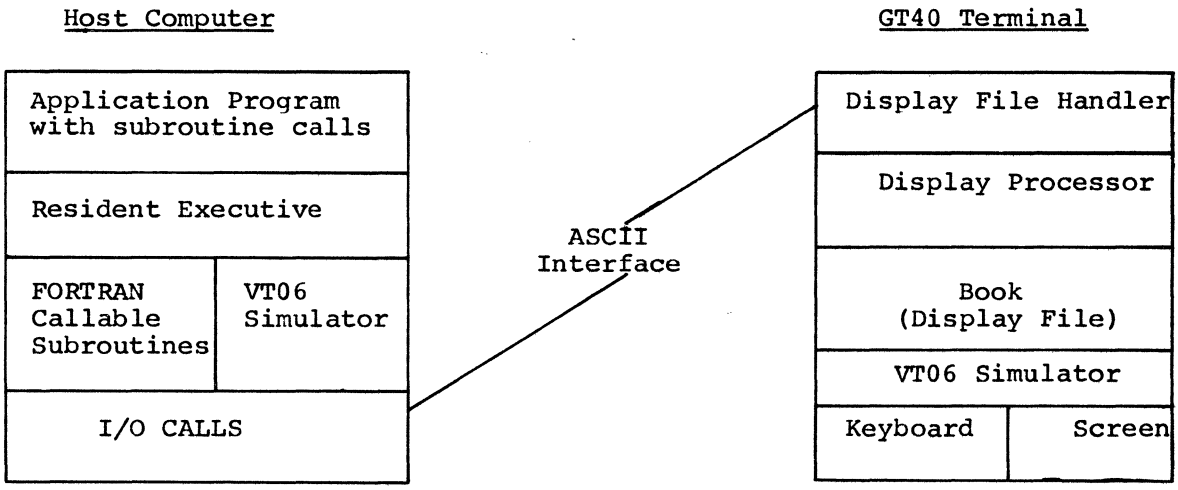
The Picture Book software package includes a teaching program called Skimming, which shows examples of FORTRAN subroutine calls and then executes the graphics that the called subroutine produces on the GT40 screen. By viewing the Skimming program the user can quickly understand how to call Picture Book subroutines with an application program, and what the subroutines draw.

The working parts of the Picture Book controller are the FORTRAN subroutines which reside in the host computer and the handler which resides in the GT40 memory. The handler consists of the Picture Book handler, the Table of Contents for the Book, the graphics Book that contains graphics data, the Index for the Book and an input/output buffer for VT06 simulation.




The Picture Book handler contains two sets of Markers; the Table of Contents and the Index (see Appendix B). The Table of Contents is a set of Markers that point to the currently open page of each chapter. The Index is a set of Markers that point to the currently open line of each page.

The following illustration is a schematic of the Picture Book processor.




When the application program is ready to draw on the GT40 screen, the program tells the Resident Executive to call the appropriate Picture Book FORTRAN subroutine (JOT, CURSOR, etc.). The Resident Executive then calls and initiates the subroutine, When the subroutine generates the appropriate code, it tells the Resident Executive to activate the appropriate I/O call (using ASCII characters) and the I/O call sends the coded data to the Display File Handler in the GT40 terminal.

The Display File Handler then places the appropriate entry into the display file, or Book. If the entry then appears in an open area of the Book, the Display File Handler automatically adds this entry to the entries already being sent from this open area of the Book to the display processor. The display file processor then sends the entry, along with the other entries in this open area of the Book, to the GT40 Screen.

When the Resident Executive sends informative or prompting messages to the GT40 screen, the Executive sends the appropriate code to the VT06 simulator. The VT06 simulator then sends the appropriate code (using ASCII characters) across the ASCII interface to the display processor in the GT40 terminal.

The display processor then sends this entry of information to the GT40 VT06 simulator. This VT06 simulator then displays the entry on the GT40 screen.

When the operator makes a keyboard entry in response to any GT40 display (VT06 simulation or a graphics display), the entry automatically enters the GT40 VT06 simulator. This VT06 simulator sends the keyboard entry to the VT06 simulator in the host computer, and that VT06 simulator relays the keyboard entry to the Resident Executive.

## 1.1 Using Picture Book

Picture Book allows the user to determine the number of lines of VT06 simulation that will appear on the screen and their length during a graphics program. This capability allows conservation of storage and keeps the terminal display from conflicting with graphics on the screen.

The Graphics Book, comprised of four types of chapters, contains the current graphics data. Each of the chapter types contains a different kind of picture data. The four types of chapters are Pictures, Figures, Graphs and Tables. Pictures contain vectors, dots(absolute points) and subpage calls. Figures contain jots, (displayable either as lines or points). Graphs contain graphs with variable axes and entries consisting of points, and Tables contain text entries that are addressable by words (character pairs).

Pictures contain long vectors and all calls to display graphics data (subpage calls). A Picture entry consists of a 3-inch line. Because of the size of a Picture line, Pictures are capable of more kinds of operations than the other three types of chapters.

Figures contain short vectors and relative or absolute points, at 1 inch per line. Both short vectors and relative points use the same data format; because of this format, the user can display a figure either as a set of points or a set of short vectors, depending upon the mode of the subroutine call. When the user specifies a short vector he may specify it to be up to 64 units in the X and Y directions on the screen. A vector longer than 64 units, in either direction, is considered a long vector and you must specify it in a Picture.

Graphs display point plotting. The user can specify any Graph to plot on either axis and he may change the graph increment.

Tables display characters which are addressable by pairs. The characters available are ASCII (English) or Special (partly Greek).

## 1.2 Execution of the User Application Program

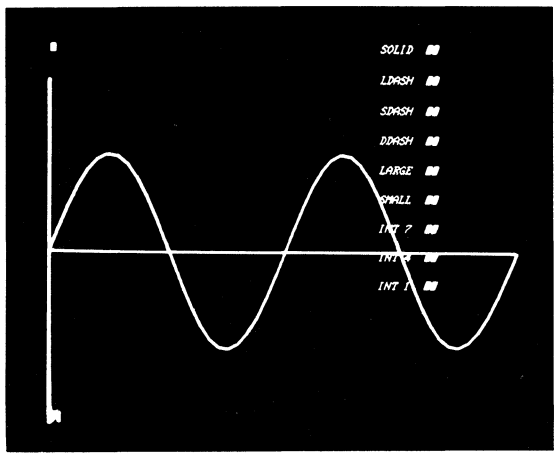
An application program for Picture Book is a central program that calls upon the Picture Book FORTRAN Subroutines to lay out the Book, determine the scale of the drawing that will appear on the screen and create the actual drawing. The Application Program may be written in any language acceptable to the host computer; however, FORTRAN is preferable, because the callable subroutines are written in FORTRAN. Some of the capabilities that this central program may provide are:

1. Making a light pen sensitive list of the parts of the drawing that the operator may change;

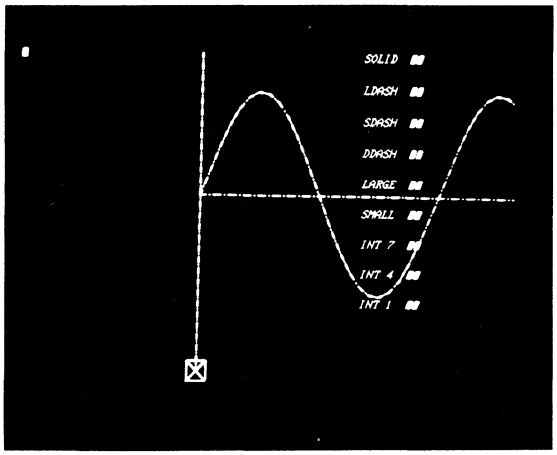
2. Requesting user entries of variable data either simultaneously with a light pen hit or independently;

3. Labeling the drawing.

When the user application program provides a list of the variable parts of the picture that appear on the screen, the user may touch the light pen to the item in the list that he wants to change.




The user application program can then request the user to enter the variable data needed to change the selected item by displaying the request on the screen.




## 2.0 PICTURE BOOK FORTRAN SUBROUTINES

Besides laying out the Book and scaling the GT40 screen, the Picture Book subroutines provide the user with the capability of opening any specific line of the Book, drawing various kinds of lines or points and calling previously drawn illustrations from the GT40 memory. These FORTRAN subroutines reside in the host computer with the application program. When the user calls Dots, Jots, Vectors or other marks, Picture Book makes these same entries in the Book, which resides in the GT40 memory. The Picture Book subroutines allow the graphics program to create a book suitable to its needs, to enter a variety of data in the book, and to perform quite complex manipulations of this data. However, the user who wishes to approach graphics on an elementary level may dispense with all but a few of Picture Book's subroutines. His program need not lay out the book since the book has a default layout. When the user calls the Picture Book Handler it automatically sets up the Book to contain 32 lines of scrolling, 1 Figure 1 line long, 1 Graph 1 line long, 1 Table 1 line long and enough pages of Pictures 100 lines long to fill the rest of core. The LAYOUT subroutine changes this original Book to match the user's specifications. This program need not manipulate the markers since they automatically increment when data enters the book. The program must scale the Book because the user's Picture will not appear on the screen until he defines the scale of the Picture. The next sections describe all the FORTRAN subroutines available.

### 2.1 Setting Up the Book

```
LAYOUT(V,CHARS,PICS,PLINES,FIGS,FLINES,GRAPHS,GLINES,TABLES,TLINES,G)
```

Layout, a FORTRAN function, sets up the user's Book according to the specifications in the argument list with each argument taken modulo 128. Picture Book employs a display file that consists of a header called the roll which calls one line after another of VT06 display, followed by the graphics file which consists of pages of Pictures, Graphs, Figures, and Tables, in turn followed by the text buffers for the VT06 display. The argument V in LAYOUT specifies how many of these buffers the display file will contain (i.e., how many lines of VT06 output the user will see before scrolling takes place). The argument CHARS determines the length in characters of these line buffers. Picture Book will display only this number of characters on

each line. The other arguments in LAYOUT specify the number of pages and their size. Picture Book converts all these arguments to modulo 128 values so that the Book created may contain a maximum of 127 pages of any type and pages a maximum of 127 lines long. Picture Book always creates at least one page of each type and makes all pages at least one line long, even for 0 arguments in LAYOUT. If the argument G is a 0, Picture Book produces a book whose tables display the standard 96 ASCII characters. If the argument is a 1, Picture Book produces a book whose tables display a set of special characters, including Greek characters and some special symbols. Picture Book does not allow mixing of the two sets.

The layout routine creates the Book from a pool of inches. Each page or buffer requires a number of these inches depending on its size. The number of inches available to the user equals the size of core minus 1250, approximately. The Layout function returns a value, modulo 4096, equal to the number of inches left over or, on overuse, puts in core a negative value indicating overuse. When overuse occurs Picture Book will reset the Book to its original layout. The following formula calculates the amount of core a layout will require:

```
INCHES=[V * CHARS + 3 * PICS * PLINES] + FIGS * FLINES + GRAPHS *
       GLINES + TABLES * TLINES + 2 * (PICS + FIGS + GRAPHS + TABLES) +
       3 * (V + PICS + FIGS + GRAPHS + TABLES)
```

The last two elements indicate overhead per page: 2 words for each index entry, 3 words for the margin between each page and text line. A program may overwrite a margin, thus using it for graphics data. The user calls LAYOUT only once, at the start of the program, to set up the graphics system. Subsequent calls to LAYOUT will re-layout the system, clearing all old information.

### 2.1.1 The Page Types

PICTURES

Pictures display long vectors and absolute points both visible and invisible. They also contain all subpage calls. Each picture line has an associated graphics mode. Picture lines require 3 inches each.

FIGURES

Figures display short vectors and relative points. Both of these have the same data format. Therefore, the user can display a figure either as a set of points or as a set of vectors, depending upon the mode of the subpage call. Figure lines require 1 inch each.

GRAPHS

Graphs display point plotting. The axis of plot depends on the type of call, the graph increment depends on the current graph increment in effect at the time of call. Graph lines require 1 inch each.

TABLES

Tables display characters at 2 characters per line. Table lines require 1 inch each.

Example:

To lay out the Book the user enters the following:

```
CALL LAYOUT (10,72,20,30,2,20,1,100,5,10,0)
```

This creates a Book consisting of:

10 lines of scrolling, 72 characters long
20 Pictures, 30 lines long
2 Figures, 20 lines long
1 Graph, 100 lines long
5 Tables, 10 lines long, with ASCII characters

SCALE(X1,X2,Y1,Y2)

This sets the scaling of the screen, setting its lower left coordinates to (X1,Y1) and its upper right coordinates to (X2,Y2). Changing the scaling does not affect graphics already produced. It only determines the size of future graphics. Original scaling is (0.,0.,0.,0.) so the user must call SCALE before doing scaled graphics.

Example:

```
CALL SCALE (-100.,-100.,100.,100.)
```

Scale the screen, using -100.,-100. as the coordinates for the lower left hand corner of the screen and 100.,100. as the coordinates for the upper right hand corner of the screen. This means that all graphics that enter the book from this time on will do so relative to these coordinates. A call to scale does not affect graphics already in the book.

THEEND

This subroutine requests that Picture Book reset the entire system to its original state, which consists of 32 lines of VT06 simulation, 72 characters long; figures, graphs, and tables, 1 each, 1 line long, and the remainder of the available inches filled with pictures 100 lines long (about 20 in an 8K system). The applications program typically requests this subroutine just before it exits. The user may also reset the Book directly from the keyboard by typing CTRL/R (depress the R key while also depressing the CTRL key).

Example:

```
CALL THEEND
```

Reset the Book to its original layout, erasing everything currently in the Book as well as all I/O displayed on the screen.

## 2.2 Drawing the Picture

The Picture Book subroutines that determine where data will enter a page follow:

OPENP(PAGE),OPENF(PAGE),OPENG(PAGE),OPENT(PAGE)
MARKP(LINE),MARKF(INCH),MARKG(INCH),MARKT(INCH)

A table of pointers to currently open pages parallels the Table of Contents in the Picture Book code. It keeps track of the currently open page in each chapter. There are always 4 open pages, 1 for each chapter. A table of book markers parallels the index. For each open page, this table contains a pointer to where the next piece of data for this page will go. Each page has its own marker. The markers originally point to the start of each page, line 0. But every time a piece of data enters a page, Picture Book updates the page's marker to point to the next free line. The user may set the open page and the markers using the above subroutines. With OPENP(5), for instance, Picture Book enters a pointer to Picture 5 in the open table. When MARKP(23) is specified, Picture Book enters a pointer to line 23 of the currently open Picture in the marker table. The next datum directed toward a Picture such as a CALL VECTOR will go in Picture 5, line 23.

Example:

OPENP(4)

Make Picture 4 the currently open picture. The next entry directed toward a picture will enter Picture 4 in the line indicated by Picture 4's marker.

OPENF(2)

Make Figure 2 the currently open figure.

OPENG(100)

Make Graph 100 the currently open graph. If the Book does not contain Graph 100, ignore the request.

OPENT(129)

Since OPEN commands take arguments modulo 128, this command makes Table 1 the currently open table.

MARKP(5)

Set the marker for the currently open picture to point to line 5. The next entry directed toward this picture will enter line 5. Its marker will then point to line 6.

MARKF(0)

Set the marker of the currently open figure to line 0 of that figure.

MARKG(100)

Set the marker of the currently open Graph to line 100. If the graphs in the Book have 100 lines (i.e., their last line is line 99) set the current graph's marker to the margin between this graph and the next graph. If the graph has 99 lines, set the current graph's marker to line 0 of the next graph. If the currently open graph is the last graph in the Book and if it has fewer than 100 lines, set its marker to line 0.

MARKT(2)

Set the marker of the currently open Table to line 2 which contains the 3rd character pair.

VECTOR(X,Y),MOVE(X,Y),POINT(X,Y),SET(X,Y)

The above routines enter graphic data in pictures, taking each argument plus or minus modulo 1024 screen units. Picture Book will, if the user specifies the margin of the page as the current line, enter data in it, thus overwriting the margin. This procedure combines the two pages previously separated by the margin.

Example:

CALL VECTOR(100.,100.)

In the line pointed to by the marker of the currently open Picture, draw a vector of coordinates 100.,100. relative to the current scaling of the screen. If the marker was pointing to the margin, entering this vector will combine the current picture with the next one.

CALL MOVE(1024.,-100.)

Assuming a scaling of (0.,1023.,0.,1023.) the invisible vector in this command has an X length of 0 since Picture Book takes Vector, Mode, Dot, and Set arguments modulo the size of the screen.

CALL DOT(0.,0.)

For a scaling of (-100.,-100.,100.,100.) draw a dot at the center of the screen.

CALL SET(-100.,100.)

This will enter an invisible absolute point in the current picture. If the negative argument refers to a point off the screen relative to the current scaling, the point produced will result from a wrap around.

BITS(BLINK,INTENSITY,TYPE,SENSITIVITY)

BITS determines the mode of the next picture entry. When Picture Book receives this command it saves for later use the graphics modes indicated in the arguments. Then when it next receives a command to add data to a picture (including a subpage call), it adds to the mode word of this line the bits it has set up. The values of the arguments determine the mode as follows:

| | | |
|---|---|---|
| BLINK | 0 | no blink |
| | 1 | blink |
| INTENSITY | 0 | (dimmest) to 7(brightest) |
| TYPE | 0 | solid |
| | 1 | long-dash |
| | 2 | short-dash |
| | 3 | dot-dash |
| SENSITIVITY | 0 | non-light pen sensitive |
| | 1 | light pen sensitive |

Picture Book ignores a negative argument. Thus, the next datum to enter a Picture for that particular mode takes on its setting from the previous line.

The user can draw every line with a different mode, or call the same figure first blinking, then not blinking, or change the mode of an entire Picture by changing the mode of the first datum in the Picture. Data entered without a specified mode takes on the mode of the previous entry in the Book.

Example:

```
CALL BITS(0,-1,0,1)
CALL VECTOR(100.,100.)
```

In the currently open Picture, enter an unblinking, dot-dash, light pen sensitive vector which will take on the intensity of the previous line in the picture.

JOT(X,Y),NOJOT(X,Y)

These routines enter a word of data in a Figure, taking each argument plus or minus modulo 64 screen units. It may also overwrite the margin that separates figures, but not the margin on the last Figure in the system, i.e., the user may not combine pages of different types.

Example:

```
CALL JOT(10.,10.)
```

Enter a jot in the currently open Figure at the line indicated by its marker, its length relative to the current scaling.

```
CALL NOJOT(-65.,65.)
```

Assuming a screen scaled to screen units, since jots have coordinates modulo 64, draw a jot of (-1,1) in the current Figure.

PLOT(Z)

The PLOT subroutine enters the coordinate Z in the currently open graph. For Z negative, it will set the graph increment to Z, modulo 64. Z is an integer value, modulo 1024. It always refers to unscaled screen units.

Examples:

```
CALL PLOT(-10)
```

Set graph increment for all graphs in the book to 10 unscaled screen units.

```
CALL PLOT(150)
```

In the currently open graph in the line indicated by its marker enter a point of coordinate 150 unscaled screen units.

TEXT(N,IARRAY)

The TEXT subroutine enters the first N characters in the array into the currently open Table, two characters per line, incrementing the marker as necessary. Each array element must contain one character. The markers for Tables always point to the next free character in the Table. However, using the MARKT routine, the user may only set the marker to a line and, thus, may not point it to the even numbered characters. Text will enter ASCII characters in a Book created for ASCII characters. For a Greek book, the text routine will enter Greek characters: an "A" in the array will produce an alpha, a "B" will produce a phi, etc.

Example:

```
DIMENSION IARRAY(4)
DATA IARRAY/'A','B','C','D'/
CALL TEXT(3,IARRAY)
```

Write the characters 'ABC' starting at the marked character of the currently open Table. The marker for this table will automatically increment 1 character space as each character enters.

ERASEP,ERASEF,ERASEG,ERASET

These routines erase the currently open Picture, Graph, Figure, and Table, respectively, starting at its marker and continuing until the first unwritten margin.

Example:

```
ERASEP
```

Starting at the line indicated by the marker in the currently open Picture, erase the remainder of the Picture until the next unwritten margin.

```
CALL ERASEG
```

Starting at the line indicated by the marker in the currently open Graph, erase each line until the next unwritten margin.

```
CALL ERASEF
```

As above, erase the currently open Figure.

```
CALL ERASET
```

As above, erase the currently open Table starting at the marked character.

HIT(PAGE,LINE),UNHIT(PAGE,LINE)

HIT and UNHIT return the picture and line number of the last light pen hit into PAGE and LINE respectively. If the hit occurred on a Figure, Graph, or Table, the value returned will point to the line in the picture that requested the hit page's display. On UNHIT Picture Book will wait for a hit before returning the picture and line number.

Example:

```
CALL HIT(I,J)
```

Return the picture number and the line number of the last light pen hit in the arguments I and J, respectively.

```
CALL UNHIT(I,J)
```

Wait for the next light pen hit and then return the picture and line number hit. If the user interrupts a program while it is waiting for a light pen hit, the user must type CTRL/R to tell Picture Book to exit from the UNHIT routine.

WAIT

WAIT waits for its own execution by Picture Book. Calling many erase subroutines without waiting for completion of each one may cause Picture Book's input buffer to overflow (this rings the GT40's bell) since erasing typically takes a long time. Inserting a WAIT after an erase ensures no output will go to Picture Book until the erase completes.

Example:

```
CALL WAIT
```

Wait until Picture Book has completed the previous request.

CURSOR(I,J)

The cursor, a blinking rectangle on the GT40 screen, indicates where the next data entry will appear on the screen. I and J can have the values +1, 0, or -1.

```
I=-1  moves the cursor down a line
I=+1  moves the cursor up a line
I= 0  moves the cursor neither up nor down
J=-1  moves the cursor left a space
J=+1  moves the cursor right a space
J= 0  moves the cursor neither right nor left
```

Example:

```
CALL CURSOR(-1,1)
```

Move the cursor one line down and one character space right.

IOTA(I)

IOTA is a function which handles the GT40 clock. The arguments may be -1, 0, or 1.

```
+1  means start the GT40's clock ticking at 1/60th second.
 0  means return the value of the timer, modulo 4096.
-1  means stop the clock and clear the timer.
```

Example:

```
I=IOTA(0)
```

Set I equal to the current value of the GT40's clock, modulo 4096.

CALL IOTA(1)

Start the GT40's clock

CALL IOTA(-1)

Stop the GT40's clock

ARC(D,IS,A,T,E)

ARC draws an arc in the currently open figure. The arguments are:

D sets the diameter.
IS sets the number of jots used to approximate the arc.
A specifies the arc in radians.
T specifies the tilt from the horizontal in radians.
E specifies the Y diameter as a fraction of the X diameter.

Example:

CALL ARC(50.,10,3,14.,0.,.75)

Starting at the marked line in the currently open Figure, draw ten Jots to form an arc comprising half of an ellipse whose Y diameter is .75 of its X diameter. The X diameter of the ellipse will lie horizontally (rotated 0 radians from the horizontal).

BELL

BELL rings the GT40's bell.

Example:

CALL BELL

Sound the GT40's bell. In PDP-10 FORTRAN this subroutine will also stop the program.

LINEX(PAGE,LINE),LINEY(PAGE,LINE)

The LINEX and LINEY functions return the X and Y coordinates (in screen units), respectively, of the picture and line number specified. Both subroutines will return -4095 when the address specified contains a subpage call. These functions can also return the coordinates of the last light pen hit. Whenever a light pen hit occurs, Picture Book's light pen interrupt routine takes the coordinates of the hit and places them in Picture 0, line 0, making sure not to alter the visibility or mode of the datum there. Picture 0, line 0, usually contains a mode word for a vector or point. Thus a light pen hit makes this vector or point go directly to the location of the hit. The user can thus draw a tracking cross directly following Picture 0, line 0 and make it light pen sensitive. Then, every time a hit occurs on the cross, the cross automatically moves because the invisible vector that determines its starting location changes. The program can also insert a subpage call following the invisible vector in Picture 0, line 0, so that an entire page will move on a light pen hit. LINEX(0,0) and LINEY(0,0) can be used to pick up the coordinates of the last light pen hit for use by the program.

Example:

```
I=LINEX(0,0)
```

Set I equal to the X-coordinate of the datum in Picture 0, line 0 in screen units, not scaled units. If this line contains a subpage call rather than a vector or dot, return a value of -4095. If this line does contain a vector and if a light pen hit has occurred, the vector will reach the point of the last hit. Thus, LINEX will return, in screen units, the X-coordinate of the hit.

```
J=LINEY(10,1)
```

Set J equal to the Y-coordinate in screen units of the vector or dot in Picture 10, line 1.

VX,(I),VY(I),DX(I),DY(I)

VX and VY convert the absolute units for a vector into scaled units. DX and DY do the same for dots.

Example:

```
X=VX(LINEX(0,0))
```

Assuming Picture 0, line 0 contains a vector, set X equal to its X length in scaled units.

```
Y=VY(J)
```

Assuming J contains a value representing a length in screen units, convert this value in Y to an equal length along the scaled Y-axis.

```
X=DX(LINEX(4,5))
```

Assuming Picture 4, line 5 contains a dot, set X equal to its X coordinate in scaled units.

```
Y=DY(LINEY(4,5))
```

Set Y to the scaled Y coordinate of the dot in Picture 4, line 5.

OUTCH(I)

OUTCH simply outputs I as a 7-bit quantity. The user who wishes to send the ASCII graphics string directly without having the FORTRAN routines perform the encoding for him, may employ OUTCH to do so.

Example:

```
CALL OUTCH(4)
```

Output the value 4 to Picture Book. 4 is the equivalent of CTRL/D which is the control character which indicates graphic data follows.

## 2.3 Calling a Picture to the Screen

PICTUR(PAGE,LINE),VECFIG(PAGE,LINE),DOTFIG(PAGE,LINE),
XGRAPH(PAGE,LINE),YGRAPH(PAGE,LINE),TABLE(PAGE,LINE)

Picture Book displays automatically only Picture 0. If a program writes in Picture 0's margin, combining it with the next picture, then both will display. But to display any other page in the system, the applications program must enter a subpage call in a Picture already on display. The subroutines above request Picture Book to insert a subpage call. All subpage calls enter the current line of the currently open Picture. As the arguments imply, a Picture can request execution of another page starting anywhere on the page. The subpage will execute up until the first unwritten margin and then return to the line following the line of the call. A subpage call may request execution of any Picture, Figure, Table or Graph; however, it may only request execution of a Picture starting at any picture line further ahead in the book than itself. This eliminates the possibility of a picture calling itself infinitely. A called Picture may, however, call a second Picture, and the second may call a third. This may continue to a depth of 50 calls.

Example:

```
CALL PICTUR(5,4)
```

In the line pointed to by the marker of the currently open Picture, enter a subpage call to Picture 5 starting the execution of Picture 5 at line 4.

```
CALL XGRAPH(0,0)
```

In the current picture enter a call to Graph 0 starting at line 0 displaying this graph along the X (horizontal) axis.

```
CALL YGRAPH(0,0)
```

Enter a call to Graph 0 from line 0 displaying this graph along the Y (vertical) axis.

```
CALL VECFIG(25,4)
```

Assuming that the book contains only 20 figures, Picture Book will ignore this request.

```
CALL DOTFIG(0,0)
```

In the line marked by the marker of the currently open figure, enter a subpage call to Figure 0 starting at line 0. Display Figure 0 as dots.

```
CALL TABLE(5,15)
```

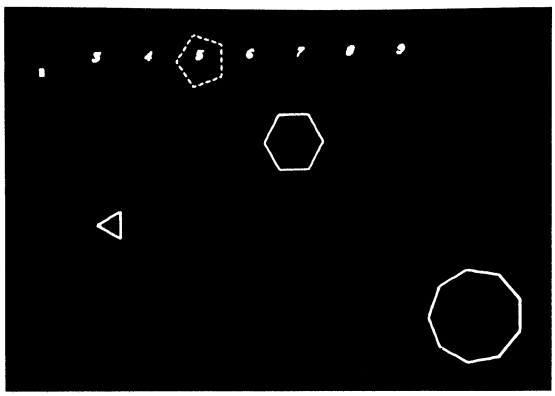
Execute Table 5 starting at line 15.

# 3.0 SAMPLE APPLICATION PROGRAMS

The following two sample application programs are examples of calling the Picture Book FORTRAN subroutines in an application program.

## 3.1 POLY

The first application program is POLY; this program first displays a list of light pen sensitive numbers across the top of the GT40 screen. When the user touches the light pen to one of the numbers, a polygon with that number of sides appears on the screen, and the user may then move the polygon across the screen. Typing carriage return will stabilize this polygon, allowing the user to pick up a new polygon.



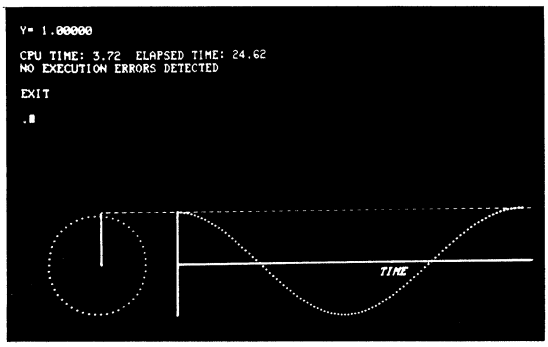
```
C                                   THIS PROGRAM ALLOWS THE USER TO
C                                   DISPLAY A POLYGON OF UP TO 9 SIDES
C                                   BY TOUCHING A DISPLAY LIST OF
C                                   NUMBERS ON THE SCREEN WITH THE
C                                   LIGHT PEN AND THEN TO MOVE THE
C                                   POLYGON TO ANY LOCATION ON THE
C                                   SCREEN AND LEAVE IT THERE
      COMMON/MARKS/MP,MF,MG,MT
C                                   MARKS CONTAINS THE COMMON MARKERS
      CALL SCALE(0.,1023.,0.,1023.)
C                                   SCALE THE SCREEN TO CORRESPOND TO
C                                   SCREEN UNITS SO THAT LINEX VALUES
C                                   NEED NOT BE UNSCALED
      CALL LAYOUT(10,72,1,127,20,20,0,0,7,1,0)
C                                   CREATE A BOOK WITH 20 FIGURES AND A
C                                   ONE LINE TABLE FOR EACH ENTRY IN
C                                   THE DISPLAY LIST
      CALL MARKP(3)
C                                   STARTING AT LINE 3 OF PICTURE 0,
C                                   ENTER CALLS TO THE SEVEN TABLES
      CALL BITS(0,7,0,1)
C                                   MAKE THESE CALLS LIGHT PEN
C                                   SENSITIVE AND BRIGHT
      DO 1 I=3,9
      CALL SET(FLOAT((I-2)*100),700.)
C                                   FIRST SET LOCATION OF THIS ENTRY ON
C                                   THE SCREEN
      CALL TABLE(I-3,0)
C                                   ENTER THE CALL
      CALL OPENT(I-3)
C                                   THEN OPEN THE CORRESPONDING TABLE
      CALL OUTCH(4)
C                                   AND INSTEAD OF USING 'CALL TEXT',
C                                   OUTPUT THE ASCII STRING THAT TEXT
C                                   WOULD OUTPUT FIRST OUTPUT CTRL/D
      CALL OUTCH(81)
```

```
C                                       THEN THE DECIMAL EQUIVALENT OF Q
        CALL OUTCH(1)
C                                       THEN INDICATE 1 CHARACTER WILL
C                                       FOLLOW
        CALL OUTCH(48+I)
C                                       NEXT COMES THE DIGIT
1       CONTINUE
        CALL BITS(0,5,0,0)
C                                       NOW TURN OFF SENSITIVITY
        CALL MOVE(0.,0.)
C                                       IN THIS DUMMY ENTRY IN THE PICTURE
        IFIG=0
C                                       SET COUNTER TO 0
2       MARK=MP
C                                       AND SAVE THE CURRENT VALUE OF THE
C                                       PICTURE MAKER
        CALL MARKP(1)
C                                       NOW CLEAR A COUPLE OF LINES BY
C                                       ENTERING INVISIBLE MOVE 0 LONG
        CALL MOVE(0.,0.)
        CALL MOVE(0.,0.)
        CALL MARKP(0)
C                                       RESET THE MARKER TO THE START OF
                                        THE PAGE
        CALL BITS(0,5,2,1)
C                                       AND SET LIGHT PEN SENSITIVITY ON
C                                       AND SHORTDASH
        CALL MOVE(0.,0.)
C                                       CLEAR OUT PICTURE 0, LINE 0, WHERE
                                        LIGHT PEN HIT COORDINATES WILL
                                        ENTER
        CALL UNHIT(I,J)
C                                       NOW WAIT FOR A HIT ON THE DISPLAY
                                        LIST
        J=J/2+1
C                                       DETERMINE THE DIGIT HIT FROM THE
C                                       LINE NUMBER THAT CAUSED THE HIT
        R=FLOAT(J)*10.
C                                       AND LET THE POLYGON'S SIZE ALSO
C                                       DEPEND ON THE NUMBER OF SIDES IT
C                                       HAS
        CALL OPENF(IFIG)
C                                       NOW OPEN THE NEXT FIGURE
        CALL ARC(R*2.,J,6.28,0.,1.)
C                                       AND DRAW AN ARC IN IT OF J SIDES,
C                                       I.E., A POLYGON
        CALL MOVE(-R,0.)
C                                       DRAW AN INVISIBLE VECTOR TO OFFSET
C                                       THE START OF THE POLYGON FROM THE
C                                       END OF THE LIGHT PEN COORDINATE
C                                       VECTOR SO THAT THE POLYGON CAN MOVE
C                                       IN ANY DIRECTION
        CALL VECFIG(IFIG,0)
C                                       AND DISPLAY THE NEW FIGURE
        READ(5,10)
C                                       NOW WAIT FOR A CARRIAGE RETURN
C                                       WHILE THE USER MOVES THE POLYGON
C                                       AROUND THE SCREEN
10      FORMAT(1X)
        I=LINEX(0,0)
C                                       WHEN IT HAPPENS, READ THE CURRENT
```

```
C                                   COORDINATES OF THE LIGHT PEN
C                                   VECTOR, I.E., THE CENTER OF THE
C                                   POLYGON
          J=LINEY(0,0)
          CALL MARKP(MARK)
C                                   MARK THE PICTURE USING THE
C                                   SAVED MARKER
          CALL SET(FLOAT(I)-R,FLOAT(J))
C                                   ENTER A SET TO THE CORNER OF THE
C                                   POLYGON
          CALL VECFIG(IFIG,0)
C                                   AND CALL THE FIGURE FROM THERE,
C                                   ALSO, THUS OVERLAYING THE SHORTDASH
C                                   POLYGON WITH A SOLID ONE
          IFIG=IFIG+1
C                                   POP THE COUNTER
          IF(IFIG.LE,19)GOTO 2
C                                   AND DO UP TO 20 FIGURES THIS WAY
          STOP
          END
```

## 3.2 CLOCK

CLOCK plots a sine wave by projecting the y coordinate of a radius of a circle onto a set of axes.




```
          INTEGER TIME (4)
C                                   TIME CONTAINS ONE ASCII CHARACTER
C                                   PER ARRAY ELEMENT
          DATA TIME/'T','I','M','E'/
          CALL SCALE(0.,100.,0.,100.)
C                                   FIRST, SCALE THE SCREEN SO THAT THE
C                                   LOWER LEFT HAND COORDINATES ARE
C                                   (0.,0.) AND THE UPPER RIGHT HAND
C                                   COORDINATES ARE (100.,100.)
          L=LAYOUT(10,72,2,20,1,100,2,127,2,10,0)
C                                   LAYOUT THE BOOK TO CONTAIN 2
C                                   PICTURES, 1 FIGURE, 2 GRAPHS, AND 1
C                                   TABLE.
C                                   NOTE THAT THE GRAPHS ARE 127 LINES
C                                   LONG, THE MAXIMUM LENGTH FOR ANY
C                                   PAGE.  WRITING IN THE 128TH LINE,
C                                   AS THIS PROGRAM DOES, EFFECTIVELY
C                                   COMBINES THE PAGES INTO ONE LONGER
C                                   PAGE
```

```
      IF((L.LT.0)STOP
C                                  IF NOT ENOUGH CORE, STOP
      CALL MOVE(0.,0.)
C                                  ZERO OUT LINE 0 OF PICTURE 0
      CALL SET(5.,25.)
C                                  AND PUT AN ABSOLUTE POINT IN LINE 1
      CALL DOTFIG(0,0)
C                                  CALL FIGURE 0 AS A SERIES OF DOTS
      CALL ARC(20.,50,6.28,0.,1.)
C                                  AND IN FIGURE 0, THE CURRENTLY OPEN
                                   FIGURE, DRAW A CIRCLE OF 50 SIDES
      CALL DOT(15.,25.)
C                                  NOW DRAW THE CENTER OF THE CIRCLE
      CALL VECTOR(0.,10.)
C                                  AND A VERTICAL RADIUS
      CALL BITS(0,2,2,0)
C                                  MAKE THE NEXT ENTRY DIM AND
                                   SHORTDASH
      CALL VECTOR(100.,0,)
C                                  DRAW A VECTOR WITH THESE MODES THAT
C                                  GOES OFF THE SCREEN HORIZONTALLY TO
C                                  THE RIGHT, IT WILL START WHEREVER
C                                  THE RADIUS ENDS.
      CALL BITS(0,5,0,0)
C                                  NOW BRIGHTEN UP THE NEXT ENTRY
      CALL SET(30.,25.)
C                                  DRAW AN INVISIBLE ABSOLUTE POINT,
C                                  IT WON'T MOVE WHEN THE RADIUS
C                                  MOVES.
      CALL PICTUR(1,0)
C                                  AND CALL PICTURE 1
      CALL SET(30.,25.)
C                                  AGAIN SET AN ABSOLUTE POINT
      CALL XGRAPH(0,0)
C                                  AND CALL THE GRAPH
      CALL SET(70.,22.)
C                                  ONCE AGAIN SET A POINT
      CALL TABLE(0,0)
C                                  AND CALL THE TABLE.  NOW PICTURE 0
C                                  CONTAINS A NUMBER OF CALLS TO PAGES
C                                  NOT YET DRAWN INTO.
      CALL OPENP(1)
C                                  OPEN PAGE 1
      CALL VECTOR(75.,0.)
C                                  AND IN IT DRAW A SET OF AXES
      CALL MOVE(-75.,10.)
      CALL VECTOR(0.,-20.)
C                                  THESE WILL DISPLAY SINCE PICTURE 0
C                                  CALLS PICTURE 1
      CALL TEXT(4,TIME)
C                                  IN THE OPEN TABLE, TABLE 0, WRITE
C                                  'TIME'
C                                  IT, TOO, WILL DISPLAY
      CALL OPENP(0)
C                                  NOW REOPEN PICTURE 0
      STEP=3.1415/50.
C                                  SET AN INCREMENT 1/100 OF 360
                                   DEGREES
      R=10.
C                                  SET THE RADIUS OF THE CIRCLE TO 10
      CALL PLOT(-7)
```

```
C                                   SET THE GRAPH INCREMENT TO 7
                                    (UNSCALED)
      DO 5 I=1,10
C                                   AND MOVE THE CURSOR DOWN 10 LINES
      CALL CURSOR(-1,0)
5     CONTINUE
      DO 10 I=1,100
C                                   THIS LOOP WILL MOVE THE RADIUS
C                                   AROUND THE CIRCLE LIKE THE HAND OF
C                                   A CLOCK AND PLOT ITS Y COORDINATE
C                                   ON THE GRAPH
      THETA=STEP*FLOAT(I)
C                                   THETA CONTAINS THE ANGLE
      X=R*SIN(THETA)
C                                   X CONTAINS ITS X COORDINATE
      Y=COS(THETA)
C                                   TAKE TIME OUT TO WRITE THE COSINE
C                                   OF THETA
      WRITE(5,100)Y
100   FORMAT('+','Y=',F8,5)
      Y=Y*R
C                                   THEN CALCULATE THE Y COORDINATE OF
                                    THE RADIUS
      CALL MARKP(4)
C                                   POINT PICTURE 0'S MARKER TO LINE 4
C                                   WHERE THE RADIUS WAS DRAWN
      CALL VECTOR(X,Y)
C                                   AND REDRAW THE RADIUS
      HT=Y+25.
C                                   NOW CONVERT ITS Y COORDINATE TO AN
                                    UNSCALED QUANTITY BY ADDING THE Y
                                    COORDINATE OF THE CENTER OF THE
                                    CIRCLE
      IHT=HT*1024./100.
C                                   AND CHANGING THE SCALING
      CALL PLOT(IHT)
C                                   PLOT THE RESULTING VALUE
10    CONTINUE
      CALL BELL
C                                   RING THE BELL, WHICH ALSO STOPS THE
                                    PROGRAM.
      STOP
      END
```

## 4.0 THE PICLET LANGUAGE

PICLET is a simple, on-line graphics language which allows its user to become acquainted with Picture Book, create simple pictures, or alter pictures drawn by a FORTRAN program.

PICLET requests input by typing 'NEXT'.

The user may then enter a command letter followed by a set of arguments each on a separate line. PICLET automatically executes the request when it has received enough arguments for the function indicated by the identification letters.

The table below contains the command letters available and their arguments.

All arguments are integers and refer to a screen scaled in screen units.

## PICLET Commands

| Command | Other Input | Result |
|---|---|---|
| A(Axis) | n1 | sets marker in graph at line n1. |
| B(Bits) | n1,n2,n3,n4 | sets mode of next data [same args as subroutine BITS] |
| C(Call) | (P,D,V,X,Y, OR T) | calls picture, dot-figure, vec-figure |
| | +n1,n2 | x-graph,y-graph,or table n1, starting at line n2 of that page. |
| D(Dot) | X,Y | absolute point at (X,Y) |
| E(Erase) | (P,F,T,G) | erase picture, figure, table or graph. |
| F(Figure) | n1 | open figure n1. |
| G(Graph) | n1 | open graph n1. |
| H(Hit) | | return picture number and line number of last light pen hit. |
| I(Inch) | n1 | marks line n1 of open figure. |
| J(Jot) | X,Y | draw a jot of (X.Y). |
| K(Karacter) | n1 | mark open table at line n1. |
| L(Line) | n1 | mark open picture at line n1. |
| M(Move) | X,Y | invisible vector (x.y). |
| N(Nojot) | X,Y | invisible jot (X,Y) |
| O(Open) | n1,n2,n3,n4,n5, n6,n7,n8,n9,n10,n11 | layout jot (X,Y) (same args as subroutine LAYOUT) Returns number of inches left over. |
| P(Picture) | n1 | open picture n1 |
| Q (Quote) | n1,c1,c2,...cn1 | insert n1 characters in current table |
| R(Reset) | | reset the book |
| S(Set) | X,Y | invisible absolute point at x,y |
| T(Table) | n1 | open table n1 |
| U(Unhit) | | wait for next light pen hit, then return picture and line hit. |
| V(Vector) | X,Y | vector (x,y) |
| W(Wait) | | wait this just returns a carriage return/line feed |
| X(X-coordinate) | | n1,n2 prints x-coordinate of picture n1, line n2 |
| Y(Y-coordinate) | | n1,n2 prints y-coordinate of picture n1, line n2 |
| Z(Z-axis) | Z | plots Z in open graph or for Z negative, set graph increment. |
| ( (arc) | diameter, sides,arc, tilt,ellipse | draws an arc in current figure. Arguments are all integer. Arc and tilt in degrees, ellipse in percent. |

## 4.1 PICLET Examples

This series of commands illustrates the interaction of PICLET and the user.

| ENTRIES | EXPLANATION |
|---|---|
| NEXT | |
| O | |
| 20 | |
| 20 | |
| 10 | LAYOUT (OPEN) THE BOOK |
| 30 | |
| 1 | (CLEARS THE SCREEN AND |
| 20 | STATES THE NUMBER OF INCHES |
| 1 | REMAINING) |
| 2 | |
| 20 | |
| 20 | |
| 0 | |
| NEXT | |
| V | |
| 100 | DRAW A VECTOR |
| 100 | |
| NEXT | |
| D | |
| 200 | DRAW A DOT |
| 200 | |
| NEXT | |
| C | |
| V | |
| 0 | CALL VEC-FIGURE |
| 0 | |
| NEXT | |
| J | |
| 30 | ENTER A JOT |
| 30 | |
| NEXT | |
| J | |
| 30 | ENTER A JOT |
| 0 | |
| NEXT | |
| C | |
| X | CALL X GRAPH |
| 0 | |
| 0 | |
| NEXT | |
| Z | SET GRAPH INCREMENT |
| -10 | |
| NEXT | |
| Z | ENTER A PLOT |
| 100 | |

```
NEXT
Z                       ENTER A PLOT
200

NEXT
Z                       ENTER A PLOT
250

NEXT
(
100                     USING ARC,
10
360                     DRAW AN ELLIPSE
0
150

NEXT
R                       RESET
```

# APPENDIX A

## ASCII TELECOMMUNICATION CODE

A table describing the ASCII code that the FORTRAN subroutines send to the GT40 follows. The user may send this code a byte at a time using CALL OUTCH(N), where N contains a 7-bit value. A CTRL/D (value=4), must precede each graphics command. An Identification letter as shown in the following chart follows the CTRL/D, then the arguments, as many as indicated in the table.

All numeric arguments represent single values except those of VECTOR, SET, MOVE, DOT, and PLOT where each pair of arguments represents a 14-bit quantity high byte first e.g., 177 followed by 177 yields the quantity -1. BITS, JOT, NOJOT, and IOTA take 1-byte, 7-bit, arguments ranging from the negative value 100 to the positive value 77. The single byte arguments of LAYOUT, OPENn, MARKn, LINEn, and the subpage calls represent positive values. In other words, 177 represents the decimal value 127.

Some routines return values, in which case FORTRAN or MACRO calling routines must wait for return of a 4-digit octal value encoded in ASCII (i.e., 65 = '5'), sometimes preceded by a minus sign (-55), and followed by a carriage return.

FORTRAN routines may input using a FORTRAN READ with octal format specifications or may use CALL INNUM(I) to return the value in I or may CALL INCH(I) to return each byte separately.

For example, to LAYOUT the BOOK send this string: (octal values)

4, 117, 6, 10, 40, 20, 5, 5, 1, 100, 3, 10, 0

then input a 4-digit octal number.

This does what CALL LAYOUT(6, 64, 8, 16, 5, 5, 1, 64, 3, 8, 0) does.

NOTE

> The second argument in the ASCII string for LAYOUT specifies character pairs, not characters as in the subroutine call.

## PICTURE BOOK CODE

In the following table, N in the arguments column equals 7 bits.

| Name | Function | Code | Arguments | Return |
|---|---|---|---|---|
| LAYOUT<br>P F G T<br>(V,C,P,L,P,L,P,L,P,L,G) | opens the book | Open | 11*N | Yes |
| BITS<br>0-1 0-7 0-3 0-1<br>(B I T S)<br>ARG=-1 MEANS NO-ENABLE | mode | B | 4*n | --- |
| OPEN(P) | open picture | P | n | --- |
| OPENG(P) | open graph | G | n | --- |
| OPENT(P) | open table | T | n | --- |
| OPENF(P) | open figure | R | n | --- |
| MARKP(L) | at line in picture | L | n | --- |
| MARKG(I) | at inch in graph | A | n | --- |
| MARKT(C) | set char in table | K | n | --- |
| MARKF(I) | set char in figure | I | n | --- |
| PICTUR(P,L) | call picture | CP | 2*n | --- |
| XGRAPH(P,L) | call graph | CX | 2*n | --- |
| YGRAPH(P,L) | call graph | CY | 2*n | --- |
| TABLE(P,L) | call table | CT | 2*n | --- |
| DOTFIG(P,L) | call figure | CD | 2*n | --- |
| VECFIG(P,L) | call figure | CV | 2*n | --- |
| VECTOR(X,Y) | rel. vector | V | 4*n | --- |
| DOT(X,Y) | abs. point | D | 4*n | --- |
| MOVE(X,Y) | inv. vector | M | 4*n | --- |
| SET(X,Y) | inv. point | S | 4*n | --- |
| TEXT(M,A) | text, N chars from array A | QM | m*n | --- |
| ERASEP | erase picture | EP | 0*n | --- |
| ERASEG | erase graph | EG | 0*n | --- |
| ERASET | erase table | ET | 0*n | --- |
| ERASEF | erase figure | EF | 0*n | --- |
| HIT(P,L) | L.P. HIT | H | 0*N | Yes |
| LINEX(P,L) | read x-coordinates | X | 0*n | Yes |
| LINEY(P,L) | read y-coordinates | Y | 0*n | Yes |
| PLOT(Z) | plots | Z | 2*n | --- |
| THEEND | resets | R | 0*n | --- |
| JOT(X,Y) | writes in figure | | 2*n | --- |
| NOJOT(X,Y) | writes in figure | | 2*n | --- |
| UNHIT(P,L) | does not return till LP hit | U | 0*n | Yes |
| WAIT | waits for PB to execute it | W | 0*n | Yes |
| IOTA(I) | services the clock | I | n | Yes |

Examples:

```
CALL OUTCH(4)
CALL OUTCH(67)
CALL OUTCH(68)
CALL OUTCH(0)
CALL OUTCH(5)
```

This set of calls to OUTCH outputs the following octal string:

```
4, 103, 104, 0, 5
```

which is CTRL/D followed by the ID letters C and D for CALL DOT-FIGURE and the numerics 0 and 5 for page 0, line 5.

The string

```
4, 86, 1, 2, 177, 171
```

draws a vector of coordinates 130, -7. It outputs a CTRL/D followed by 86, the ASCII equivalent of V. It then outputs the high byte of the X-coordinate, 1, followed by the low byte, 2, which form the value 128+2 or 130. The next two bytes contain the Y coordinate as a 14-bit value, its highest bit set, indicating negativity.

## APPENDIX B

## COMMON MARKERS

A FORTRAN common area labelled MARKS keeps track of the value of the markers so that the user can save the value of the marker in an array or variable just before writing into the line it points to. This allows him to later reset the marker to this saved value so that he can re-write the line previously modified. Common area MARKS consists of 4 integer variables:

PICTURE-MARKER
FIGURE-MARKER
GRAPH-MARKER
TABLE-MARKER

i.e., COMMON/MARKS/MP,MF,MG,MT Will contain the figure marker in MF. Each common marker contains the marker for the currently open page of its type, but only if set correctly when the page is opened. Each addition to a current page increments its common marker by 1. Each call to a mark subroutine such as MARKP sets the marker to the value of the subroutine's argument. In the case of the table marker, however, the common marker points to the current character, not the current line. Since each line contains two characters, the user would have to divide the character marker by 2 to obtain the line currently marked.

The user may set the common markers by writing into them when opening a new page, e.g., MP=5, or by marking a newly opened page using a MARKn subroutine. But if the user does not, when opening a new page, set the common marker of its type to the current marker for the new page, the common marker continues to contain the marker for the previously current page.

Example:

```
      COMMON/MARKS/MP,MF,MG,MT
      .
      .
      .
      M=MF
      DO 120 I=3,10
      CALL MARKF(M)
      CALL ARC(100.,I,6.28,0.,1.)
120   CONTINUE
      .
      .
```

This draws 7 closed polygons in the currently open figure; each polygon having one more side than the previous one. For each pass through the loop, MARKF(M) resets the marker to the line where the first polygon started so that each polygon overwrites the previous polygon. M was set to the current value of the figure marker with M=MF.

## APPENDIX C

## THE PICTURE BOOK LIBRARY (GLIB)

The following list includes all the Picture Book subroutines. These subroutines collectively appear in the Picture Book Library (GLIB).

SCALE
VX
VY
DX
DY
CURSOR
ARC
LAYOUT
OPENP
OPENG
OPENG
OPENT
MARKP
MARKF
MARKG
MARKT
ERASEP
ERASEF
ERASEG
ERASET
BITS
PICTUR
VECFIG
DOTFIG
XGRAPH
YGRAPH
TABLE

VECTOR
MOVE
DOT
SET
JOT
NOJOT
PLOT
TEXT
LINEX
LINEY
HIT
UNHIT
IOTA
WAIT
THEEND
BELL
INCH
OUTCH

## APPENDIX D

## CORE REQUIREMENTS

Because of its design, Picture Book requires a minimum of core to implement. For instance, rather than requiring a separate file of subpictures in core, Picture Book allows the user to set up his own display file in Picture 0 of whatever dimensions he desires. And rather than having a separate header for each subpicture called which would specify the mode of that subpicture, Picture Book allows the user to set the mode himself in the subpicture call. In addition, Picture Book does not require a complicated light pen routine or one that must perform several operations such as displaying a tracking cross, connecting a subpicture, or returning coordinates of the last light pen hit. These and other Picture Book features limit the code to 1-1/4K, leaving 2-3/4K of display file in a 4K system.

## APPENDIX E

## THE GT40 CHARACTER SET

The following table lists all the characters that the GT40 handles.

| 7-bit code (octal) | ASCII Representation | Keyboard | GT40 Printing | When preceded by Shift-Out = 016 |
|---|---|---|---|---|
| 000 | NUL | CTRL @ | | λ |
| 001 | SOH | CTRL A | | α |
| 002 | STX | CTRL B | | ϕ |
| 003 | ETX | CTRL C | | Σ |
| 004 | EOT | CTRL D | | δ |
| 005 | ENQ | CTRL E | | Δ |
| 006 | ACK | CTRL F | | ι |
| 007 | BEL | CTRL G | | |
| 010 | BS | CTRL H | Backspace | Π |
| 011 | HT | CTRL I (TAB) | | ψ |
| 012 | LF | CTRL J (LF) | Line Feed | ÷ |
| 013 | VT | CTRL K | | o |
| 014 | FF | CTRL L | | ∴ |
| 015 | CR | CTRL M (CR) | Carriage Return | μ |
| 016 | SO | CTRL N | | £ |
| 017 | SI | CTRL O | | Shift In |
| 020 | DLE | CTRL P | | π |
| 021 | DC1 | CTRL Q | | ‖ |
| 022 | DC2 | CTRL R | | Ω |
| 023 | DC3 | CRTL S | | σ |
| 024 | DC4 | CRTL T | | τ |
| 025 | NAK | CTRL U | | ε |
| 026 | SYN | CTRL V | | ← |
| 027 | ETB | CTRL W | | → |
| 030 | CAN | CTRL X | | ↑ |
| 031 | EM | CTRL Y | | ↓ |
| 032 | SUM | CTRL Z | | ⌜ |
| 033 | ESC | CTRL [ (ALT) | | ⌟ |
| 034 | FS | CTRL \ | | ≠ |
| 035 | GS | CTRL ] | | ~ |
| 036 | RS | CTRL ↑ | | ∨ |
| 037 | US | CTRL ← | | □ |
| 40 | SP | SPACE BAR | Space 1 character | |
| 41 | ! | SHIFT 1 | ! | |
| 42 | " | SHIFT 2 | " | |
| 43 | # | SHIFT 3 | # | |
| 44 | $ | SHIFT 4 | $ | |
| 45 | % | SHIFT 5 | % | |
| 46 | & | SHIFT 6 | & | |
| 47 | ' | SHIFT 7 | ' | |
| 50 | ( | SHIFT 8 | ( | |
| 51 | ) | SHIFT 9 | ) | |
| 52 | * | SHIFT : | * | |
| 53 | + | SHIFT ; | + | |
| 54 | , | , | , | |
| 55 | -(MINUS) | - | - | |
| 56 | . | . | . | |
| 57 | / | / | / | |
| 60 | 0 | 0 | 0 | |
| 61 | 1 | 1 | 1 | |
| 62 | 2 | 2 | 2 | |

| | | | |
|---|---|---|---|
| 63 | 3 | 3 | 3 |
| 64 | 4 | 4 | 4 |
| 65 | 5 | 5 | 5 |
| 66 | 6 | 6 | 6 |
| 67 | 7 | 7 | 7 |
| 70 | 8 | 8 | 8 |
| 71 | 9 | 9 | 9 |
| 72 | : | : | : |
| 73 | ; | ; | ; |
| 74 | < | SHIFT , | < |
| 75 | = | SHIFT - | = |
| 76 | > | SHIFT . | > |
| 77 | ? | SHIFT / | ? |
| 100 | @ | @ | @ |
| 101 | A | SHIFT A | A |
| 102 | B | SHIFT B | B |
| 103 | C | SHIFT C | C |
| 104 | D | SHIFT D | D |
| 105 | E | SHIFT E | E |
| 106 | F | SHIFT F | F |
| 107 | G | SHIFT G | G |
| 110 | H | SHIFT H | H |
| 111 | I | SHIFT I | I |
| 112 | J | SHIFT J | J |
| 113 | K | SHIFT K | K |
| 114 | L | SHIFT L | L |
| 115 | M | SHIFT M | M |
| 116 | N | SHIFT N | N |
| 117 | O | SHIFT O | O |
| 120 | P | SHIFT P | P |
| 121 | Q | SHIFT Q | Q |
| 122 | R | SHIFT R | R |
| 123 | S | SHIFT S | S |
| 124 | T | SHIFT T | T |
| 125 | U | SHIFT U | U |
| 126 | V | SHIFT V | V |
| 127 | W | SHIFT W | W |
| 130 | X | SHIFT X | X |
| 131 | Y | SHIFT Y | Y |
| 132 | Z | SHIFT Z | Z |
| 133 | [ | [ | [ |
| 134 | \ | \ | \ |
| 135 | ] | ] | ] |
| 136 | ^ | ^ | ^ |
| 137 | - | - | - |
| 140 | \ | SHIFT @ | \ |
| 141 | a | A | a |
| 142 | b | B | b |
| 143 | c | C | c |
| 144 | d | D | d |
| 145 | e | E | e |
| 146 | f | F | f |
| 147 | g | G | g |
| 150 | h | H | h |
| 151 | i | I | i |
| 152 | j | J | j |
| 153 | k | K | k |
| 154 | l | L | l |
| 155 | m | M | m |
| 156 | n | N | n |
| 157 | o | O | o |

| | | | |
|---|---|---|---|
| 160 | p | P | p |
| 161 | q | Q | q |
| 162 | r | R | r |
| 163 | s | S | s |
| 164 | t | T | t |
| 165 | u | U | u |
| 166 | v | V | v |
| 167 | w | W | w |
| 170 | x | X | x |
| 171 | y | Y | y |
| 172 | z | Z | z |
| 173 | [ | SHIFT [ | |
| 174 | | SHIFT \ | |
| 175 | | SHIFT ] | |
| 176 | | SHIFT | |
| 177 | RUB OUT | R.O. | ■ |

Function Key Codes

| | | | | | |
|---|---|---|---|---|---|
| ←10 | ↑ 32 | HOME | 35 | EOS | 37 |
| →30 | ↓ 33 | EOL | 36 | | |

To run an application program, the user should assign unit 5 to the GT40 because the FORTRAN subroutines use this unit number for input and output. The Picture Book FORTRAN subroutines use a common area labelled BOOK; therefore the User application program should not use this label for a new common area.

## APPENDIX F

## OPERATING INSTRUCTIONS FOR THE PDP-10

The following series of steps show how to load the Picture Book Controller for the PDP-10. This sequence shows how to run either an application program or SKIM, a teaching program, after loading Picture Book. SKIM illustrates the correct usage of Picture Book subroutine calls and executes them.

1. Load address 166000 and start.
   This starts the bootstrap Teletype simulator. Press START to reset the screen.

2. LOGIN

3. Transfer all files from DECtape to disk using PIP or FILE.

4. Type: RU LOADVT
   To start the loader program, which clears the screen and prompts with an asterisk.

5. Type: ←BOOK and wait for the **RESET** message, which indicates the load has completed.

6. To run SKIM, the teaching program, type RU SKIM.

7. To run PICLET, type RU PICLET.

8. To execute a FORTRAN program that uses the FORTRAN graphics subroutine package. Use GLIB.REL as the standard library file.

   e.g. EX NAME.F4, GLIB/LIB

Note: The graphic data generated by the graphics subroutines travels over the same communications line as ordinary Teletype data including characters echoed from the keyboard. Typing on the keyboard during execution of a program that does graphics can cause Picture Book to receive erroneous data. Therefore, the user should refrain from typing during the execution of a program that does graphics except in response to Teletype input requests, i.e. READs or ACCEPTs.

## APPENDIX G

## PICTURE BOOK ASSEMBLY INSTRUCTIONS

These instructions are provided for assembling PICTURE BOOK sources on a DECsystem 10. They assume the user has transferred all source files to a disk area on the DECsystem 10. All command lines are underlined and are terminated with a carriage return. For further information on DECsystem 10 monitor commands, consult the DECsystem 10 User's Handbook, DEC-10-NGZB-D.

### BOOK.BIN

To assemble the display file handler for the GT40, BOOK.BIN, use MACDLX, the PDP-11 assembler, with the current version of BOOK.

```
.R MACDLX

*BOOK.BIN/I←BOOK.002
```

### GLIB.REL

To assemble the relocatable FORTRAN library used with FORTRAN application programs, several steps are necessary. You must first assemble three source files, BOOK.F4, INCH.MAC, and OUTCH.MAC and then combine them into a single library file:

```
.COMPILE INCH.MAC
MACRO:  INCH

EXIT

.COMPILE OUTCH.MAC
MACRO:  OUTCH

EXIT

.COMPILE BOOK.F4
FORTRAN:  BOOK.F4

EXIT


.R PIP

*DSK:GLIB.REL=INCH.REL,OUTCH.REL,BOOK.REL
```

### PICLET.SAV

To create a save image of PICLET, you must first compile and link it with GLIB.REL and then save the file:

```
.LOAD PICLET.F4,GLIB/L
FORTRAN:  PICLET.F4
LINK:  LOADING

EXIT

.SAVE
JOB SAVED
```

SKIM.SAV

To create a save image of SKIM you must compile and link it with GLIB.REL and then save it:

```
.LOAD SKIM.F4,GLIB/L
FORTRAN:  SKIM.F4
LINK:  LOADING

EXIT

.SAVE
JOB SAVED
```

# APPENDIX H

# PICTUREBOOK IN THE DOS/BATCH SYSTEM

## H.1.0 INTRODUCTION

This is an addendum to the Picture Book manual; the reader should be familiar with that manual before reading this addendum.

This package of software for the GT40 provides the means of interfacing the GT40 graphics terminal to the DOS/BATCH System. The loader provided with the package loads programs into the GT40 through a DL11 serial asynchronous interface, using the ROM bootstrap in the GT40. It may be used to load any program in paper tape binary format such as Picture Book, the graphics software supplied with this package.

The PICTURE BOOK software supplied for DOS/BATCH is an adaptation of the software originally developed for DECsystem-10. It retains all of the graphics capabilities of the original software described in the PICTURE BOOK REFERENCE MANUAL (DEC-11-GPBMA-A-D), but it differs in one important aspect: the GT40 is not supported as a terminal or console device under DOS/BATCH.

The PICTURE BOOK graphics package consists of a display file handler resident in the GT40 and a library of FORTRAN subroutines. Displays are produced by a series of calls to the subroutines, which create the display file in the GT40 by transmitting data coded in ASCII format to the display file handler in the GT40. The display file handler decodes the characters and makes the appropriate entry of graphics data into the display file.

Light pen support is provided locally by the display file handler, rendering fast response. Light pen tracking is easy to implement and light pen data may be requested at any time by a FORTRAN call. Keyboard support is provided by INLIN, a FORTRAN call to read a line from the keyboard, along with the standard DECODE and FORMAT statements, as explained in the FORTRAN IV Manual (DEC-11-LFIVA-A-D).

## H.2.0 LOADGT, THE DOS/BATCH LOADER

### H.2.1 Description

LOADGT is a loader program which functions under DOS/BATCH and replaces LOADVT, the DECsystem-10 loader supplied with the original PICTURE BOOK package. The program will load a GT40 with any binary program in paper tape format from any DOS/BATCH standard device capable of binary input (paper tape reader, disk, DECtape).

LOADGT uses the DL11 serial asynchronous interface between the PDP-11 and GT40 to transmit the requested program to the GT40. Since the ROM terminal bootstrap supplied with the GT40 is used for the loading function, the data being transferred will be coded by the loader into 6-bit printing ASCII characters, as described in the GT40 USER'S GUIDE (DEC-11-HGTGA-A-D).

The loader program does not check parity or checksums when transmitting the program. It relies on the bootstrap loader to detect any errors in the program being transmitted. LOADGT monitors the DL11

receiver for transmissions from the GT40, but it responds only to three character combinations: ALT MODE B, ALT MODE G, and CTRL C.

The ALT MODE B combination is sent by the bootstrap when a checksum error is detected. It causes the loader to abort the load, print an error message, and restart. The ALT MODE G combination, sent by the bootstrap, and CTRL C, sent by the PICTURE BOOK display file handler, indicate a satisfactory load. The loader then terminates the load and restarts.

The LOADGT source is conditionalized to permit its easy adaptation to various DOS/BATCH configurations. On most single keyboard systems, the DL11 for the GT40 connection will be located at interrupt trap address 300 and device addresses 175610 to 175616. These are the default values used by the loader. For systems with other values, the source of the loader should be obtained and assembled with the correct assignments, as explained in section H.5.0 on assembly instructions.

### H.2.2 Command Syntax

The loader uses standard DOS/BATCH command string syntax (CSI format as described in the DOS/BATCH MONITOR PROGRAMMER'S HANDBOOK, DEC-11-OMPMA-A-D). However, it does not permit the use of the asterisk wild-card convention or the use of any switches in the command string. The command string format is as follows:

```
<dev:filnam.ext[uic]
```

where:

| | |
|---|---|
| dev | is a 2- or 3-character designation for the standard device on which the file to be loaded is located; |
| filnam | is a 1- to 6-character name of the file to be loaded; |
| ext | is the 3-character filename extension; |
| uic | is the user identification code, |

The loader identifies itself and then prompts command input by printing a #.

```
LOADGT V01
#
```

It then accepts a command string with a single input dataset specification in the format previously described. A file name need not be given if the device specified is not directory-structured (e.g., the paper tape reader, PR). If the device is directory-structured, a file name must be specified.

Default values are used for some components of the command string. The default for device is the system device. The default file name extension is LDA. The default uic is the project-programmer number currently logged in.

### H.2.3 Examples

Some examples of loader command strings for various situations are given below. All commands are terminated with a carriage return.

1. To load from the system device:

```
#<BOOK.BIN
```

2. To load from another UIC e.g. (100,100), on DK1:

```
#<DK1:BOOK.BIN[100,100]
```

3. To load from the paper tape reader:

```
#<PR:
```

4. To load from DECtape:

```
#<DT0:BOOK.BIN
```

### H.2.4 Error Messages

In using the loader, various conditions may produce errors. Some conditions produce error messages through the monitor error routine. These are explained in the DOS/BATCH MONITOR PROGRAMMER'S HANDBOOK. Other errors are handled by the loader itself, and produce one of the error messages explained below. The loader restarts after issuing the error message.

| Message | Significance |
|---|---|
| COMMAND STRING SYNTAX ERROR AT: x | A syntactical error was made in the command string, detected at the letter (x) printed after the colon. |
| FILE NAME ERROR | An error was made in the file name or extension, such as using an *, neglecting to specify a file for a directoried device, etc. |
| FILE NOT FOUND | On a directoried device, the specified file was not found in the directory for the specified UIC. |
| FILE READ ERROR | Either file format was incorrect (e.g., an ASCII file) or an error was persistent after many repetitions by the file handler. |
| XMIT ERROR | A checksum error was detected by the GT40 bootstrap loader. |

## H.3.0 PICTURE BOOK UNDER DOS/BATCH

### H.3.1 General Description and Limitations

PICTURE BOOK consists of a display file handler and a library of FORTRAN subroutine calls. This implementation under DOS/BATCH-11 affects only the set of subroutines - the display file handler remains unchanged. In addition, the use of the FORTRAN calls remains the same, the changes being internal to the subroutines. The descriptions of these calls in the PICTURE BOOK manual are still applicable.

The major change in implementation is that the GT40 is not supported as a terminal device. Input and output between DOS/BATCH and the GT40 must be done through the PICTURE BOOK calls. FORTRAN READ and WRITE statements can not be used; however, ENCODEs and DECODEs permit formatted transfers of data.

### H.3.2 SETUP Subroutine

A new assembly language subroutine named SETUP has been added to the PICTURE BOOK library. This subroutine sets up the interrupt structure for communicating with the GT40.

The syntax for the SETUP subroutine call is:

```
CALL SETUP (ivec,idev)
```

where:

ivec — is a decimal or octal constant giving the legal address of the DL11 receiver interrupt trap vector for this configuration; (octal constants must be preceded by a quotation mark);

idev — is a decimal or octal constant giving the legal address of the DL11 receiver status register.

Example:

```
CALL SETUP ("304,"175620)
```

This call to SETUP defines the address 304 as the vector transfer address, and the address 175620 as the device address.

A CALL SETUP must be included in every FORTRAN program using the PICTURE BOOK graphics subroutine calls. It must also precede the first call of a PICTURE BOOK subroutine, including LAYOUT. This is necessary because SETUP makes the necessary linkages to the DOS/BATCH interrupt structure.

The location of the DL11 asynchronous interface to the GT40 may "float" from configuration to configuration. The call to SETUP enables the user to specify the exact locations used in his particular configuration. The values are stored as global values to be used by OUTCH and the receiver interrupt handler located in the SETUP code.

Consequently, SETUP must always be core resident and can not be overlaid.

If no arguments are specified with CALL SETUP, the default values "300,"175610 are assumed. The default call to SETUP then appears as follows:

```
CALL SETUP (1)
```

The value 1 in the above example is the default value assigned by the system.

Since SETUP has inserted the address of its own interrupt handler into the DL11 trap address, a means is provided to restore control to DOS/BATCH through a terminating call to SETUP. An argument of zero causes SETUP to restore the previous contents of the trap address. This call to SETUP should be used at the end of every program before exiting to the monitor.

Example: `CALL SETUP (0)`

### H.3.3 The INLIN Function

A new function has been added to the PICTURE BOOK library to support the GT40 keyboard. The function accepts characters from the keyboard and packs them into a byte array until a character count is exceeded or a carriage return/line feed combination is encountered. The function then returns the actual number of characters transferred, including the carriage return and line feed. The syntax for the call is:

```
CALL INLIN (n, array)
```

or

```
I = INLIN (n, array)
```

where:

| | |
|---|---|
| n | is the number of characters to transfer. |
| array | is the name of the byte array to receive the characters. |
| I | is the actual number of characters transferred. |

An example of the use of INLIN follows the description of the OUTLIN function.

### H.3.4 The OUTLIN Function

Another function was added to the PICTURE BOOK library to facilitate transmitting strings of ASCII characters to the GT40. These character strings differ from the strings sent by the CALL TEXT subroutine in that they are not preceded by any control characters. The characters are sent to the scroll buffer which is maintained by the display file

handler for terminal communications. They are not preserved in the buffer but are scrolled off the top of the screen as new characters fill the buffer. This feature is useful for sending messages to the GT40, since it is otherwise not supported as a terminal.

Before the call is made, the character string must be placed in a byte array, using a DATA or ENCODE statement. OUTLIN interprets the first character in the array as a standard FORTRAN carriage control character and then discards the character. See the FORTRAN IV COMPILER AND OBJECT TIME SYSTEM PROGRAMMER'S MANUAL (DEC-11-LFRTA-A-D), section 7.18, for a list of carriage control characters.

The syntax of the call is:

```
CALL OUTLIN (n, array)
```

or

```
I = OUTLIN (n, array)
```

where:

| | |
|---|---|
| n | is the number of characters to transfer; |
| array | is the name of the byte array containing the characters. |
| I | is the actual number of characters transferred. |

Example:

```
          BYTE IN(72) OUT(72)
          CALL SETUP(1)
          ENCODE(8,100,OUT)
100       FORMAT ('0', 7HTYPE J=)
          CALL OUTLIN  (8, OUT)
          I = INLIN (72,IN)
          DECODE (I, 101, IN) J
101       FORMAT (F8.5)
          ENCODE (21,102,OUT)J
102       FORMAT (X, 'YOU TYPED J=', F8.5)
          CALL OUTLIN (21,OUT)
          CALL SETUP (0)
          END
```

This example demonstrates the use of INLIN and OUTLIN with ENCODE and DECODE. The program will request the operator to type a number in the format F8.5 (for example:12.34567), and will convert this entry to a floating point decimal number, then back to an ASCII format number. The program will then output this number on the GT40 terminal.

## H.4.0 OPERATING PROCEDURES

### H.4.1 Using the Loader

To use DOS/BATCH-11 to load the GT40 with a program, the user should first ensure that the GT40 is interfaced through a DL11 (other than the DL11 used for the console device) and that LOADGT is present on his disk as a linked file. The loader is supplied as an object module and must be linked on the system where it will be used. (See section H.5.1.1.) The following procedure should then be followed (underlined text is output by DOS/BATCH or the program):

1. Start the execution of LOADGT.

```
.RU LOADGT)
LOADGT V01
#
```

   LOADGT identifies itself and prompts command input with a "#".

2. Enter the command string.

```
LOADGT V01
#<BOOK.BIN)
```

3. If the file was successfully loaded, the LOADGT program will start over, repeating its identification. The user should return to the monitor and then kill LOADGT before starting another program. For example,

```
LOADGT V01
#↑C
.KI)
$
```

4. If an error occurred, two things can happen. Errors handled by LOADGT result in an error message being printed on the DOS/BATCH console printer, followed by a restart of the program. Section H.2.4 explains these Errors. Or other errors cause an exit to the monitor's error processing facility. An error printout is produced on the DOS/BATCH console printer, consisting of a letter and three digits. These codes are explained in the DOS/BATCH MONITOR PROGRAMMER'S HANDBOOK.

5. When transferring absolute files such as BOOK.BIN using PIP, the switch /UB (unformatted binary) should be used. PIP does not recognize the BIN extension, although it is the assembler default extension for absolute files. For example, to transfer BOOK.BIN to disk from DECtape, use the PIP command string:

```
#DK:<DT0:BOOK.BIN[1,1]/UB
```

### H.4.2 Using PICTURE BOOK

The user must have an application program in FORTRAN that has been properly linked with both the FORTRAN system library and the special PICTURE BOOK library, which is called DLIB.LIB to distinguish it from the DECsystem-10 version of PICTURE BOOK.

The linking procedure is as follows, assuming POLY.OBJ is the compiled object module for POLY.FTN, the FORTRAN application program:

```
$RU LINK)
LINK-11 V011A
#POLY<POLY,DLIB.LIB/L,FTNLIB/L/E)
```

The linked program can then be executed by:

```
$RU POLY)
```

## H.5.0 ASSEMBLY INSTRUCTIONS

### H.5.1 Assembling the Loader

LOADGT need be assembled only if it is to be used on a DOS/BATCH configuration where the DL11 interface to the GT40 is not located at a trap address of 300 and device addresses of 175610 to 175616. In that case, the user should procure a copy of the loader source and follow the procedures described below.

Use MACRO (Refer to DOS/BATCH-11 ASSEMBLER MANUAL (DEC-11-LASMA-A-D) to assemble the new loader, entering the values of the trap address and device address from the keyboard during the first pass of the assembler. All command lines and text lines below are terminated with a carriage return except END, which must be terminated with two (2) carriage returns. The ↑C represents a CTRL C combination.

```
$RU MACRO
MACRO V001A
#LOADGT<KB:/PA:1,DK:LOADGT.MAC
DLVEC=xxx           ;trap address
DLIN=yyyyyy         ;device address (of
                    ;DL11 status word)
↑C
.END              (terminate with (2) carriage returns)
```

The result will be a new LOADGT.OBJ which can be linked and used in place of the object module supplied with the package.

### H.5.1.1 Linking the Loader Object Module

The LOADGT module should be linked on the machine on which it will be used. It is supplied as an object module. After placing it on the system device (or reassembling it, using the instructions above) the following procedure can be used to link the module.

```
$RU LINK
LINK V11A01
#LOADGT<LOADGT.OBJ/E

TRANSFER ADDRESS: 075042
LOW LIMIT: 075042
HIGH LIMIT: 077460

LINK V11A01
#↑C
.KI
```

## H.5.2 Creating the DLIB.LIB Library

The library of PICTURE BOOK calls is a relocatable collection of FORTRAN subroutines (contained on BOOK.FTN) and MACRO subroutines (INCH, OUTCH, and SETUP). If the object modules of these subroutines are not available, they can be created from the sources using the following procedure:

```
$RU FORTRN
FORTRAN V004A
#BOOK<BOOK.FTN
#↑C
.KI
$RU MACRO
MACRO V001A
#INCH<INCH.MAC
END OF PASS 1
ERRORS DETECTED:  0
FREE CORE:  nnnn Words (Where nnnn=# of free words)
#OUTCH<OUTCH.MAC
END OF PASS 1
ERRORS DETECTED:  0
FREE CORE:  nnnn Words (Where nnnn=# of free words)
#SETUP<SETUP.MAC
END OF PASS 1
ERRORS DETECTED:  0
FREE CORE:  nnnn Words (Where nnnn=# of free words)
#↑C
.KI
```

With the object modules available, the DOS/BATCH librarian LIBR can be used to create the library, using this procedure:

```
$RU LIBR
LIBR V001A
#DLIB.LIB<,BOOK.OBJ,INCH.OBJ,OUTCH.OBJ,SETUP.OBJ
#↑C
.KI
$
```

NOTE

The comma precedes the book in the above command string because there is not an input library.

If the paper tape kit is purchased, a single paper tape containing object modules of INCH, OUTCH and SETUP is supplied. This should be put on disk as a single file, e.g., INOUT.OBJ. The librarian command string would then be:

```
$RU LIBR
LIBR V001A
#DLIB.LIB<,BOOK.OBJ,INOUT.OBJ
#↑C
.KI
$
```

H.5.3 Assembling BOOK

The assembly instructions for BOOK are as follows:

```
$RU MACRO
MACRO VR05A03
#BOOK<BOOK.003
END OF PASS 1
ERRORS DETECTED:   0
FREE CORE:   nnnn Words (Where nnnn=# of free words)
#↑C
.KI
$
```

## H.6.0 SAMPLE APPLICATION PROGRAMS

The following two examples show coding for the POLY and CLOCK programs, to be run under the DOS/BATCH-11 system.

### H.6.1 CLOCK

```
C CLOCK PLOTS A SINE WAVE BY PROJECTING THE Y COORDINATE OF A
C RADIUS OF A CIRCLE ONTO A SET OF AXES.
        INTEGER TIME(4)
        BYTE LINOUT(72)
C                       TIME CONTAINS ONE ASCII INTEGER PER CELL
        DATA TIME/'T','I','M','E'/
        CALL SCALE(0.,100.,0.,100.)
C                       FIRST,SCALE THE SCREEN SO THAT THE LOWER LEFT
C                       HAND COORDINATES ARE (0.,0.) AND THE UPPER
C                       RIGHT HAND COORDINATES ARE (100.,100.)
        CALL SETUP(1)
        L=LAYOUT(10,72,2,20,1,100,2,127,2,10,0)
C                       LAYOUT THE BOOK TO CONTAIN 2 PICTURES,1 FIGURE,
C                       2 GRAPHS, AND 1 TABLE.
C                       NOTE THAT GRAPHS ARE 127 POINTS LONG, THE MAX.
C                       LENGTH FOR ANY PAGE. WRITING IN THE 128TH LINE,
C                       AS THIS PROGRAM DOES, EFFECTIVELY COMBINES THE
C                       PAGES INTO ONE LONGER PAGE.
        IF(L.LT.0)STOP
C                       ON A BAD LAYOUT(NOT ENOUGH CORE)STOP
        CALL MOVE(0.,0.)
C                       ZERO OUT LINE 0 OF PICTURE 0
        CALL SET(5.,25.)
C                       AND PUT AN ABSOLUTE POINT IN LINE 1
        CALL DOTFIG(0,0)
C                       CALL FIGURE 0 AS A SERIES OF DOTS
        CALL ARC(20.,50,6.28,0.,1.)
C                       AND IN FIGURE 0,THE CURRENTLY OPEN FIGURE,
C                       DRAW A CIRCLE OF 50 SIDES
        CALL DOT(15.,25.)
C                       NOW DRAW THE CENTER OF THE CIRCLE
        CALL VECTOR(0.,10.)
C                       AND A VERTICAL RADIUS
        CALL BITS(0,2,2,0)
C                       MAKE THE NEXT ENTRY DIM AND SHORTDASH
        CALL VECTOR(100.,0.)
C                       DRAW A VECTOR WITH THESE MODES THAT GOES OFF THE
C                       SCREEN HORIZONTALLY TO THE RIGHT,
C                       IT WILL START WHERE THE RADIUS ENDS.
        CALL BITS(0,5,0,0)
C                       NOW BRIGHTEN UP THE NEXT ENTRY
        CALL SET(30.,25.)
C                       DRAW AN INVISIBLE ABSOLUTE POINT,
C                       IT WON'T MOVE WHEN THE RADIUS MOVES.
        CALL PICTUR(1,0)
C                       AND CALL PICTURE 1
        CALL SET(30.,25.)
C                       AGAIN SET AN ABSOLUTE POINT
        CALL XGRAPH(0,0)
```

```
C                       AND CALL THE GRAPH
      CALL SET(70.,22.)
C                       ONCE AGAIN SET A POINT
      CALL TABLE(0,0)
C                       AND CALL THE TABLE
C                       NOW PICTURE 0 CONTAINS A NUMBER OF CALLS TO
C                       PAGES NOT YET DRAWN INTO.
      CALL OPENP(1)
C                       OPEN PICTURE 1
      CALL VECTOR(75.,0.)
C                       AND IN IT DRAW A SET OF AXES
      CALL MOVE(-75.,10.)
      CALL VECTOR(0.,-20.)
C                       THESE WILL DISPLAY SINCE PICTURE 0 CALLS


C                       PICTURE 1
      CALL TEXT(4,TIME)
C                       IN THE OPEN TABLE, TABLE 0, WRITE 'TIME'
C                       IT,TOO,WILL DISPLAY.
      CALL OPENP(0)
C                       NOW REOPEN PICTURE 0
      STEP=3.1415/50.
C                       SET AN INCREMENT 1/100 OF 360 DEGREES
      R=10.
C                       SET THE RADIUS OF THE CIRCLE TO 10
      CALL PLOT(-7)
C                       SET THE GRAPH INCREMENT TO 7(UNSCALED)
      DO 5 I=1,10
C                       AND MOVE THE CURSOR DOWN 10 LINES
      CALL CURSOR(-1,0)
5     CONTINUE
      DO 10 I=1,100
C                       THIS LOOP WILL MOVE THE RADIUS AROUND THE
C                       CIRCLE LIKE THE HAND OF A CLOCK
C                       AND PLOT ITS Y COORDINATE ON THE GRAPH.
      THETA=STEP*FLOAT(I)
C                       THETA CONTAINS THE ANGLE
      X=R*SIN(THETA)
C                       X ITS X COORDINATE
      Y=COS(THETA)
C                       TAKE TIME OUT TO WRITE THE COSINE OF THETA
      ENCODE(11,100,LINOUT)Y
100   FORMAT('+','Y=',F8.5)
      CALL OUTLIN(11,LINOUT)
      Y=Y*R
C                       THEN CALCULATE THE Y COORDINATE OF THE RADIUS
      CALL MARKP(4)
C                       POINT PICTURE 0'S MARKER TO LINE 4
C                       WHERE THE RADIUS WAS DRAWN
      CALL VECTOR(X,Y)
C                       AND REDRAW THE RADIUS
      HT=Y+25.
C                       NOW CONVERT ITS Y COORDINATE TO AN UNSCALED
C                       QUANTITY BY ADDING THE Y COORDINATE OF THE
C                       CENTER OF THE CIRCLE
      IHT=HT*1024./100.
C                       AND CHANGING THE SCALING
      CALL PLOT(IHT)
C                       PLOT THE RESULTING VALUE
```

```
10      CONTINUE
        CALL BELL
C                           RING THE BELL, WHICH ALSO STOPS THE PROGRAM
        STOP
        END
```

H.6.2 POLY

```
FORTRAN V004A                    18:15:09    02-OCT-73   PAGE    1

        C THIS PROGRAM ALLOWS THE USER TO DISPLAY A POLYGON OF UP TO 9 SIDES
        C BY TOUCHING A MENU WITH THE LIGHT PEN AND THEN TO MOVE THE POLYGON
        C TO ANY LOCATION ON THE SCREEN AND LEAVE IT THERE

0001            COMMON/MARKS/MP,MF,MG,MT
        C                       MARKS CONTAINS THE COMMON MARKERS
0002            CALL SETUP(1,1)
0003            CALL SCALE(0.,1023.,0.,1023.)
        C                       SCALE THE SCREEN TO CORRESPOND TO SCREEN UNITS
        C                       SO THAT LINEX VALUES NEED NOT BE UNSCALED
0004            CALL LAYOUT(10,72,1,127,20,20,0,0,7,1,0)
        C                       CREATE A BOOK WITH 20 FIGURES AND A ONE LINE TAB
        C                       FOR EACH ENTRY IN THE MENU
0005            CALL MARKP(3)
        C                       STARTING AT LINE 3 OF PICTURE 0, ENTER
        C                       CALLS TO THE SEVEN TABLES
0006            CALL BITS(0,7,0,1)
        C                       MAKE THESE CALLS LIGHT PEN SENSITIINE AVD BRIGHT
0007            DO 1 I=3,9
0008            CALL SET(FLOAT((I-2)*100),700.)
        C                       FIRST SET LOCATION OF THIS ENTRY ON THE SCREEN
0009            CALL TABLE(I-3,0)
        C                       ENTER THE CALL
0010            CALL OPENT(I-3)
        C                       THEN OPEN THE CORRESPONDING TABLE
0011            CALL OUTCH(4)
        C                       AND INSTEAD OF USING 'CALL TEXT', OUTPUT
        C                       THE ASCII STRING THAT TEXT WOULD OUTPUT
        C                       FIRST OUTPUT CONTROL D
```

```
0012            CALL OUTCH(81)
      C                         THEN THE DECIMAL EQUIVALENT OF Q
0013            CALL OUTCH(1)
      C                         THEN INDICATE 1 CHARACTER WILL FOLLOW
0014            CALL OUTCH(48+I)
      C                         NEXT COMES THE DIGIT
0015  1         CONTINUE
0016            CALL BITS(0,5,0,0)
      C                         NOW TURN OFF SENSITIVITY
0017            CALL MOVE(0.,0.)
      C                         IN THIS DUMMY ENTRY IN THE PICTURE
0018            IFIG=0
      C                         SET COUNTER TO 0
0019  2         MARK=MP
      C                         AND SAVE THE CURRENT VALUE OF THE PICTURE MARKER
0020            CALL MARKP(1)
      C                         NOW CLEAR A COUPLE OF LINES BY ENTERING INVISIBL
      C                         0 LONG

0021            CALL MOVE(0.,0.)
0022            CALL MOVE(0.,0.)
0023            CALL MARKP(0)
      C                         RESET THE MARKER TO THE START OF THE PAGE
0024            CALL BITS(0,5,2,1)
      C                         AND SET LIGHT PEN SENSITIVITY ON AND SHORTDASH
0025            CALL MOVE(0.,0.)


FORTRAN V004A                   18:15:09    02-OCT-73   PAGE    2

      C                         CLEAR OUT PICTURE 0, LINE 0, WHERE LIGHT PEN
      C                         HIT COORDINATES WILL ENTER
0026            CALL UNHIT(I,J)
      C                         NOW WAIT FOR A HIT ON THE MENU
0027            J=J/2+1
      C                         DETERMINE THE DIGIT HIT FROM THE LINE
      C                         NUMBER THAT CAUSED THE HIT
0028            R=FLOAT(J)*10.
      C                         AND LET THE POLYGON'S SIZE ALSO DEPEND ON THE
      C                         NUMBER OF ITS SIDES IT HAS
0029            CALL OPENF(IFIG)
      C                         NOW OPEN THE NEXT FIGURE
0030            CALL ARC(R*2.,J,6.28,0.,1.)
      C                         AND DRAW AN ARC IN IT OF J SIDES, I.E. A POLYGON
0031            CALL MOVE(-R,0.)
      C                         DRAW AN INVISIBLE VECTOR TO OFFSET THE START OF
      C                         POLYGON FROM THE END OF THE LIGHT PEN COORDINATE
      C                         VECTOR SO THAT THE POLYGON CAN MOVE IN
      C                         ANY DIRECTION
0032            CALL VECFIG(IFIG,0)
      C                         AND DISPLAY THE NEW FIGURE
0033            CALL INNUM(NN)
      C                         NOW WAIT FOR A CARRIAGE RETURN WHILE THE USER
      C                         MOVES THE POLYGON AROUND THE SCREEN
0034            I=LINEX(0,0)
      C                         WHEN IT HAPPENS, READ THE CURRENT COORDINATES
      C                         OF THE LIGHT PEN VECTOR, I.E. THE CENTER OF THE
      C                         POLYGON
```

```
0035            J=LINEY(0,0)
0036            CALL MARKP(MARK)
        C                       MARK THE PICTURE USING THE SAVED MARKER
0037            CALL SET(FLOAT(I)-R,FLOAT(J))
        C                       ENTER A SET TO THE CORNER OF THE POLYGON
0038            CALL VECFIG(IFIG,0)
        C                       AND CALL THE FIGURE FROM THERE, ALSO, THUS
        C                       OVERLAYING THE SHORTDASH POLYGON WITH A SOLID ON
0039            IFIG=IFIG+1
        C                       POP THE COUNTER
0040            IF(IFIG.LE.19)GOTO 2
        C                       AND DO UP TO 20 FIGURES THIS WAY
0041            STOP
0042            END

        ROUTINES CALLED:
        SETUP , SCALE , LAYOUT, MARKP , BITS  , SET   , FLOAT
        TABLE , OPENT , OUTCH , MOVE  , UNHIT , OPENF , ARC
        VECFIG, INNUM , LINEX , LINEY

        BLOCK       LENGTH
        MAIN.   564     (002150)*
        MARKS   8       (000020)

        **COMPILER ----- CORE**
           PHASE       USED  FREE
        DECLARATIVES 00365 05487
        EXECUTABLES  00859 04994
        ASSEMBLY     01227 07543
```

# INDEX

# HOW TO OBTAIN SOFTWARE INFORMATION

## SOFTWARE NEWSLETTERS, MAILING LIST

The Software Communications Group, located at corporate headquarters in Maynard, publishes newsletters and Software Performance Summaries (SPS) for the various Digital products. Newsletters are published monthly, and contain announcements of new and revised software, programming notes, software problems and solutions, and documentation corrections. Software Performance Summaries are a collection of existing problems and solutions for a given software system, and are published periodically. For information on the distribution of these documents and how to get on the software newsletter mailing list, write to:

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

## SOFTWARE PROBLEMS

Questions or problems relating to Digital's software should be reported to a Software Support Specialist. A specialist is located in each Digital Sales Office in the United States. In Europe, software problem reporting centers are in the following cities.

| | |
|---|---|
| Reading, England | Milan, Italy |
| Paris, France | Solna, Sweden |
| The Hague, Holland | Geneva, Switzerland |
| Tel Aviv, Israel | Munich, West Germany |

Software Problem Report (SPR) forms are available from the specialists or from the Software Distribution Centers cited below.

## PROGRAMS AND MANUALS

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center.

Digital Equipment Corporation
Software Distribution Center
146 Main Street
Maynard, Massachusetts 01754

Digital Equipment Corporation
Software Distribution Center
1400 Terra Bella
Mountain View, California 94043

Outside of the United States, orders should be directed to the nearest Digital Field Sales Office or representative.

## USERS SOCIETY

DECUS, Digital Equipment Computer Users Society, maintains a user exchange center for user-written programs and technical application information. A catalog of existing programs is available. The society publishes a periodical, DECUSCOPE, and holds technical seminars in the United States, Canada, Europe, and Australia. For information on the society and membership application forms, write to:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

DECUS
Digital Equipment, S.A.
81 Route de l'Aire
1211 Geneva 26
Switzerland

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form (see the HOW TO OBTAIN SOFTWARE INFORMATION page).

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

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you do not require a written reply, please check here. ☐

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

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

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

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

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

Digital Equipment Corporation
Maynard, Massachusetts

digital

printed in U.S.A.