DIGITAL RESEARCH™

CBASIC Compiler™

Language

Graphics Guide

CBASIC® Compiler Language Graphics Guide

Digital Research
P.O. Box 579
801 Lighthouse Avenue
Pacific Grove, CA 93950
TWX 910 360 5001

The CBASIC Compiler Language Graphics Guide was prepared using the Digital Research TEX Text Formatter and printed in the United States of America.

****
* First Edition: May 1983 *
****

Foreword

CBASIC® Compiler is Digital Research's powerful compiler version of CBASIC, the commercial BASIC dialect recognized as the industry standard. With CBASIC Compiler's comprehensive graphics extensions, you can now write versatile graphics programs for a multitude of applications.

Both the 8-bit and 16-bit versions of CBASIC Compiler support graphics extensions. To write graphics programs, you need CBASIC Compiler and the GSX™ Graphics System Extension that fits your CP/M® operating system and hardware.

CBASIC Compiler's graphics extensions are device-independent; you can direct output to any graphics peripheral without recompiling your programs. Your GSX software issues all the necessary commands to control the peripherals you select.

For example, if you do not have a plotter, you needn't worry about converting your programs for a plotter when one becomes available. You simply change the GSX ASSIGN.SYS file so that it assigns your output to the plotter instead of to the screen. Your GSX Graphics Extensions Programmer's Guide shows you how to modify ASSIGN.SYS.

The GSX software contains routines that control the peripheral devices. These routines, known as device drivers, provide physical control of the devices. Your GSX guide includes specifications for all the device drivers you can use with your software.

Get familiar with your software and hardware before you attempt extensive application programming. You can determine most of your system's features by watching DEMOGRAF, the CBASIC Compiler graphics tutorial.

This CBASIC Compiler Language Graphics Guide describes CBASIC Compiler's graphics statements and functions with the assumption that you are acquainted with CBASIC Compiler and GSX graphics.

- Section 1 defines the concepts underlying CBASIC Compiler graphics.
- Section 2 explains how to compile and link CBASIC graphics programs and discusses DEMOGRAF, a demonstration of CBASIC Compiler's graphics statements and functions. DEMOGRAF is included on your demonstration disk.
- Section 3 catalogs all the graphics statements and functions.
- Section 4 presents sample graphics programs.
- Appendix A is a listing of DEMOGRAF.

Table of Contents

Table of Contents
(continued)

Appendixes

Tables, Figures, and Listings

Tables

Figures

Listings

Section 1
Introduction to CBASIC Compiler Graphics

This section introduces the CBASIC Compiler graphics statements and functions and defines some elementary graphics concepts.

1.1 Graphics Statements

The graphics statements and functions fall into five groups, according to function. Each statement or function is described individually in Section 3. The names of CBASIC Compiler graphics statements and functions are reserved words.

Table 1-1. Graphics Extensions

Group	Statement or Function
OUTPUT	GRAPHIC PRINT
	MAT FILL
	MAT MARKER
	MAT PLOT
	PLOT
FORMAT	CHARACTER HEIGHT (SET/ASK)
	COLOR (SET/ASK)
	COLOR COUNT (ASK)
	JUSTIFY (SET/ASK)
	LINE STYLE (SET/ASK)
	MARKER HEIGHT (SET/ASK)
	MARKER TYPE (SET)
	STYLE COUNT (ASK)
	TEXT ANGLE (SET/ASK)
VIEWING AREA	BOUNDS (SET/ASK)
	DEVICE (ASK)
	VIEWPORT (SET/ASK)
	WINDOW (SET/ASK)

Table 1-1. (continued)

Group	Statement or Function
INPUT	GRAPHIC INPUT
CONTROL	BEAM (SET/ASK) CLEAR CLIP (SET/ASK) GRAPHIC CLOSE GRAPHIC OPEN POSITION (SET/ASK)

1.2 Definitions

You should understand the following concepts before you turn to the statement and function definitions.

1.2.1 Coordinates

Positions within the display area of the graphics device are defined by X and Y coordinates. The X axis is the horizontal axis. The Y axis is the vertical axis. Both coordinate axes begin at the lower left corner of the device.

When you initialize the graphics system with the GRAPHIC OPEN statement, the coordinates initially range from 0.0 to 1.0 for both axes, regardless of the physical dimensions of the device.

After you initialize a device with the GRAPHIC OPEN statement, your program can address the lower left corner of the display device with X and Y coordinates 0,0. The upper right corner of the device is address 1,1. You define a box around the border of your graphics output device by connecting the coordinate pairs (0,0), (0,1), (1,1), (1,0), (0,0).

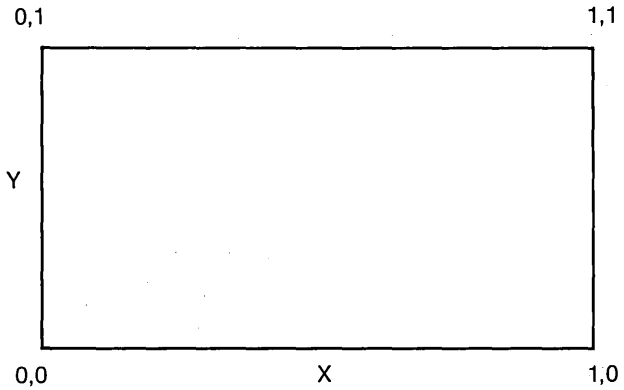

Figure 1-1. Device Coordinates

1.2.2 Bounds

The bounds of a device are its physical dimensions in pixels, inches, or whatever measure a particular device uses. Frequently, these dimensions are not the same for the X and Y axes. The height of the drawing area might not be equal to the width. With the BOUNDS statement, you can control the the length of the axes, thus keeping your images proportioned, regardless of the device you are using.

After a GRAPHIC OPEN statement, the bounds of the device are set to 100% of the physical extent of the X and Y axes. If there is a difference in the extents of the two axes, a line along the X axis is not the same length as a line along the Y axis. For example, the Y axis might be shorter relative to the X axis. The ratio of the Y axis to the X axis is called the aspect ratio.

You can use the DEVICE statement to determine the aspect ratio of a device. For example, if the Y axis is 80% as long as the X axis, the ASK DEVICE statement returns 1.0 and .8 as the relative values of the X and Y axes.

The BOUNDS statement is the first basic dimensioning statement. The second and third are the VIEWPORT and WINDOW statements.

1.2.3 Viewport

Within the bounds of a device, the area in which graphics data prints is called the viewport. You define the viewport by X and Y coordinates ranging from 0.0 to 1.0.

Use the VIEWPORT statement to define the literal viewing area within the physical bounds of the device. VIEWPORT lets you specify beginning and end points for the X and Y axes within the device's current bounds. Subsequent graphics statements operate inside the area you specify in the VIEWPORT statement.

Figure 1-2 illustrates a viewport. You establish this viewport with the following statement:

```
SET VIEWPORT .2,.5,.2,.6
```

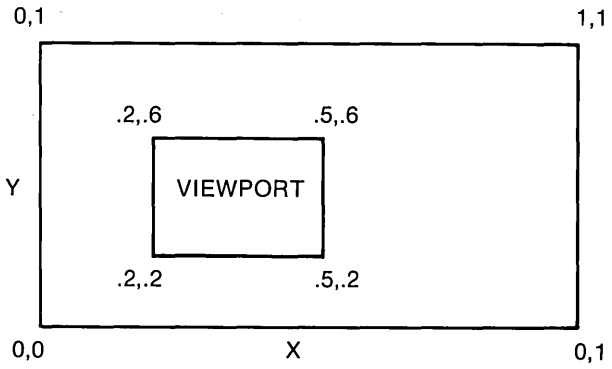

Figure 1-2. Viewport

All graphics statements after the VIEWPORT statement refer to the area within this viewport.

In this case the bounds of the device are 100% of the available capacity. If you use a SET BOUNDS statement to alter the extents of the device, the viewport automatically adjusts within the new bounds.

1.2.4 Window

You can think of a window as a frame with tick marks around the viewport. The window defines the scale of the X and Y coordinates of a viewport. With the WINDOW statement, you can restate the scale of the viewport coordinate system to whatever values your application requires.

The WINDOW statement lets you automatically map real-world values onto the coordinate system of your device. The X and Y coordinates of a viewport initially range from 0.0 to 1.0. You can change the initial viewport ranges with the WINDOW statement. WINDOW can adjust these ranges to the scale required by your application. For example,

```
SET WINDOW 0,100,0,100
```

scales all coordinate references in subsequent CBASIC graphics statements to a range of 0 to 100 for both axes.

In summary, the window scales the viewport; the viewport resides within the bounds; the bounds reside within the physical extents of the device. For further information, see the explanations of the BOUNDS, VIEWPORT, and WINDOW statements in Section 3.

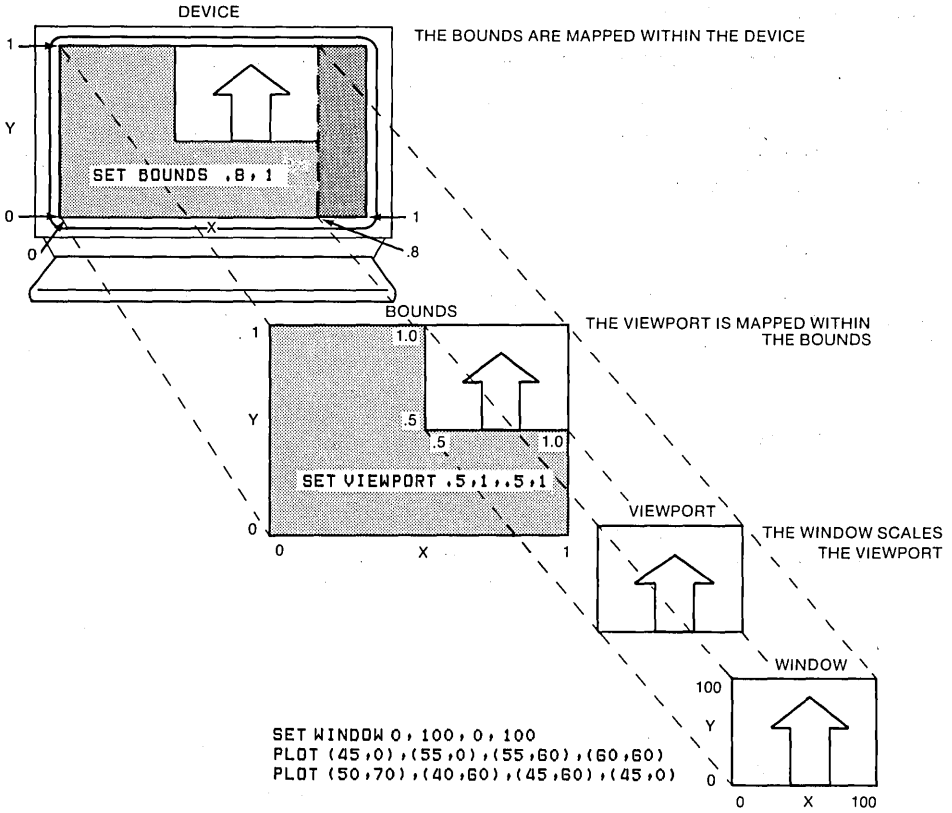

Figure 1-3. WINDOW

1.2.5 Graphics Cursor

The cursor indicates the current location of the drawing device. The GRAPHIC INPUT statement has a special graphics cursor that you can maneuver around the screen. How you control the movement of the graphics cursor (arrow keys, control keys, or a mouse), depends on your hardware. See your GSX Graphics Extension Programmer's Guide.

1.2.6 Beam

The beam is the drawing device within a particular output device. For a plotter, the beam is a pen. For a CRT, the beam is a stream of electrons. You turn the beam on or off with the BEAM statement.

1.2.7 Marker

Markers are predefined symbols you use to identify points or intersections on a graph or drawing. You can use the MAT MARKER statement to plot a scatter graph of points with markers. The available markers are described in the documentation for your device driver. See the MARKER TYPE statement in Section 3.

1.2.8 Clipping

Sometimes the coordinate references you give are out-of-bounds, extending beyond the bit-map memory area that some devices use to generate displays. Clipping refers to the chopping off of parts of drawings that would otherwise trespass beyond this area. Without clipping, memory outside the bit-map area might be altered.

Use the CLIP statement to turn clipping on or off. Clipping is on by default at the beginning of a program. Turning clipping off increases operating speed slightly, but is risky. Turn clipping off only if you are sure that your application program will not try to reference coordinates outside the current window. See the CLIP statement in Section 3.

End of Section 1

Section 2
Compiling and Linking

To create object programs from CBASIC Compiler graphics programs, you need special files and procedures. This section explains these requirements and tells you how to compile, link, and run DEMOGRAF, the graphics demonstration program.

2.1 Compilation

Your source program must include the following statement:

```
%INCLUDE GRAPHCOM.BAS
```

The statement above assumes the file GRAPHCOM.BAS is on the same disk drive as the source program. If it is on a different drive, precede the filename in the %INCLUDE statement with a drive specification. For example,

```
%INCLUDE B:GRAPHCOM.BAS
```

The GRAPHCOM.BAS file contains variable names to include in the common program area. The variable names in this file are reserved words in CBASIC Compiler programs that use graphics statements.

2.2 Linking

You need no special procedures to link CBASIC Compiler graphics programs. Follow the instructions in your CBASIC Compiler documentation.

2.3 GENGRAF

Use the GENGRAF program to incorporate run-time loaders into programs generated by the 8-bit CBASIC Compiler, CB80™. If you are a 16-bit CBASIC Compiler (CB86™) user, you do not need GENGRAF, nor do you need to follow this procedure. The form of the GENGRAF command is

```
GENGRAF <filespec>
```

GENGRAF expects to find a .COM file that was output from the 8-bit linker, LK80™. GENGRAF modifies the .COM file by including a run-time loader for GSX. When GENGRAF finishes, your program is ready to run.

2.4 Run-time

At run-time, the following files must be on the current default drive:

CBASIC Compiler (CB80) Users

- GSX.SYS
- ASSIGN.SYS
- Device drivers as required by ASSIGN.SYS

CBASIC Compiler (CB86) Users

- GRAPHICS.CMD
- ASSIGN.SYS
- Device drivers as required by ASSIGN.SYS

See your GSX Graphics Extension Programmer's Guide for details regarding these files.

2.5 DEMOGRAF

A graphics tutorial, DEMOGRAF.BAS, is included on your CBASIC software disk. It contains examples of all the graphics statements. The examples are the same as those shown in the definitions of the graphics statements and functions in Section 3. The tutorial is in alphabetical order like Section 3. A program listing of DEMOGRAF.BAS is included in Appendix A.

To use the demonstration program, you must first compile and link the source program DEMOGRAF.BAS, using one of the following procedures:

- CBASIC Compiler (CB80) Users

```
A>CB80 DEMOGRAF

A>LK80 DEMOGRAF

A>GENGRAF DEMOGRAF
```

- CBASIC Compiler (CB86) Users

  A>**CB86 DEMOGRAF**

  A>**LINK86 DEMOGRAF**

  A>**GRAPHICS** (load GSX-86™)

To run the DEMOGRAF program, enter the following command:

A>**DEMOGRAF**

DEMOGRAF contains statement demonstrations that display the name of the statement and show how it works. Read the explanations of the graphics statements in Section 3 as you watch DEMOGRAF.

DEMOGRAF pauses at the end of each example. Press any key to continue. Enter CTRL-C to interrupt and return to the operating system.

End of Section 2

Section 3
Graphics Statements and Functions

This section presents the CBASIC Compiler graphics statements and functions in alphabetical order. Statement keywords preceded by SET and ASK are alphabetized by the keyword. For example, the SET WINDOW statement is found under WINDOW.

The syntax notation in this guide employs the following conventions.

- Upper-case letters designate CBASIC Compiler keywords.
- Lower-case letters indicate variables.
- Angle brackets < > enclose syntactic items.
- Square brackets [ ] enclose optional items.
- Braces { } enclose optional items that can be repeated.
- The OR bar, |, indicates a choice between two or more syntactic items.

You must include all other punctuation in the syntax line, such as delimiters and parentheses. Table 3-1 defines some syntactic items:

Table 3-1. Syntax Definitions

Syntactic Item	Definition
numeric expression	numeric constants (integer or real), numeric variables, or combination of constants, variables, and numeric operators
real variable	floating point variable
integer variable	integer (-32768 to 32767) variable
string variable	character string ($ type) variable
X coordinate	horizontal position in coordinate system
Y coordinate	vertical position in coordinate system

BEAM Statement

The BEAM statement turns the pen or drawing beam on or off.

Syntax:

```
SET BEAM "ON" | "OFF"
ASK BEAM <string variable>
```

Explanation:

Switch the beam on or off by writing a SET BEAM statement with "ON" or "OFF", upper-case and in quotation marks, as the argument. This statement is like a PEN UP/PEN DOWN statement for a plotter.

The ASK BEAM statement tells you whether the beam is currently on or off, returning the value of the beam state in <string variable>.

Example:

```
GRAPHIC PRINT AT (0,.9): "BEAM STATEMENT"
SET BEAM "OFF"
PLOT (0,1),(1,1),(1,0),(0,0)
KEY%=PAUSE
REM  PAUSE IS A LOCALLY DEFINED FUNCTION
REM  THAT WAITS FOR CONSOLE INPUT.
CLEAR
SET BEAM "ON"
PLOT (0,1),(1,1),(1,0),(0,0)
KEY%=PAUSE
```

In this example, the SET BEAM "OFF" statement turns off the beam before the PLOT statement runs. Only three sides of a screen border are drawn, because the beam is OFF before the first pair of coordinates is plotted (see Figure 3-1).

PAUSE is locally defined at the beginning of DEMOGRAF. The pause halts the program so you can see the display. Press any key to continue. To interrupt the program, enter CTRL-C.

The CLEAR statement turns the beam off and leaves it at 0,0. The SET BEAM "ON" statement draws a line from (0,0) to (0,1), and draws the full screen border.

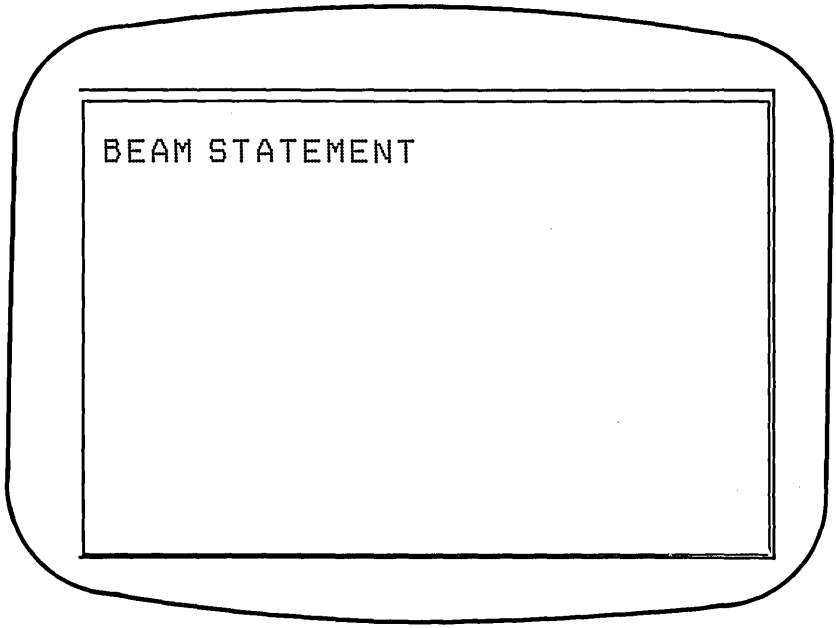

Figure 3-1. The BEAM Statement

BOUNDS Statement

The BOUNDS statement sets the aspect ratio of the X and Y axes of the output device.

Syntax:

```
SET BOUNDS <height>,<width>
ASK BOUNDS <height>,<width>
```

Explanation:

Graphics devices often have different horizontal and vertical dimensions. The ratio of the length of the axes is called the aspect ratio.

The BOUNDS statement changes the aspect ratio. (To find the aspect ratio of the device, use the ASK DEVICE statement.) The values you give for the new height and width of the device must be greater than 0.0 and less than or equal to 1.0. At least one of the values must be equal to 1.0.

The new boundaries of the device are always anchored at coordinates 0,0, as shown in Figures 3-2 and 3-3. All subsequent graphics operations are bounded by the new height and width of the SET BOUNDS statement.

ASK BOUNDS assigns the current vertical and horizontal dimensions of the screen to the real variables <height> and <width>.

With the BOUNDS statement, you can make different display devices proportional. Proportional devices retain the shapes of figures so that, for example, your circles are circles from device to device. Your application program can adjust the aspect ratio of any device to fit a planned ratio. The BOUNDS statement is only one way to change the aspect ratio; you can also change it with the WINDOW statement.

Example:

You can use the BOUNDS statement with the ASK DEVICE statement to alter the aspect ratio of a device.

```
CLEAR
GRAPHIC PRINT AT (0,.9):  "BOUNDS STATEMENT"
ASK DEVICE X.AXIS,Y.AXIS
PRINT "THE ASPECT RATIO IS= "; Y.AXIS; "/"; X.AXIS
KEY% = PAUSE
REM  PAUSE IS A LOCALLY DEFINED FUNCTION
PLOT (0,0),(0,1),(1,1),(1,0),(0,0)
KEY% = PAUSE
CLEAR
SET BOUNDS Y.AXIS,X.AXIS
PLOT (0,0),(0,1),(1,1),(1,0),(0,0)
SET BOUNDS 1,1
```

The second PLOT statement draws a perfect square because reversing the physical dimensions in the SET BOUNDS statement squares the aspect ratio of the screen.

The SET BOUNDS 1,1 statement returns the screen to its full X and Y capacities.

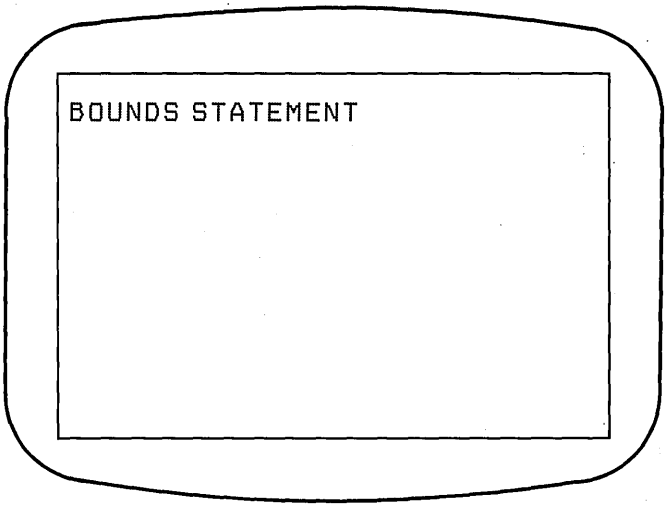

Figure 3-2. The BOUNDS Statement--Rectangle

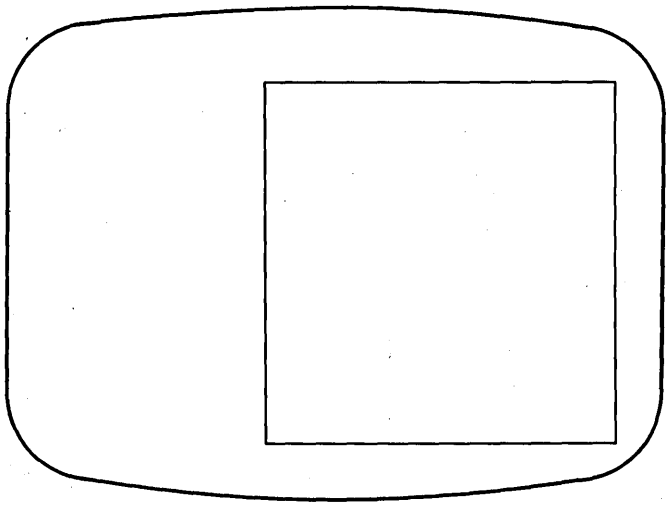

Figure 3-3. BOUNDS Statement--Square

CHARACTER HEIGHT Statement

The CHARACTER HEIGHT statement defines the height of characters or assigns the height to a variable.

Syntax:

```
SET CHARACTER HEIGHT <numeric expression>
ASK CHARACTER HEIGHT <real variable>
```

Explanation:

SET CHARACTER HEIGHT defines the height of characters relative to the length of the Y coordinate. Initially, Y extends from 0.0 to 1.0. A character height of .2 results in characters that are 20% of the length of the Y coordinate.

The argument you give for <numeric expression> is the desired height of the character. The resulting character height is the largest hardware character size that does not exceed the size you requested. Every output device has a number of character sizes. You can find the character sets of the output devices available to your system in your GSX reference manual.

SET CHARACTER HEIGHT 0 sets the character height to the minimum size possible within the current window.

ASK CHARACTER HEIGHT assigns the current character height value to the <real variable>. You can use ASK CHARACTER HEIGHT to find the height assigned by a previous SET CHARACTER HEIGHT statement. The actual value of a character height can differ from what you specify in a SET CHARACTER HEIGHT statement, because the choice of character heights is determined by the available character set.

You can change the extent of the X and Y coordinates with the WINDOW statement. After such a change, any CHARACTER HEIGHT statements use the new X and Y coordinate extents as a base. If you use WINDOW to change the Y coordinate to 0.0 to 100.0, the correct value for a 20% character height is 20.0.

If characters to be displayed by the GRAPHIC PRINT statement exceed the limits of the current window, they are not displayed at all. It is good practice to reestablish the character height after a SET WINDOW statement to ensure that the character set fits the new window dimensions.

For example, the minimum character height in a large window is greater than the minimum character height in a small window. If you start with a large window and shrink the window without changing the size of the characters, the characters might not fit in a smaller window. If not, they do not print.

Examples:

The following statement sets the character height to 10% of the screen if the extent of the Y coordinate is 0.0 to 1.0.

```
SET CHARACTER HEIGHT .1
```

The following statement sets the character height to 15% (15/100) of the screen. The window statement has set the Y axis to a value ranging from 0 to 100.

```
SET WINDOW 0,100,0,100
SET CHARACTER HEIGHT 15
```

The next routine returns the minimum character height in the numeric variable CH:

```
SET CHARACTER HEIGHT 0
ASK CHARACTER HEIGHT CH
PRINT "MINIMUM DEVICE CHARACTER HEIGHT IS = "; CH
```

The CHARACTER HEIGHT demonstration in the DEMOGRAF program is performed by the following commands:

```
CLEAR
SET CHARACTER HEIGHT 0
GRAPHIC PRINT AT (0,.9):  "CHARACTER HEIGHT STATEMENT"
SET CHARACTER HEIGHT .1
GRAPHIC PRINT AT (0,.7):  "10 PERCENT"
KEY% = PAUSE
REM  PAUSE IS A LOCALLY DEFINED FUNCTION
SET WINDOW 0,100,0,100
SET CHARACTER HEIGHT 15
GRAPHIC PRINT AT (0,40):  "15 PERCENT"
KEY% = PAUSE
SET CHARACTER HEIGHT 0
ASK CHARACTER HEIGHT CH
PRINT "MINIMUM CHARACTER HEIGHT IS = "; CH
GRAPHIC PRINT AT (0,20):  "MINIMUM HEIGHT"
```

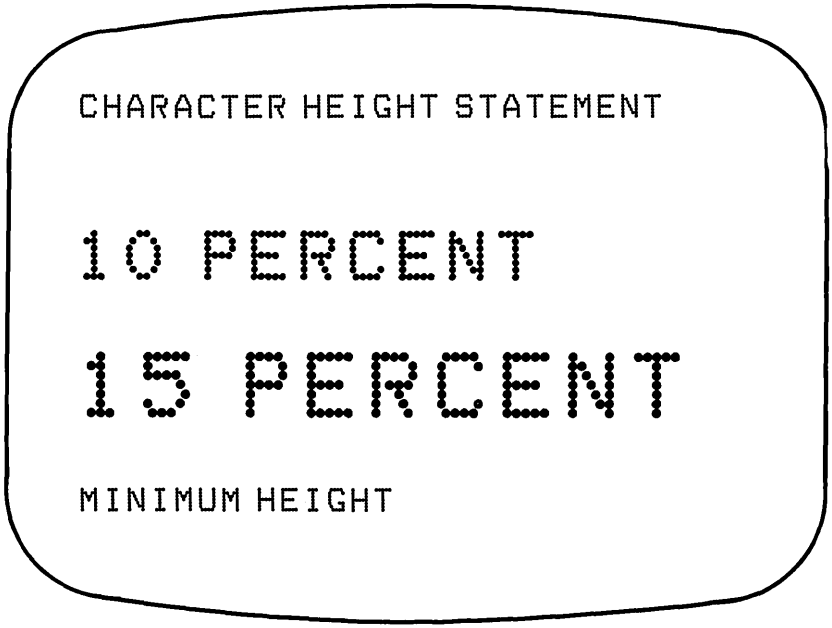

Figure 3-4. The CHARACTER HEIGHT Statement

CLEAR Statement

Syntax:

```
CLEAR
```

Explanation:

The CLEAR statement clears the screen, returns the cursor to (0,0), and turns the beam off.

Example:

```
INPUT ""; LINE SEED$
RANDOMIZE
CLEAR
GRAPHIC PRINT AT (0,90):  "CLEAR STATEMENT"
SET WINDOW 0,1,0,1
FOR I.INT% = 1 TO 10
    PLOT (RND,RND),(RND,RND)
NEXT I.INT%
KEY% = PAUSE
REM  PAUSE IS A LOCALLY DEFINED FUNCTION
CLEAR
```

This routine draws ten random line segments, pauses for you to press a key on the console, and clears the screen with CLEAR.

CLIP Statement

The CLIP statement turns clipping on or off.

Syntax:

```
SET CLIP "ON" | "OFF"
ASK CLIP <string variable>
```

Explanation:

SET CLIP requires a string expression "ON" or "OFF", upper-case and in quotation marks, as the argument. ASK CLIP returns the current value of the clip state in <string variable>.

Clipping edits portions of line segments or figures that extend outside the limits of the current window. With clipping on, memory areas in a bit-mapped display device are protected from possible overlay by out-of-range coordinate references.

It is advisable to keep clipping on. Otherwise, you might overlay data that the software or hardware needs to operate.

Example:

This program illustrates the effect of automatic clipping when a figure exceeds allowable boundaries. Figure 3-5 shows the unclipped image. With clipping OFF, the computer attempts to draw the area outside the window. Note that this can be undesirable. Figure 3-6 shows the same figure, clipped.

```
SET WINDOW 0,100,0,100
PLOT (25,10),(50,150),(75,10),(25,10)
END
```

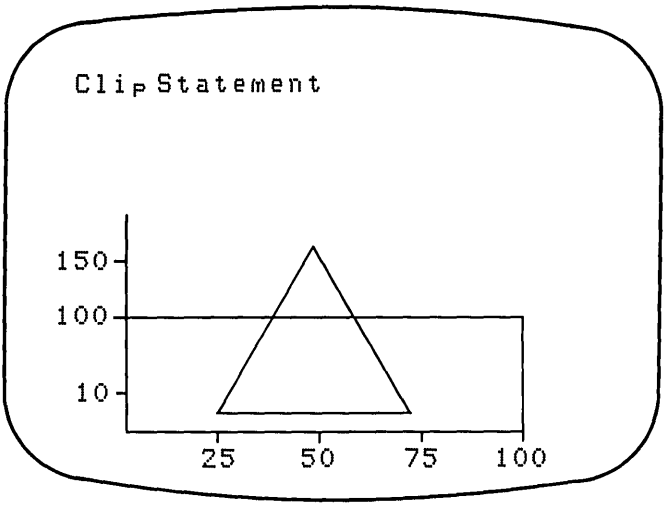

Figure 3-5. Unclipped Image

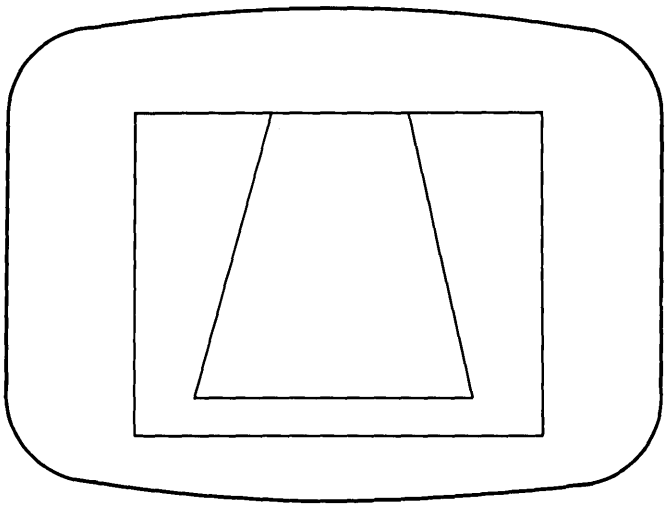

Figure 3-6. Clipped Image

COLOR Statement

The COLOR statement establishes the foreground color or assigns the value of the foreground color to a variable.

Syntax:

```
SET COLOR <color number>
ASK COLOR <integer variable>
```

Explanation:

The SET COLOR statement establishes the color of subsequent lines and filled polygons as one of n colors available on your device. The variable <color number> is an integer expression representing the desired color number.

You can find the number of colors available on your device with an ASK COLOR COUNT statement. The number and meaning of color numbers varies according to your implementation.

The ASK COLOR statement assigns the value of the current foreground color to <integer variable>.

Example:

```
CLEAR
GRAPHIC PRINT AT (0,90):  \
        "COLOR AND COLOR COUNT STATEMENTS"
SET WINDOW 0,1,0,1
ASK COLOR COUNT CT%
FOR I.INT% = 1 TO CT%
    SET COLOR I.INT%
    PLOT  (0,0),(0,1),(1,1),(1,0),(0,0)
    KEY% = PAUSE
REM  PAUSE IS A LOCALLY DEFINED FUNCTION
NEXT I.INT%
END
```

This routine draws a border around the screen in each of the device colors.

COLOR COUNT Statement

The COLOR COUNT statement assigns the number of available colors to a variable.

Syntax:

ASK COLOR COUNT <integer variable>

Explanation:

ASK COLOR COUNT assigns the number of available colors in <integer variable>.

Example:

See the COLOR Statement.

DEVICE Statement

The DEVICE statement assigns the physical limits of a device to variables.

Syntax:

ASK DEVICE <height variable>,<width variable>

Explanation:

The ASK DEVICE statement assigns the physical limits of the currently open graphics device to the real variables for height and width.

The physical limits of a graphics device are often not the same for the X and Y axes. A screen might have 60% as much viewing space vertically as horizontally. The ASK DEVICE statement returns the dimensions of the physical device.

Example:

```
CLEAR
SET CHARACTER HEIGHT 0
SET COLOR 1
GRAPHIC PRINT AT (0,.8):  "DEVICE STATEMENT"
ASK DEVICE X.AXIS,Y.AXIS
PRINT "THE VERTICAL AXIS IS ";\
      Y.AXIS*100/X.AXIS;"PERCENT OF THE"\
PRINT " HORIZONTAL AXIS"
PRINT "X= ";X.AXIS," Y= ";Y.AXIS
END
```

This program prints the values for the vertical and horizontal axes of the device. You can square a device for output by reversing the X and Y limits in a SET BOUNDS statement. See the BOUNDS statement for an example.

GRAPHIC CLOSE Statement

The GRAPHIC CLOSE statement closes the current graphics display device.

Syntax:

```
GRAPHIC CLOSE
```

Explanation:

The GRAPHIC CLOSE statement closes the graphics output device that you opened with a GRAPHIC OPEN statement.

After you GRAPHIC CLOSE a device, you can open a new device with GRAPHIC OPEN.

GRAPHIC INPUT Statement

The GRAPHIC INPUT statement accepts X and Y coordinates from the cursor position.

Syntax:

GRAPHIC INPUT <X coordinate>, <Y coordinate>, <string variable>

Explanation:

A GRAPHIC INPUT statement lets you position the cursor on your terminal using cursor control keys. When you press any key other than a cursor control key, the statement stores the X and Y coordinates of the cursor in <X coordinate> and <Y coordinate>. The character code for the key you press to complete the command is stored in the third operand, <string variable.>

The GRAPHIC INPUT statement lets you communicate with the program by indicating areas on the screen. Moving the cursor to an option on a menu is one use of this command. GRAPHIC INPUT also can store the coordinates of figures that you enter. Use the MAT PLOT and MAT FILL statements to manipulate these figures.

For information on how your device treats the graphics cursor, see your GSX Graphics Extensions manual.

Example:

```
SET WINDOW 0,100,0,100
SET CHARACTER HEIGHT 0
GRAPHIC PRINT AT (0,80): "GRAPHIC INPUT STATEMENT"
GRAPHIC PRINT AT (0,25): "OPTION 1  ."
SET COLOR 2
GRAPHIC PRINT AT (0,50): "OPTION 2  ."
SET COLOR 3
GRAPHIC PRINT AT (0,75): "OPTION 3  ."
GRAPHIC INPUT X.AXIS,Y.AXIS,A$
N = INT((Y.AXIS+5)/25)
IF N = 0 THEN N = 1       REM NO OPTION ZERO
IF N > 3 THEN N = 3       REM ONLY THREE OPTIONS
PRINT "THE CURSOR WAS POSITIONED AT: "; X.AXIS,Y.AXIS
PRINT "YOU SELECTED OPTION:          "; N
PRINT "THE TERMINATING KEY WAS:      "; A$
END
```

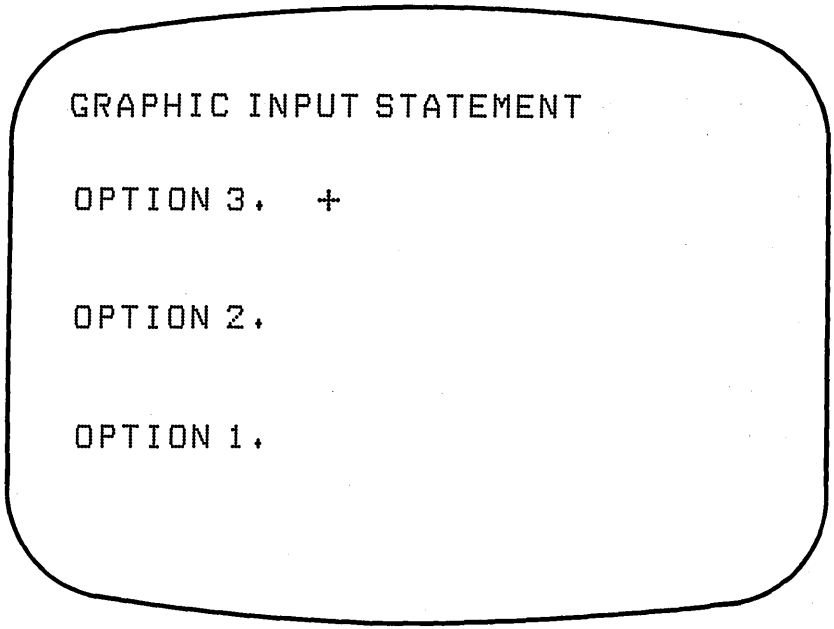

Figure 3-7. The GRAPHIC INPUT Statement

GRAPHIC OPEN Statement

The GRAPHIC OPEN statement initializes the graphics system and selects the output device.

Syntax:

GRAPHIC OPEN <integer expression>

Explanation:

The GRAPHIC OPEN statement initializes the graphics system and selects an output device such as a graphics terminal, plotter, or printer. The devices available to you depend upon your implementation.

The integer expression corresponds to an output device driver listed in the ASSIGN.SYS file by that number. Usually, device number 1 is the graphics terminal.

Example:

```
GRAPHIC OPEN 1
```

This example opens the terminal as the output device if the terminal has been defined as device driver 1 in the ASSIGN.SYS file. Your GSX documentation discusses the ASSIGN.SYS file and graphics devices.

GRAPHIC PRINT Statement

The GRAPHIC PRINT statement prints an alphanumeric string at a given point.

Syntax:

GRAPHIC PRINT AT (X,Y): <string constant or variable>

Explanation:

The GRAPHIC PRINT statement displays an alphanumeric string at (X,Y), the coordinates where printing is to begin. The (X,Y) variables are numeric expressions, scaled by the current window ranges. The string constant or variable contains the items to print.

You can use alphanumeric strings to label parts of a drawing. When you label, you frequently need to center your text. The GRAPHIC PRINT statement works with the JUSTIFY statement to center and justify alphanumeric output horizontally and vertically.

The output string from the GRAPHICS PRINT statement is considered a block of occupied screen area. The JUSTIFY statement sets justification parameters for the X and Y axis of subsequent GRAPHIC PRINT statements. The block of screen area is realigned before display according to the current justification values.

Examples:

The following fragments are annotated extracts from DEMOGRAF's GRAPHIC PRINT Statement section. The first example prints with the lower left corner of the B of "BEGINS" at coordinates .5,.5, the exact center of the screen.

```
SET JUSTIFY 0,0
GRAPHIC PRINT AT (.5,.5): "BEGINS AT CENTER"
```

The next example prints horizontally centered text at coordinates .5,.3. The C of the word CENTERED is at .5,.3.

```
SET JUSTIFY .5,0
GRAPHIC PRINT AT (.5,.3): "THIS IS CENTERED"
```

The following example is vertically and horizontally centered. Notice that the lettering is shifted half a character down the Y axis.

```
SET JUSTIFY .5,.5
GRAPHIC PRINT AT (.5,.3): "THIS IS CENTERED"
```

In the following example, with X set to 1.0 by the JUSTIFY statement, the right side of the text is placed on the given coordinates, in this case at the center of the screen. This arrangement is useful for drawing graphs. You do not need to calculate string size to determine the beginning point of a label. For an example of graph-coordinate labels, see the JUSTIFY statement.

```
SET JUSTIFY 1.0,1.0
GRAPHIC PRINT AT (.5,.5):  "ENDS AT CENTER"
```

The example for the GRAPHIC PRINT statement in DEMOGRAF reads as follows:

```
CLEAR
SET WINDOW 0,1,0,1
SET CHARACTER HEIGHT 0
GRAPHIC PRINT AT (0,9):  "GRAPHIC PRINT STATEMENT"
SET JUSTIFY 0,0
GRAPHIC PRINT AT (.5,.5):  "BEGINS AT CENTER"
KEY% = PAUSE
REM  PAUSE IS A LOCALLY DEFINED FUNCTION
SET JUSTIFY .5,0
GRAPHIC PRINT AT (.5,.3):  "THIS IS CENTERED"
KEY% = PAUSE
SET JUSTIFY .5,.5
GRAPHIC PRINT AT (.5,.3):  "THIS IS CENTERED
SET JUSTIFY 1.0, 1.0
GRAPHIC PRINT AT (.5,.5):  "ENDS AT CENTER"
KEY% = PAUSE
```

JUSTIFY Statement

The JUSTIFY statement positions alphanumeric strings for the GRAPHIC PRINT statement or returns the current justification settings.

Syntax:

```
SET JUSTIFY <horizontal numeric expression>,
            <vertical numeric expression>
ASK JUSTIFY <horizontal real variable>, <vertical real variable>
```

Explanation:

The SET JUSTIFY statement defines the justified position of alphanumeric strings relative to their stated X, Y locations. The default justification of a string is (0,0), which puts the first character's lower left corner at the X, Y values of the GRAPHIC PRINT statement.

The <horizontal numeric expression> is a numeric expression representing the horizontal justification. The <vertical numeric expression> is a numeric expression representing the vertical justification.

The graphics system treats an alphanumeric string as if it occupies the smallest rectangle that can contain it. It gives this rectangle a coordinate system with (0,0) in the lower left corner and (1,1) in the top right corner. The graphics system displays the rectangle relative to the X, Y values of the GRAPHIC PRINT statement and the horizontal and vertical variables of the JUSTIFY statement. The horizontal and vertical expressions give the relative horizontal and vertical justification values for the rectangle of text.

The ASK JUSTIFY statement assigns the current values of horizontal and vertical justification parameters to their respective variables.

Example:

```
CLEAR
SET JUSTIFY (0,0)
SET WINDOW 0,100,0,100
SET CHARACTER HEIGHT 0
GRAPHIC PRINT AT (0,90):  "JUSTIFY STATEMENT"
PLOT (20,80),(20,20),(80,20)
PLOT (15,40),(20,40)
PLOT (15,60),(20,60)
PLOT (15,80),(20,80)
PLOT (40,15),(40,20)
PLOT (60,15),(60,20)
PLOT (80,15),(80,20)
SET JUSTIFY 1,.5
GRAPHIC PRINT AT (14,20): "20"
GRAPHIC PRINT AT (14,40): "40"
GRAPHIC PRINT AT (14,60): "60"
GRAPHIC PRINT AT (14,80): "80"
SET JUSTIFY .5,1
GRAPHIC PRINT AT (20,14): "20"
GRAPHIC PRINT AT (40,14): "40"
GRAPHIC PRINT AT (60,14): "60"
GRAPHIC PRINT AT (80,14): "80"
```

This routine prepares a graph window with tick marks, as illustrated in Figure 3-8.

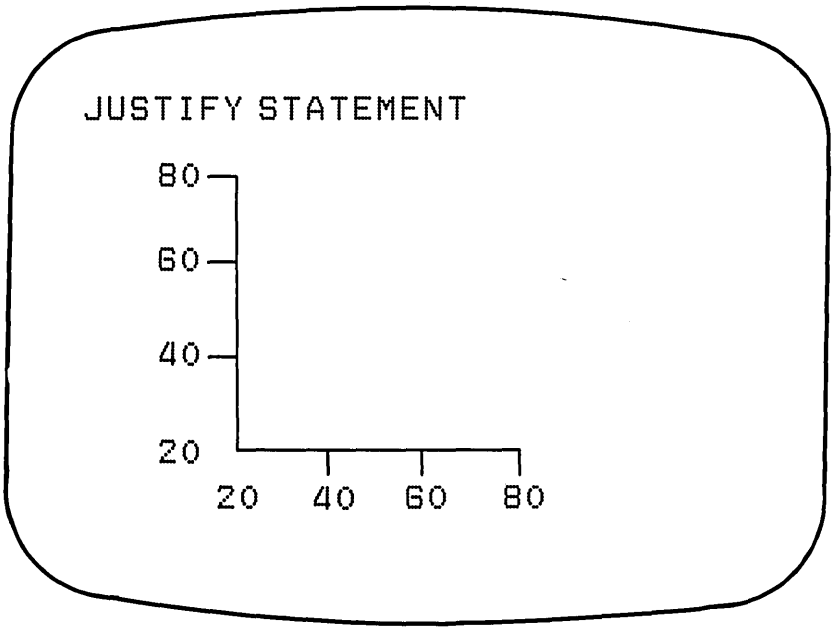

Figure 3-8. The JUSTIFY Statement

LINE STYLE Statement

The LINE STYLE statement specifies a line style for drawing or assigns the current line style value to a variable.

Syntax:

```
SET LINE STYLE <line style number>
ASK LINE STYLE <line style variable>
```

Explanation:

The SET LINE STYLE statement establishes the line style for subsequent PLOT or MAT PLOT statements as one of the n styles available on your device. The <line style number> is a numeric expression. You can find the the line styles available on your device with an ASK STYLE COUNT statement.

The following line styles are standard:

- 1 solid
- 2 dashed
- 3 dotted
- 4 dashed-dotted

The ASK LINE STYLE assigns the number of the current line style to the integer variable <line style variable>.

Example:

```
CLEAR
SET JUSTIFY 0,0
SET WINDOW 0,1,0,1
GRAPHIC PRINT AT (0,.9):  "LINE STYLE STATEMENT"
SET LINE STYLE 3
SET JUSTIFY 1,0
GRAPHIC PRINT AT (0.5,0.5) "Sign here"
PLOT (0.5,0.5),(0.8,0.5)
```

This example directs you to sign on a dotted line.

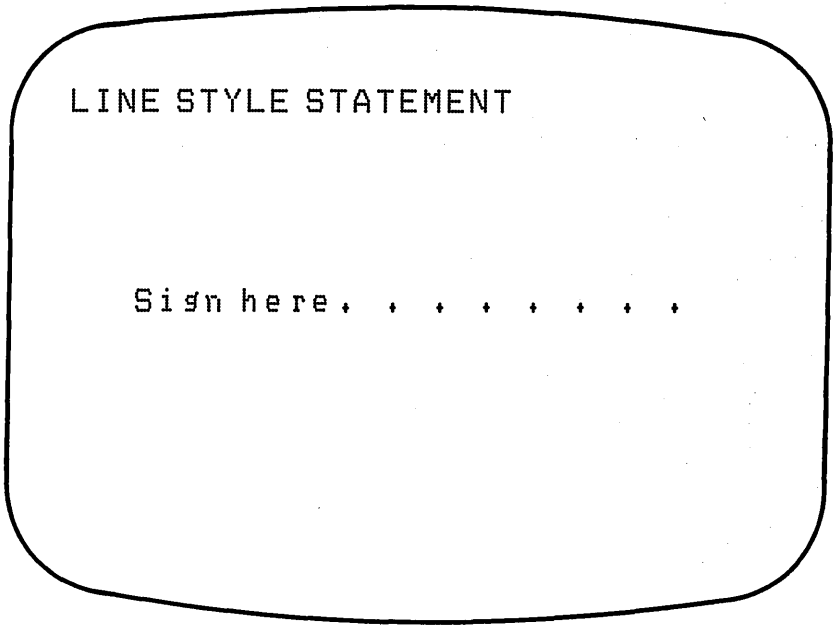

Figure 3-9. The LINE STYLE Statement

MARKER HEIGHT Statement

The MARKER HEIGHT statement sets the marker height or assigns the current value of the marker height to a variable.

Syntax:

```
SET MARKER HEIGHT <marker height>
ASK MARKER HEIGHT <marker variable>
```

Explanation:

Markers are special symbols you can use to highlight the demarkation points of lines and graphs. Markers are defined in the MARKER TYPE explanation, which describes the types of markers. The SET MARKER HEIGHT statement defines the height of markers relative to the present extent of the Y coordinate. The <marker height> is the height of the marker relative to the present extent of the Y coordinate. The initial extent of the Y coordinate is 0.0 to 1.0. A marker height of .1 results in markers that are 10% of the height of the display device.

You can change the range of the Y coordinate with the WINDOW statement. If you do change it, you might need to enter a new SET MARKER HEIGHT statement to remap the marker height to the new Y coordinate range.

To set the minimum possible marker height value within the window, use the statement,

```
SET MARKER HEIGHT 0
```

ASK MARKER HEIGHT assigns the current marker height value to the real variable <marker variable>. You can use the ASK MARKER HEIGHT statement after a SET MARKER HEIGHT to find the marker height assigned for the current character set.

Examples:

The following statement sets the marker height to 10% of the screen when the extent of the Y coordinate is 0 to 1.

```
SET MARKER HEIGHT .1
```

The following statements set the marker height to 15% (15/100) of the screen. The window statement sets the extent of Y axis to 0 to 100.

```
SET WINDOW 0,100,0,100
SET MARKER HEIGHT 15
```

The next statements return the minimum marker height in the variable MK.

```
SET MARKER HEIGHT 0
ASK MARKER HEIGHT MK
PRINT "MINIMUM MARKER HEIGHT IS = "; MK
```

The MARKER HEIGHT demonstration in DEMOGRAF is programmed as follows:

```
CLEAR
SET WINDOW 0,1,0,1
SET CHARACTER HEIGHT 0
SET LINE STYLE 1
SET JUSTIFY 0,0
GRAPHIC PRINT AT (0,.9):  "MARKER HEIGHT STATEMENT"
DIM MX(5)
DIM MY(5)
MX(0) = .3 : MY(0) = .7
MX(1) = .7 : MY(1) = .7
SET MARKER HEIGHT .1
MAT MARKER 1: MX,MY
SET WINDOW 0,100,0,100
MX(0) = 30 : MY(0) = 50
MX(1) = 70 : MY(1) = 50
SET MARKER HEIGHT 15
MAT MARKER 1:  MX,MY
SET MARKER HEIGHT 0
ASK MARKER HEIGHT MK
PRINT "MINIMUM MARKER HEIGHT IS = "; MK
```

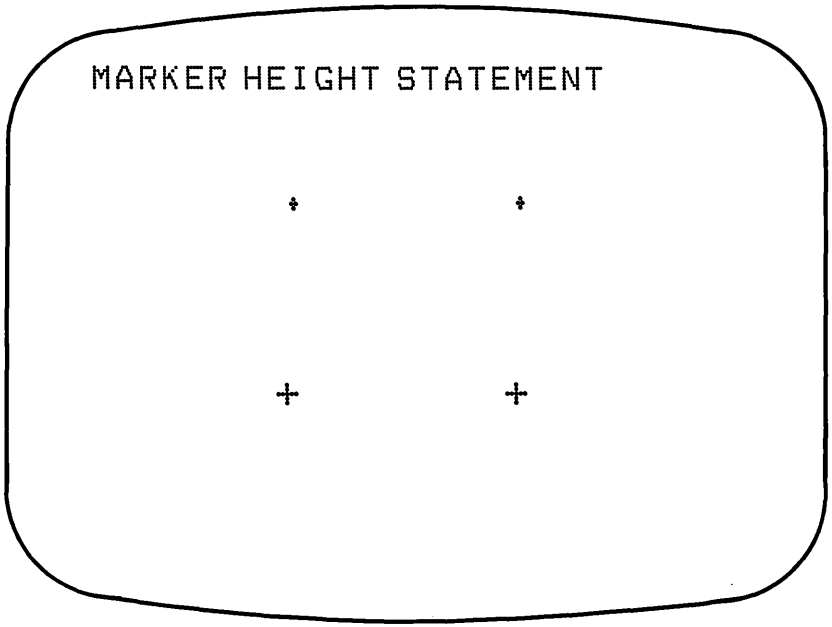

Figure 3-10. The MARKER HEIGHT Statement

MARKER TYPE Statement

The MARKER TYPE statement sets the marker type.

Syntax:

```
SET MARKER TYPE <integer expression>
```

Explanation:

The SET MARKER TYPE statement sets the marker type for subsequent MAT MARKER statements. There are at least five types of markers. The number of markers available to you depends on your implementation.

Table 3-2. Marker Types

TYPE	MARKER
1	.
2	+
3	*
4	O
5	X

Example:

```
CLEAR
SET WINDOW 0,1,0,1
SET MARKER HEIGHT 0
GRAPHIC PRINT AT (0,.9):  "MARKER TYPE STATEMENT"
MX(0) = .5 : MY(0) = .7
FOR I.INT% = 1 TO 5
    SET MARKER TYPE I.INT%
    MAT MARKER 0:  MX,MY
    MY(0) = MY(0) - .1
NEXT I.INT%
```

This routine displays the five standard marker types, from the top of the screen down. The SET MARKER HEIGHT 0 statement sets the marker height to the minimum available size.

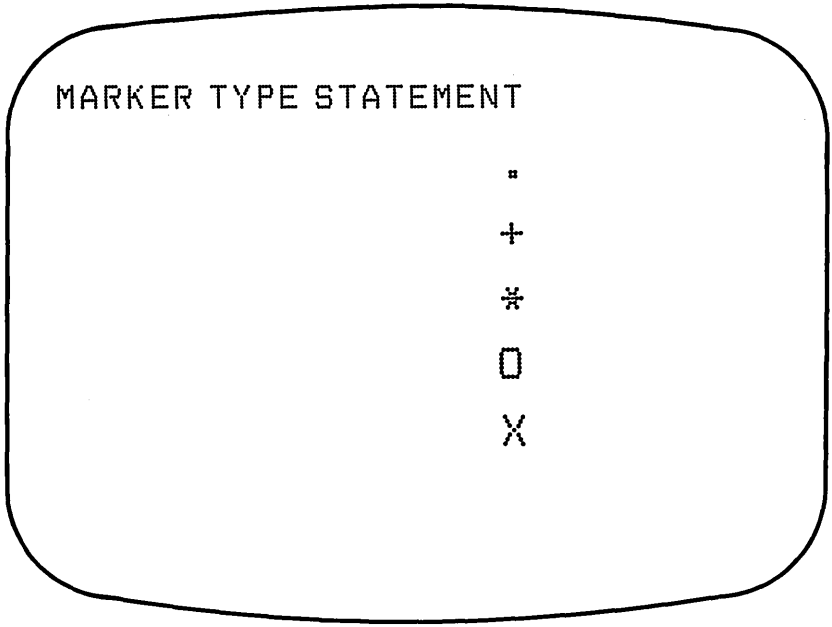

Figure 3-11. The MARKER TYPE Statement

MAT FILL Statement

The MAT FILL statement draws a filled polygon from arrays of X and Y coordinates.

Syntax:

MAT FILL <maximum array subscript>: <X array,Y array>

Explanation:

A MAT FILL statement draws a filled polygon that you define by the coordinate pairs in <X array,Y array>. The edges of the polygon are the line segments defined by the sequential points in <X array,Y array>. A closing line segment is assumed from the last point to the first. If the graphics device supports fill, the interior and edges are filled with the current color or pattern.

The <maximum array subscript> is a numeric expression defining the maximum array subscript. The MAT FILL statement begins taking coordinate pairs from element zero (0) of the X and Y arrays. The statement terminates after the coordinate pairs at <maximum array subscript>.

Note that the maximum array size is 72 elements. The maximum value for <maximum array subscript> is 71. You can draw figures requiring more points by using additional arrays.

Example:

```
CLEAR
SET LINE STYLE 1
SET JUSTIFY 0,0
GRAPHIC PRINT AT (0,90):  "MAT FILL STATEMENT"
SET WINDOW 0,100,0,100
SET CHARACTER HEIGHT 0
SET COLOR 1
DIM X.ARRAY(10)
DIM Y.ARRAY(10)
X.ARRAY(0) = 40 : Y.ARRAY(0) = 10
X.ARRAY(1) = 35 : Y.ARRAY(1) = 25
X.ARRAY(2) = 50 : Y.ARRAY(2) = 40
X.ARRAY(3) = 65 : Y.ARRAY(3) = 25
X.ARRAY(4) = 60 : Y.ARRAY(4) = 10
MAT FILL 4: X.ARRAY,Y.ARRAY
```

This routine draws a pentagon filled with color 1. Note that the MAT FILL statement automatically fills the line segment from 60,10 to 40,10.

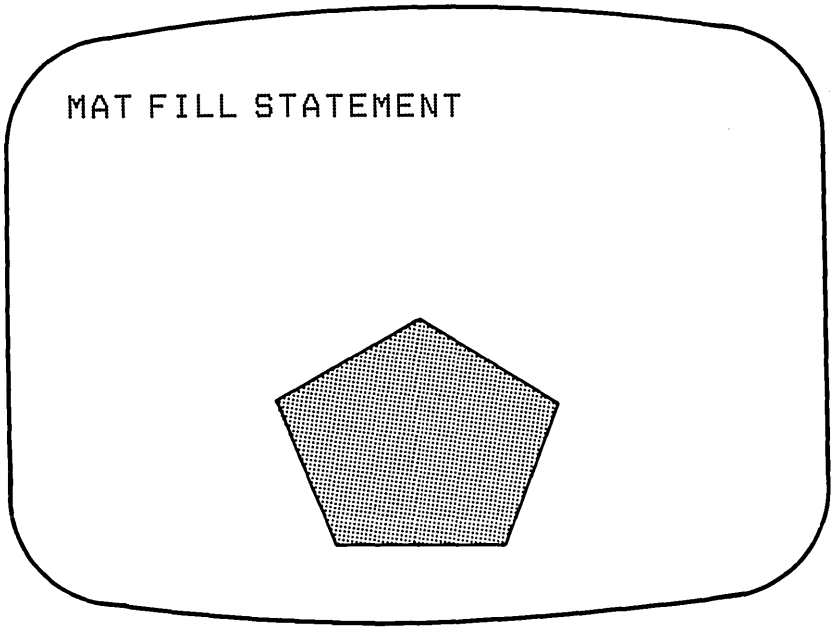

Figure 3-12. The MAT FILL Statement

MAT MARKER Statement

The MAT MARKER statement plots markers at an array of X and Y coordinates.

Syntax:

MAT MARKER <maximum array subscript>:<X array,Y array>

Explanation:

The MAT MARKER statement places markers at successive coordinate pairs as defined in <X array,Y array>. Use the MARKER HEIGHT and MARKER TYPE statements to define the size and type of the markers.

The <maximum array subscript> is a numeric expression defining the maximum array subscript. The MAT MARKER statement begins taking coordinate pairs from element zero (0) of the X and Y arrays. The statement ends after displaying a marker at the <maximum array subscript> element of the X and Y arrays.

The maximum array size is 72 elements. The maximum value for <maximum array subscript> is 71. You can draw figures requiring more points by using additional arrays.

Example:

```
CLEAR
SET WINDOW 0,100,0,100
GRAPHIC PRINT AT (0,90):  "MAT MARKER STATEMENT"
SET MARKER TYPE 1
SET MARKER HEIGHT 0
SET COLOR 1
MAT MARKER 4:  X.ARRAY,Y.ARRAY
```

X.ARRAY and Y.ARRAY are defined in the MAT FILL statement. This routine places markers at coordinates (40,10), (35,25), (50,40), (65,25), and (60,10), the axes of a pentagon.

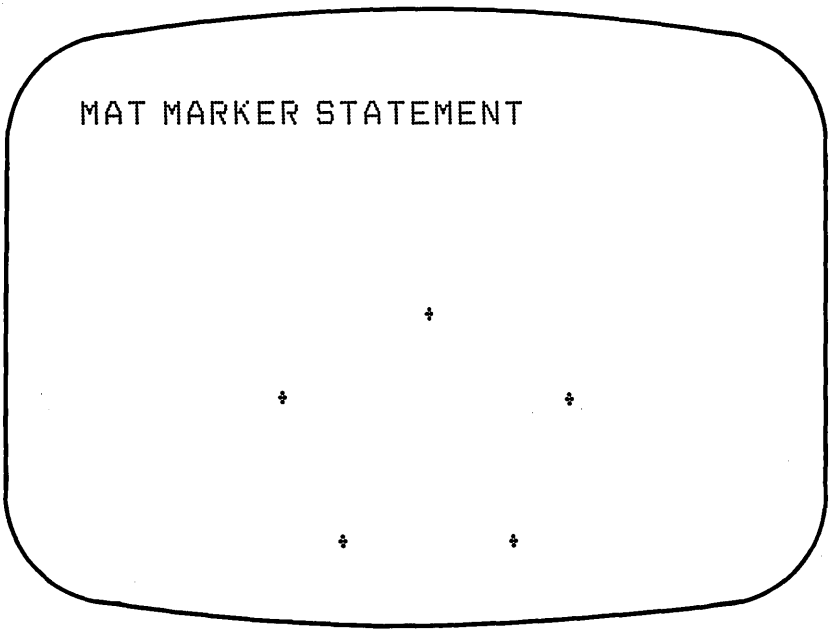

Figure 3-13. The MAT MARKER Statement

MAT PLOT Statement

The MAT PLOT statement connects points defined by arrays of X and Y coordinate pairs with lines.

Syntax:

MAT PLOT <maximum array subscript>: <X array,Y array>

Explanation:

The MAT PLOT statement plots a series of line segments from the <X array,Y array> list of coordinates. This statement is like the MAT FILL statement except the figure is not filled with color and the ending coordinates are not automatically connected to the beginning coordinates. The last element plotted is array(<maximum array subscript>).

Note that the maximum array size is 72 elements. The maximum value for <maximum array subscript> is 71. You can draw figures requiring more points by using additional arrays.

Example:

```
CLEAR
GRAPHIC PRINT AT (0,90):  "MAT PLOT STATEMENT"
SET COLOR 1
SET WINDOW 0,1,0,1
SET CHARACTER HEIGHT 0
FOR I.INT% = 0 TO 4
    X.ARRAY(I.INT%) = .01 * X.ARRAY(I.INT%)
    Y.ARRAY(I.INT%) = .01 * Y.ARRAY(I.INT%)
NEXT I.INT%
X.ARRAY(5) = .40 : Y.ARRAY(R) = .10
SET BEAM "OFF"
MAT PLOT 4: X.ARRAY, Y.ARRAY
KEY% = PAUSE
REM  PAUSE IS A LOCALLY DEFINED FUNCTION
CLEAR
MAT PLOT 5: X.ARRAY,Y.ARRAY
```

The first MAT PLOT statement draws a pentagon without a bottom. The second MAT PLOT draws the bottom because it uses the fifth element of both the X and Y arrays.

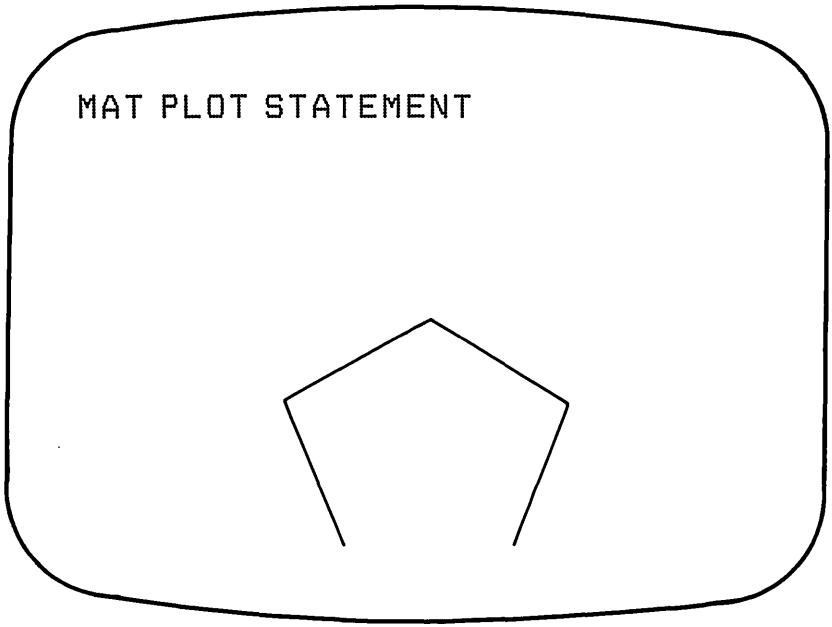

Figure 3-14. The MAT PLOT Statement

PLOT Statement

The PLOT statement connects a series of coordinate pairs with lines.

Syntax:

```
PLOT (x1,y1), (x2,y2), (x3,y3) ... [;]
```

Explanation:

The PLOT statement plots a series of line segments from the first coordinate pair through succeeding pairs. If the list ends with a semicolon, the beam stays on after the last point. Without the semicolon, the beam turns off.

If the beam is on before execution, the graphics system draws a line to the first point. If the beam is off when execution begins, drawing begins at the first point.

x1 represents an x coordinate as a numeric expression. y1 represents a y coordinate as a numeric expression.

You can use the PLOT statement to draw lines and figures when you want greater control of the drawing than MAT PLOT provides, or if using arrays for coordinate points is inconvenient.

Example:

```
CLEAR
SET WINDOW 0,100,0,100
SET CHARACTER HEIGHT 0
SET COLOR 1
GRAPHIC PRINT AT (0,90):  "PLOT STATEMENT"
PLOT (40,10),(35,25);
SET COLOR 2
PLOT (35,25),(50,40);
SET LINE STYLE 2
PLOT (50,40),(65,25);
SET LINE STYLE 1
SET COLOR 3
PLOT (65,25),(60,10),(40,10)
```

This routine draws a pentagon with different colors and line styles.

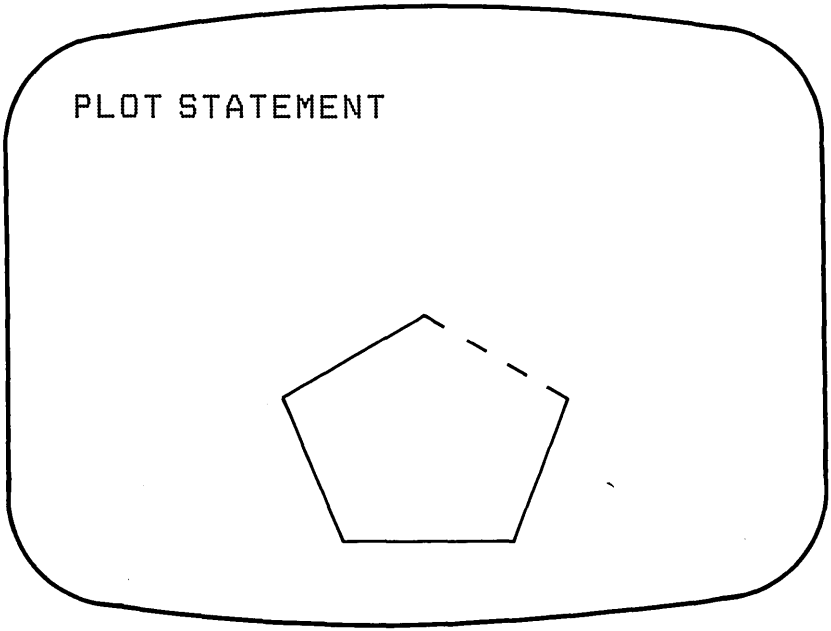

Figure 3-15. The PLOT Statement

POSITION Statement

The POSITION statement sets the beam position or assigns the value of the beam position to a variable.

Syntax:

```
SET POSITION <X coordinate>, <Y coordinate>
ASK POSITION <X coordinate variable>, <Y coordinate variable>
```

Explanation:

The SET POSITION statement positions the beam at the horizontal and vertical coordinates specified by <X coordinate> and <Y coordinate>.

The ASK POSITION statement assigns the current coordinates of the beam position to the <X coordinate variable> and <Y coordinate variable>.

If the beam is on when a SET POSITION statement executes, a line segment is drawn from the current position of the beam to the new beam location.

Example:

```
CLEAR
SET WINDOW 0,100,0,100
GRAPHIC PRINT AT (0,90):  "POSITION STATEMENT"
SET BEAM "OFF"
SET POSITION 50,50
SET POSITION 50,100
SET BEAM "ON"
SET POSITION 0,0
SET POSITION 50,50
```

This routine draws two line segments. The two SET POSITION statements with the beam off do not draw a line.

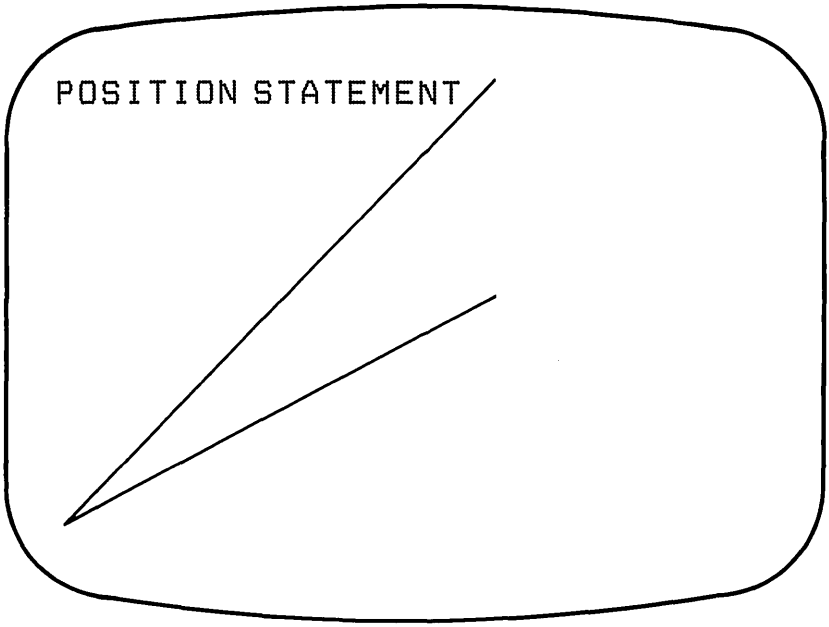

Figure 3-16. The POSITION Statement

STYLE COUNT Statement

The STYLE COUNT statement assigns the number of available line styles to a variable.

Syntax:

ASK STYLE COUNT <integer variable>

Explanation:

ASK STYLE COUNT assigns the number of line styles available on your device to an integer variable.

Example:

```
CLEAR
GRAPHIC PRINT AT (0,90):  "STYLE COUNT STATEMENT"
SET WINDOW 0,100,0,100
SET CHARACTER HEIGHT 0
ASK STYLE COUNT ST%
PRINT "THE NUMBER OF LINE STYLES IS: ",ST%
FOR I.INT% = 1 TO ST%
    SET LINE STYLE I.INT%
    SET BEAM "OFF"
    PLOT (10*I.INT%,10),(10*I.INT%,90)
NEXT I.INT%
KEY% = PAUSE
REM  PAUSE IS A LOCALLY DEFINED FUNCTION
SET LINE STYLE 1
```

This routine draws lines in each of the available line styles.

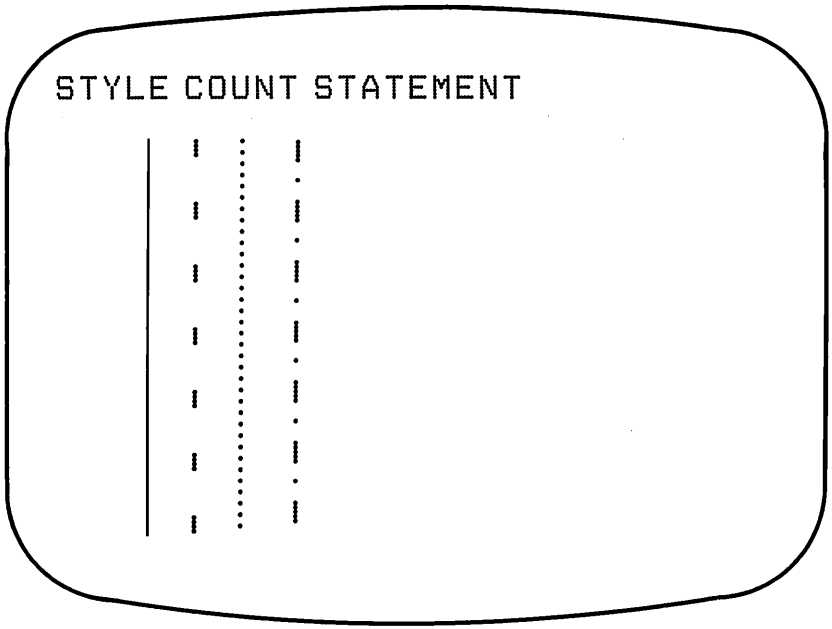

Figure 3-17. The STYLE COUNT Statement

TEXT ANGLE Statement

The TEXT ANGLE statement sets the number of degrees off horizontal at which text prints.

Syntax:

```
SET TEXT ANGLE <angle in radians>
ASK TEXT ANGLE <angle variable>
```

Explanation:

The SET TEXT ANGLE statement defines the approximate angle at which character strings print in GRAPHIC PRINT statements. The numeric expression <angle in radians> gives the angle in radians. The radians are measured counterclockwise, starting at a horizontal line across the screen and a vertical line through the center of the string. The intersection of these lines is the hub from which the text angle radiates.

The ASK TEXT ANGLE statement assigns the current text angle to the real variable <angle variable>.

To convert degrees to radians, you can use one of the following formulas:

- degrees * 2pi/360 = radians
- degrees / (360/2pi) = radians

The approximate value of pi is 3.1415926.

The approximate values in radians for some often-needed angles are shown in Table 3-3:

Table 3-3. Degrees-to-Radians Conversion Chart

Degrees	Approximate Radians
30	.52
45	.78
60	1.04
90	1.57
135	2.35
180	3.14

Example:

```
CLEAR
GRAPHIC PRINT AT (0,90): "TEXT ANGLE STATEMENT"
SET WINDOW 0,1,0,1
SET CHARACTER HEIGHT 0
PI = 3.1415926
RAD = PI*2
DEG = RAD/360
      FOR I.INT% = 90 TO 360 STEP 90
      SET TEXT ANGLE I.INT%*DEG
      GRAPHIC PRINT AT (.5,.5): "ROTATE ME"
NEXT I.INT%
KEY% = PAUSE
REM  PAUSE IS A LOCALLY DEFINED FUNCTION
SEE TEXT ANGLE 0
```

This routine prints ROTATE ME at 90-degree intervals around the center of the screen.

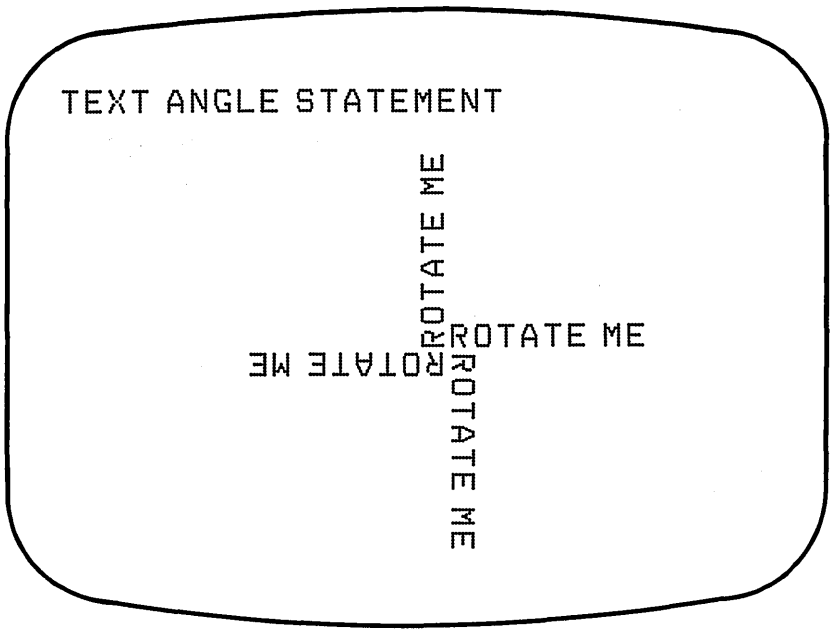

Figure 3-18. The TEXT ANGLE Statement

VIEWPORT Statement

The VIEWPORT statement establishes viewport boundaries or assigns the values of the current boundaries to variables.

Syntax:

```
SET VIEWPORT <X left bounds>,<X right bounds>,
             <Y lower bounds>,<Y upper bounds>
ASK VIEWPORT <X left bounds variable>,
             <X right bounds variable>,
             <Y lower bounds variable>,
             <Y upper bounds variable>
```

Explanation:

The SET VIEWPORT statement sets the boundaries of the viewport. Specify the boundaries as coordinates from 0.0 to 1.0, giving the right, left, upper, and lower boundaries as shown in the syntax.

The SET VIEWPORT statement establishes the area of the display device that can be used within the device boundaries set by the SET BOUNDS statement.

There is a hierarchy for the display area. First, the SET BOUNDS statement defines the available physical area of the display device. Second, the SET VIEWPORT statement defines the area that is used within those bounds. Third, the SET WINDOW statement defines the scale of the X and Y axes of the viewport.

The SET VIEWPORT and SET BOUNDS statements always use coordinate values from 0.0 to 1.0. The SET WINDOW statement defines the scale for subsequent statements that use X and Y coordinate values to define position or size. PLOT, MAT PLOT, GRAPHIC PRINT, POSITION, CHARACTER HEIGHT, and other input/output statements use the window's scale to translate the X and Y coordinate values into the viewport coordinate system.

Changing the viewport does not affect data currently displayed on the screen. You can use this command to switch from section to section on the screen. This lets you display multiple functions without worrying about overlap.

The ASK VIEWPORT statement assigns the current viewport boundaries, specified in coordinates ranging from 0.0 to 1.0 to real variables representing the left, right, lower, and upper bounds of the viewport.

Example:

```
CLEAR
GRAPHIC PRINT AT (0,.9):  "VIEWPORT STATEMENT"
X.ARRAY(0) = 0   : Y.ARRAY(0) = 0
X.ARRAY(1) = 0   : Y.ARRAY(1) = 100
X.ARRAY(2) = 100 : Y.ARRAY(2) = 100
X.ARRAY(3) = 100 : Y.ARRAY(3) = 0
X.ARRAY(4) = 0   : Y.ARRAY(4) = 0
SET VIEWPORT 0,1,0,1
SET WINDOW 0,100,0,100
MAT PLOT 4: X.ARRAY,Y.ARRAY
SET VIEWPORT .1,.9,.1,.9
MAT PLOT 4: X.ARRAY,Y.ARRAY
SET VIEWPORT .2,.8,.2,.8
MAT PLOT 4: X.ARRAY,Y.ARRAY
SET VIEWPORT .3,.5,.3,.5
MAT PLOT 4: X.ARRAY,Y.ARRAY
SET VIEWPORT .5,.7,.5,.7
MAT PLOT 4: X.ARRAY,Y.ARRAY
```

This routine draws five different boxes from the same coordinate arrays. The SET VIEWPORT statements change the dimensions and locations of the boxes.

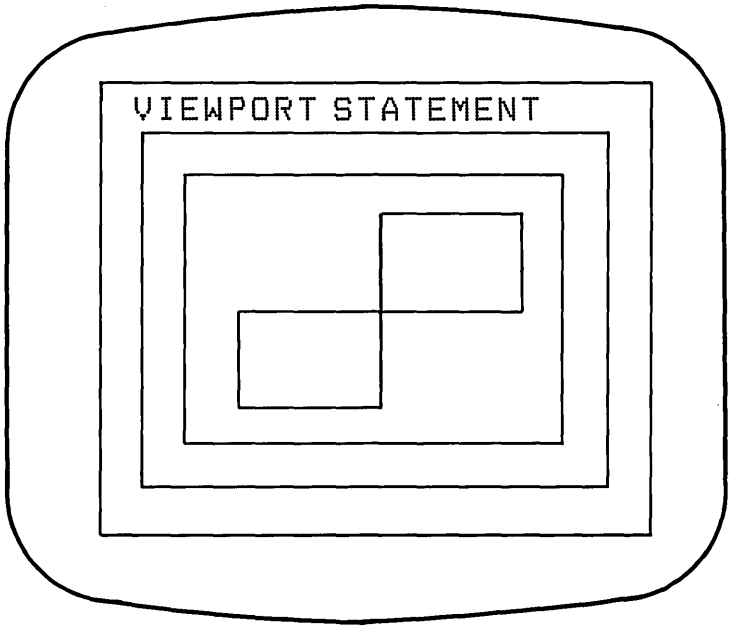

Figure 3-19. The VIEWPORT Statement

WINDOW Statement

The WINDOW statement establishes the scale of the X and Y axes within the viewport and bounds.

Syntax:

```
SET WINDOW <X left bounds>,<X right bounds>,
           <Y lower bounds>,<Y upper bounds>
ASK WINDOW <X left bounds variable>,<X right bounds variable>,
           <Y lower bounds variable>,<Y upper bounds variable>
```

Explanation:

A SET WINDOW statement establishes the range of the coordinate system. The statements PLOT, MAT PLOT, MAT FILL, GRAPHIC PRINT, CHARACTER HEIGHT, GRAPHIC INPUT, and POSITION operate within the X and Y coordinates established by SET WINDOW. Clipping also uses the scale set by the SET WINDOW statement to determine the usable area of the output device.

The window is set to 0,1,0,1 at initial program load. An ASK WINDOW statement assigns the current window extents to the real variables for the left, right, lower, and upper dimensions.

You can use the WINDOW statement to change the aspect ratio of a device. For example,

```
ASK DEVICE Y,X
SET WINDOW 0,Y,0,X
```

is one way to square a device.

Example:

```
CLEAR
SET VIEWPORT 0,1,0,1
SET WINDOW 0,100,0,100
SET CHARACTER HEIGHT 0
GRAPHIC PRINT AT (0,90):  "WINDOW STATEMENT"
PLOT (0,0),(60,60),(60,0),(0,0)
SET WINDOW 0,200,0,200
SET CHARACTER HEIGHT 0
PLOT (0,0),(60,60),(60,0),(0,0)
SET VIEWPORT .0,.5,.5,1.0
PLOT (0,0),(60,60),(60,0),(0,0)
```

This routine draws two triangles with the same coordinates and different window-scaling factors. The second figure nests within the first. A third triangle is plotted in a different viewport, again using the same coordinates.

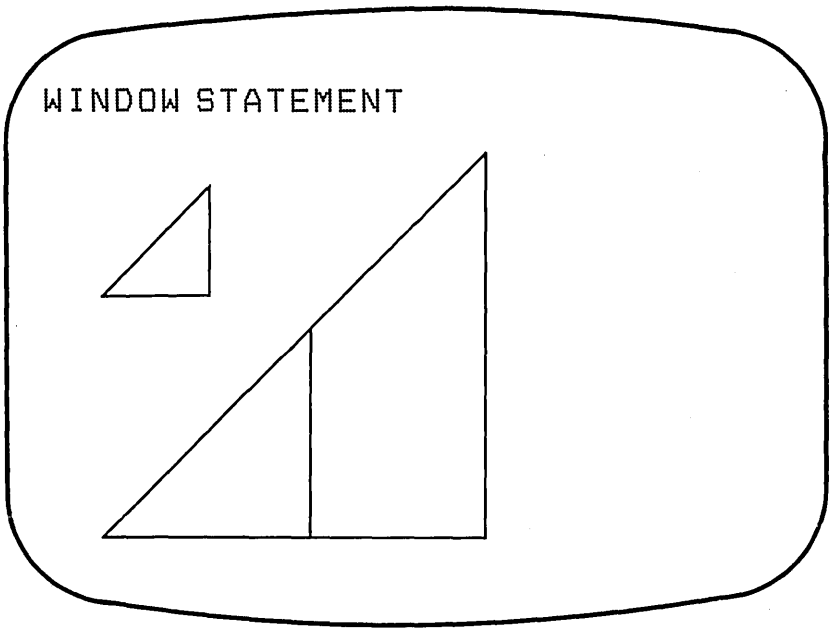

Figure 3-20. The WINDOW Statement

End of Section 3

Section 4
Sample Functions and Programs

The following listings contain samples of programming techniques for drawing circles, pie charts, and bar graphs.

The file CIRCOM.BAS includes circle drawing functions that you can include in your programs. TSTCIR.BAS is a test program for the circle drawing functions. GRAPHR.BAS is a demonstration program for pie charts and bar graphs.

4.1 CIRCOM.BAS

The following listing of CIRCOM.BAS includes remarks that describe the operation of the circle-drawing functions. Read the program listing to determine how to use the functions.

Listing 4-1. CIRCOM.BAS Program

```
REM         These are the circle drawing functions.
REM
REM         Use %INCLUDE CIRCOM.BAS to include the functions
REM         in your program.
REM
REM
REM         CALL BEG.CIR  to initialize the circle arrays
REM         CALL PLOT.CIR to draw a circle without fill
REM         CALL FILL.CIR to draw a circle with fill
REM
REM         The circle is centered at .5,.5 with a radius
REM         of .5 in a coordinate system ranging from 0 to 1
REM         on the X and Y axes.
REM
REM         Initialize the circle drawing arrays X.ARRAY and
REM         Y.ARRAY by using CALL BEG.CIR in the beginning
REM         of your program. You only need to execute this
REM         statement once. There is a long delay while
REM         this CALL is completed; you might want to put
REM         a message in your program to assure the user
REM         that the machine is computing.
REM
REM         Use the SET VIEWPORT and SET WINDOW statements
REM         to position the circle before calling the
REM         drawing functions.
REM
REM         You must adjust the aspect ratio of the device
REM         to proportion the circle. The following statements
REM         round the circle:
REM
```

Listing 4-1. (continued)

```
REM             ASK DEVICE X.AXIS,Y.AXIS
REM             SET WINDOW 0,X.AXIS/Y.AXIS,0,1
REM
REM     These statements scale the window so that the X
REM     and Y axes use the same unit scaling regardless
REM     of the aspect ratio of the device. The circle
REM     is drawn with a shift to the left of the viewport
REM     because of the decreased scaling of the X axis.
REM     You can also use the BOUNDS statement to square
REM     the device or the VIEWPORT statement to rescale
REM     the viewport.
REM
REM     The functions use the variables L.CIR,
REM     X.ARRAY, Y.ARRAY, and I.ANGLE. Variable X.ARRAY
REM     is an array of the X coordinates of the points
REM     around the circle. Y.ARRAY contains the corresponding Y
REM     coordinates. L.CIR contains a count of the
REM     number of coordinate pairs. I.ANGLE is only
REM     used by the FOR loop in BEG.CIR.
REM

DEF BEG.CIR
        DIM X.ARRAY(64)
        DIM Y.ARRAY(64)
        L.CIR=0

REM     THIS FOR LOOP STEPS THROUGH 0 TO 360 DEGREES
REM     USING RADIANS. THERE ARE 2PI RADIANS IN 360 DEGREES.
REM     THE FORMULA FOR CALCULATING THE POINTS AROUND
REM     A CIRCLE IS:
REM          X.COORDINATE = CENTER.POINT + RADIUS * COS(I.ANGLE)
REM          Y.COORDINATE = CENTER.POINT + RADIUS * SIN(I.ANGLE)

        FOR I.ANGLE = 0 TO 6.28 STEP .1
            X.ARRAY(L.CIR) = .5 + (.5 * COS(I.ANGLE))
            Y.ARRAY(L.CIR) = .5 + (.5 * SIN(I.ANGLE))
            L.CIR = L.CIR + 1
            NEXT I.ANGLE
REM     THE CIRCLE MUST BE CLOSED FOR MAT PLOT

        X.ARRAY(L.CIR) = X.ARRAY(0)
        Y.ARRAY(L.CIR) = Y.ARRAY(0)
        RETURN
```

Listing 4-1. (continued)

```
FEND

DEF PLOT.CIR
        MAT PLOT L.CIR: X.ARRAY,Y.ARRAY
        RETURN
FEND

DEF FILL.CIR
        MAT FILL L.CIR-1: X.ARRAY,Y.ARRAY
        RETURN
FEND
```

4.2 TSTCIR.BAS

The listing of TSTCIR.BAS includes remarks describing the program operation. After drawing each circle, the program waits for you to hit a key before it continues.

Listing 4-2. TSTCIR.BAS Program

```
REM     DEMONSTRATION PROGRAM FOR CIRCLE DRAWING FUNCTIONS
REM
REM     PROGRAM NAME: CIRTST.BAS
REM
        %INCLUDE GRAPHCOM.BAS
        %INCLUDE CIRCOM.BAS
        GRAPHIC OPEN 1
        CLEAR
        PRINT "COMPUTING"
        CALL BEG.CIR
        PRINT "ENDED"
        CALL PLOT.CIR
        KEY% = PAUSE            REM WAIT FOR KEYBOARD

REM     SCALE THE WINDOW TO DRAW A PROPERLY PROPORTIONED CIRCLE

        ASK DEVICE X.AXIS,Y.AXIS
        PRINT X.AXIS,Y.AXIS
        SET WINDOW 0,X.AXIS/Y.AXIS,0,1
        CALL PLOT.CIR
        KEY%=PAUSE
        CALL FILL.CIR
        KEY%=PAUSE
```

Listing 4-2. (continued)

```
REM     CHANGE THE VIEWPORT TO REPOSITION THE CIRCLE

        SET VIEWPORT 0,.5,0,.5  REM LOWER LEFT QUARTER
        CLEAR

        CALL PLOT.CIR
        KEY%=PAUSE

        SET VIEWPORT .5,1,0,.5  REM LOWER RIGHT QUARTER
        CALL PLOT.CIR
        KEY%=PAUSE

        SET VIEWPORT 0,.5,.5,1  REM UPPER LEFT QUARTER
        CALL PLOT.CIR
        KEY%=PAUSE

        SET VIEWPORT .5,1,.5,1  REM UPPER RIGHT QUARTER
        CALL PLOT.CIR
        KEY%=PAUSE

        STOP
        END
```

4.3 GRAPHR.BAS

GRAPHR.BAS draws a pie chart and a bar graph for up to nine items. The program includes remarks that guide you through the operation. You can use the DRAW.SLICE function to draw pie slices in your programs.

Listing 4-3. GRAPHR.BAS Program

```
REM     THIS IS A DEMONSTRATION PROGRAM FOR DRAWING
REM     PIE AND BAR CHARTS.
REM
REM     PROGRAM NAME: GRAPHR.BAS
REM
        %INCLUDE GRAPHCOM.BAS
        GRAPHIC OPEN 1
        CLEAR

        REM     If the device supports color fill, use MAT
        REM     FILL statements. Otherwise, use MAT PLOT
        REM     to draw figures.
```

Listing 4-3. (continued)

```
IN.FL:  INPUT "DOES THIS DEVICE SUPPORT COLOR FILL? Y/N: ";FILL.FLG$
        IF FILL.FLG$ = "Y" OR FILL.FLG$ = "N" THEN GOTO OK.FL
        PRINT "ENTER Y OR N, PLEASE"
        GOTO IN.FL
OK.FL:  PRINT "THANK YOU"

        REM     Initialize the arrays used for drawing the
        REM     slices in the pie chart. Two 100-element arrays
        REM     are constructed for drawing a full circle. Each
        REM     point in the arrays then represents one percent.

        PRINT "CALCULATING OCCURRING --- PLEASE WAIT"
        DIM X.ARRAY(100)
        DIM Y.ARRAY(100)
        DIM A.ARRAY(72)
        DIM B.ARRAY(72)
        A.ARRAY(0) = .5
        B.ARRAY(0) = .5
        L.CIR = 0
        FOR I.ANGLE = 0 TO 6.28-.0628 STEP .0628
                X.ARRAY(L.CIR) = .5 + (.5 * COS(I.ANGLE))
                Y.ARRAY(L.CIR) = .5 + (.5 * SIN(I.ANGLE))
                L.CIR = L.CIR + 1
                NEXT I.ANGLE

        REM     Close the circle

        X.ARRAY(L.CIR) = X.ARRAY(0)
        Y.ARRAY(L.CIR) = Y.ARRAY(0)
        GOTO START.IT

        REM     This function draws a slice beginning at the
        REM     point represented by BEG.PER and extending
        REM     through PER.CENT points. The color is set to
        REM     COL.OR and the ASCII.ID prints as an identifier
        REM     for the slice.

        REM     The function extracts the points from X.ARRAY
        REM     and Y.ARRAY and places them in A.ARRAY and
        REM     B.ARRAY. MAT FILL and MAT PLOT always begin
        REM     drawing at the first elements of the arrays, so
        REM     the slice must be extracted from the arrays.

        REM     The function makes provision for slices that
        REM     exceed 71 points. MAT FILL and MAT PLOT allow
        REM     a maximum element number of 72.
```

Listing 4-3. (continued)

```
DEF DRAW.SLICE (BEG.PER,PER.CENT,COL.OR,ASCII.ID)
        REAL BEG.PER,PER.CENT,COL.OR
        STRING ASCII.ID
        L.CIR = 1
        SET COLOR COL.OR
        OVR.FLOW = 0

        REM     Setup for slices greater than 71 percent.

        IF PER.CENT > 71 THEN SAVE.PER = 71:OVR.FLOW = 1\
                ELSE SAVE.PER = PER.CENT

        REM     Extract points from circle array.

BAK.UP: FOR CNT.ER = BEG.PER TO BEG.PER + SAVE.PER
                IN.DEX = CNT.ER
                IF CNT.ER > 100 THEN IN.DEX = CNT.ER - 100
                A.ARRAY(L.CIR) = X.ARRAY(IN.DEX)
                B.ARRAY(L.CIR) = Y.ARRAY(IN.DEX)
                L.CIR = L.CIR + 1
                NEXT CNT.ER

        REM     OVER.FLOW is 1 for slices over 71 percent.

        IF OVR.FLOW <> 1 THEN GOTO OVER.A

        REM     FILL.FLG$ is "N" for non-color-fill devices.

        IF FILL.FLG$ = "N" THEN MAT PLOT L.CIR-1: A.ARRAY,B.ARRAY\
                ELSE MAT FILL L.CIR-1: A.ARRAY,B.ARRAY
        OVR.FLOW = 0
        BEG.PER = BEG.PER + 71
        SAVE.PER = PER.CENT - 71
        IF FILL.FLG$ = "N" THEN L.CIR = 0 ELSE L.CIR = 1
        GOTO BAK.UP

OVER.A: A.ARRAY(0) = .5
        B.ARRAY(0) = .5

        REM     The slice must be closed for MAT PLOT. MAT FILL
        REM     closes automatically.

        IF FILL.FLG$ = "N" THEN\
                A.ARRAY(L.CIR) = .5:\
                B.ARRAY(L.CIR) = .5:\
                MAT PLOT L.CIR: A.ARRAY,B.ARRAY\
                ELSE\
                MAT FILL L.CIR-1: A.ARRAY,B.ARRAY
```

Listing 4-3. (continued)

```
        REM     Expand the viewport for printing the slice ID.
        REM     The minimum character height is used to adjust
        REM     the window so the slice ID will appear outside
        REM     the slice perimeter.

        SET VIEWPORT 1.0-Y.AXIS,1,0,1
        ADJ.IT = MIN.HGT/1.45
        SET WINDOW -ADJ.IT,1+ADJ.IT,-ADJ.IT,1+ADJ.IT

        REM     MID.PT is the center elements in the slice. This
        REM     is the position where the ID is printed.

        MID.PT = INT(BEG.PER+(PER.CENT/2))
        X.AXIS = X.ARRAY(MID.PT)
        Y.AXIS = Y.ARRAY(MID.PT)
        GRAPHIC PRINT AT (X.AXIS,Y.AXIS): ASCII.ID
        SET WINDOW 0,1,0,1
        RETURN
FEND

        REM     The first portion of the program lets you
        REM     enter up to 9 slices. Enter the item number (1-9)
        REM     and press the return key. Then type the slice
        REM     description (up to 6 characters), the dollar
        REM     value of the slice, and the color code for
        REM     the slice.

        REM     The following entries illustrate:

        REM     1 <return>
        REM     RENT,550,1 <return>
        REM     2 <return>
        REM     FOOD,450,2 <return>
        REM     3 <return>
        REM     CAR,225,3 <return>
        REM     4 <return>
        REM     OTHER,750,4 <return>

        REM     This sets up a graph of four items--rent of
        REM     $550 in color 1, food for $450 in color 2,
        REM     car for $225 in color 3, other for $750 in
        REM     color 4.

        REM     Terminate the input by typing 0 in response
        REM     to the prompt.
```

Listing 4-3. (continued)

```
        REM     After the 0 entry, the program calculates the
        REM     percentages and prints a listing of the entries.

        REM     You can change your entries by entering the
        REM     item number to change and typing in
        REM     the correct data.

START.IT: PRINT
        DIM ITM.DESC$(9)
        DIM ITM.VALUE(9)
        DIM ITM.COLOR(9)
        DIM ITM.PERC(9)
GO.A:   PRINT "ENTER AN ITEM NUMBER FROM 1 TO 9 TO ADD OR CHANGE"
        PRINT
        PRINT "THEN ENTER--DESCRIPTION,AMOUNT,COLOR,RETURN"
        PRINT
        PRINT "  DESCRIPTION IS THE SLICE DESCRIPTION"
        PRINT "  AMOUNT IS THE QUANTITY/AMOUNT OF THE SLICE"
        PRINT "  COLOR IS THE COLOR NUMBER TO USE FOR THE SLICE"
        PRINT "  RETURN MEANS TO PRESS THE RETURN KEY"
        PRINT
        PRINT "THE FIELDS ARE SEPARATED BY COMMAS"
        PRINT
IN.IT:  INPUT "ITEM NUMBER(0 TO FINISH): "; ITM.NUMBER%
        IF ITM.NUMBER% = 0 THEN GOTO PRT.EM
        IF ITM.NUMBER% > 0 AND ITM.NUMBER% < 10 THEN GOTO OKAY.IN
                PRINT "THE ITEM NUMBER MUST BE FROM 1 TO 9"
        GOTO IN.IT
OKAY.IN: IF ITM.VALUE(ITM.NUMBER%) = 0 THEN GOTO NEW.IN
        PRINT ITM.DESC$(ITM.NUMBER%),ITM.VALUE(ITM.NUMBER%),
        PRINT ITM.COLOR(ITM.NUMBER%)
NEW.IN: INPUT "DESC,AMOUNT,COLOR: ";DESC.IN$,VAL.IN,CLR.IN%
        ITM.DESC$(ITM.NUMBER%) = DESC.IN$
        ITM.VALUE(ITM.NUMBER%) = VAL.IN
        ITM.COLOR(ITM.NUMBER%) = CLR.IN%
        PRINT
        GOTO IN.IT
PRT.EM: TOT.VAL = 0

        REM     Calculate the total for percentages.

        FOR CNT.R = 1 TO 9
                TOT.VAL = TOT.VAL + ITM.VALUE(CNT.R)
                NEXT CNT.R
        PRINT

        REM     Print the item list with percentages.
```

Listing 4-3. (continued)

```
        FOR CNT.R = 1 TO 9
                IF ITM.VALUE(CNT.R) <> 0 THEN\
                        ITM.PERC(CNT.R) = ITM.VALUE(CNT.R)/TOT.VAL:\
                        ITM.PERC(CNT.R) = INT((100*ITM.PERC(CNT.R))+.5):\
                        PRINT CNT.R;"-";ITM.DESC$(CNT.R),ITM.VALUE(CNT.R),:\
                        PRINT ITM.COLOR(CNT.R);" ";ITM.PERC(CNT.R);"%"
                NEXT CNT.R
        PRINT:PRINT "TOTAL VALUE: ";TOT.VAL
        PRINT:INPUT "DRAW THE GRAPH? ";Y.N$
        IF Y.N$ <> "Y" THEN GOTO IN.IT
        CLEAR
        BEG.PER = 0

        REM     THE MINIMUM CHARACTER HEIGHT FOR THE DEVICE
        REM     IS USED TO ESTABLISH A BORDER AROUND THE CIRCLE
        REM     WHERE THE SLICE ID (THE ITEM NUMBER) CAN BE
        REM     PRINTED.

        SET CHARACTER HEIGHT 0
        ASK CHARACTER HEIGHT MIN.HGT
        MIN.HGT = 2 * MIN.HGT
        FOR CNT.R = 1 TO 9
                IF ITM.VALUE(CNT.R) = 0 THEN GOTO NXT.CNT

        REM     Determine the aspect ratio and square the device.
        REM     A border is left around the viewport for the
        REM     slice ID. The viewport is set to the right
        REM     of the device.

                ASK DEVICE X.AXIS,Y.AXIS
                SET VIEWPORT 1-Y.AXIS+MIN.HGT,1-MIN.HGT,MIN.HGT,1-MIN.HGT
                DESC.IN$ = ITM.DESC$(CNT.R)
                VAL.IN = ITM.VALUE(CNT.R)
                CLR.IN% = ITM.COLOR(CNT.R)
                PER.CENT = ITM.PERC(CNT.R)
                CALL DRAW.SLICE (BEG.PER,PER.CENT,CLR.IN%,STR$(CNT.R))
                BEG.PER = BEG.PER + PER.CENT
                SET VIEWPORT 0,1,0,1
                S.1$ = DESC.IN$+" "+STR$(PER.CENT)+"%"
                GRAPHIC PRINT AT (0,1-(CNT.R/10)):S.1$
NXT.CNT:        NEXT CNT.R

        REM     Is the graph filled? The percentage calculation
        REM     can be less than 100 percent due to roundoff.

        IF BEG.PER >= 100 THEN GOTO BAR.A
        PER.CENT = 100 - BEG.PER
        DESC.IN$ = " "
        ASK DEVICE X.AXIS,Y.AXIS
        SET VIEWPORT 1-Y.AXIS+MIN.HGT,1-MIN.HGT,MIN.HGT,1-MIN.HGT
        CALL DRAW.SLICE (BEG.PER,PER.CENT,CLR.IN%,DESC.IN$)
```

Listing 4-3. (continued)

```
        REM     This routine draws a simple bar chart of the
        REM     data. The window range is set to 1/3 greater
        REM     than the largest item in the array. This
        REM     makes the longest bar extend to
        REM     75% of the viewport.

BAR.A:  KEY%=PAUSE
        DIM BAR.X(4)
        DIM BAR.Y(4)
        SET VIEWPORT 0,1,0,1
        SET WINDOW 0,1,0,1
        SET CHARACTER HEIGHT 0
        ASK CHARACTER HEIGHT MIN.HGT
        CLEAR
        SET JUSTIFY .5,0
        SET COLOR 1
        GRAPHIC PRINT AT (.5,.99-MIN.HGT):"BAR CHART"
        SET JUSTIFY 0,0
        MAX.VAL = 0

        REM     Determine the maximum percentage.

        FOR CNT.R = 1 TO 9
                IF MAX.VAL < ITM.PERC(CNT.R) THEN\
                        MAX.VAL = ITM.PERC(CNT.R)
                NEXT CNT.R
        MAX.VAL = 1.33 * MAX.VAL

        REM     Scale the window. The X axis is 1/3 larger
        REM     than the largest item to be graphed.
        REM     The Y axis is scaled to 10 lines.

        SET WINDOW 0,MAX.VAL,0,10
        SET CHARACTER HEIGHT 0
        ASK CHARACTER HEIGHT MIN.HGT

        REM     Draw the items.

        FOR CNT.R = 1 TO 9
                IF ITM.VALUE(CNT.R) = 0 THEN GOTO NXT.A
                SET COLOR ITM.COLOR(CNT.R)
                P.LINE = 10 - CNT.R
                S.1$ = ITM.DESC$(CNT.R)+"-"+STR$(ITM.PERC(CNT.R))+"%"
                IF ITM.VALUE(CNT.R) <> ITM.PERC(CNT.R) THEN\
                        S.1$ = S.1$+" $"+STR$(ITM.VALUE(CNT.R))
                GRAPHIC PRINT AT (0,P.LINE): S.1$
```

Listing 4-3. (continued)

```
        REM     Set up the BAR.X and BAR.Y arrays to draw the
        REM     bar. MAX.VAL is the percentage for the item.
        REM     The window scaling automatically scales the
        REM     bar. No special calculations are required.

                MAX.VAL = ITM.PERC(CNT.R)
                TOP = P.LINE - .1
                BOT = TOP - .4
                BAR.Y(0) = BOT
                BAR.Y(1) = TOP
                BAR.X(2) = MAX.VAL
                BAR.Y(2) = TOP
                BAR.X(3) = MAX.VAL
                BAR.Y(3) = BOT
                BAR.Y(4) = BOT
                IF FILL.FLG$ = "N" THEN MAT PLOT 4: BAR.X,BAR.Y\
                        ELSE MAT FILL 3: BAR.X,BAR.Y
NXT.A:          NEXT CNT.R
                KEY% = PAUSE
                STOP
        END
```

End of Section 4

Appendix A
DEMOGRAF Program Listing

Listing A-1. DEMOGRAF Program

```
REM THIS IS A DEMONSTRATION PROGRAM FOR
REM CBASIC GRAPHICS EXTENSIONS
REM
REM PROGRAM NAME: DEMOGRAF
REM
        %INCLUDE GRAPHCOM.BAS

DEF PAUSE
REM UTILITY TO SUSPEND PROGRAM EXECUTION UNTIL CHARACTER IS
REM ENTERED AT CONSOLE, STOPPING PROGRAM IF CTRL-C IS ENTERED,
REM OTHERWISE RETURNING INTEGER VALUE OF CHARACTER ENTERED.
REM CHARACTER IS NOT DISPLAYED.

INTEGER PAUSE,CHOICE

        CHOICE = INKEY
        IF CHOICE = 3 THEN STOP
        PAUSE = CHOICE
FEND

        GRAPHIC OPEN 1
        CLEAR

BEEM:   GRAPHIC PRINT AT (0,.9): "BEAM STATEMENT"
        SET BEAM "OFF"
        PLOT (0,1),(1,1),(1,0),(0,0)
        KEY% = PAUSE            REM WAIT FOR CONSOLE INPUT
        CLEAR
        SET BEAM "ON"
        PLOT (0,1),(1,1),(1,0),(0,0)
        KEY% = PAUSE

REM ILLUSTRATE TECHNIQUE OF SQUARING A DISPLAY

BOWNDS: CLEAR
        GRAPHIC PRINT AT (0,.9): "BOUNDS STATEMENT"
        ASK DEVICE X.AXIS,Y.AXIS
        PRINT "THE ASPECT RATIO IS = ";Y.AXIS;"/";X.AXIS
        KEY% = PAUSE
```

Listing A-1. (continued)

```
        PLOT (0,0),(0,1),(1,1),(1,0),(0,0)
        KEY% = PAUSE
        CLEAR
        SET BOUNDS Y.AXIS,X.AXIS
        PLOT (0,0),(0,1),(1,1),(1,0),(0,0)
        SET BOUNDS 1,1
        KEY% = PAUSE

REM DEMONSTRATE CONTROL OF GRAPHIC CHARACTER HEIGHT
REM AND MINIMUM HEIGHT FOR GRAPHIC CHARACTERS

HIGH:   CLEAR
        SET CHARACTER HEIGHT 0
        GRAPHIC PRINT AT (0,.9): "CHARACTER HEIGHT STATEMENT"
        SET CHARACTER HEIGHT .1
        GRAPHIC PRINT AT (0,.7): "10 PERCENT"
        KEY% = PAUSE
        SET WINDOW 0,100,0,100
        SET CHARACTER HEIGHT 15
        GRAPHIC PRINT AT (0,40): "15 PERCENT"
        KEY% = PAUSE
        SET CHARACTER HEIGHT 0
        ASK CHARACTER HEIGHT CH
        PRINT "MINIMUM CHARACTER HEIGHT IS = "; CH
        GRAPHIC PRINT AT (0,20): "MINIMUM HEIGHT"

REM DISPLAY SEVERAL RANDOM LINES ON SCREEN, THEN MAKE
REM THEM DISAPPEAR VIA "CLEAR" STATEMENT

        INPUT ""; LINE SEED$
        RANDOMIZE
CLR:    CLEAR
        GRAPHIC PRINT AT (0,90): "CLEAR STATEMENT"
        SET WINDOW 0,1,0,1
        FOR I.INT% = 1 TO 10
                PLOT (RND,RND),(RND,RND)
        NEXT I.INT%
        KEY% = PAUSE
        CLEAR

REM ILLUSTRATE EFFECT OF AUTOMATIC CLIPPING WHEN FIGURE
REM EXCEEDS ALLOWABLE BOUNDARIES

CLP:    SET WINDOW 0,100,0,100
        GRAPHIC PRINT AT (0,90): "CLIP STATEMENT"
        PLOT (25,10),(50,150),(75,10),(25,10)
        KEY% = PAUSE

REM DRAW BORDER IN EACH AVAILABLE COLOR (NUMBER OF
REM COLORS VARIES WITH RESOLUTION)
```

Listing A-1. (continued)

```
COLR:   CLEAR
        GRAPHIC PRINT AT (0,90): \
                    "COLOR AND COLOR COUNT STATEMENTS"
        SET WINDOW 0,1,0,1
        ASK COLOR COUNT CT%
        FOR I.INT% = 1 TO CT%
                SET COLOR I.INT%
                PLOT (0,0),(0,1),(1,1),(1,0),(0,0)
                KEY% = PAUSE
        NEXT I.INT%

REM RETRIEVE AND DISPLAY SPECIFICATIONS FOR CURRENT DEVICE

DEVC:   CLEAR
        SET CHARACTER HEIGHT 0
        SET COLOR 1
        GRAPHIC PRINT AT (0,.8): "DEVICE STATEMENT"
        ASK DEVICE X.AXIS,Y.AXIS
        PRINT "THE VERTICAL AXIS IS "; \
              Y.AXIS*100/X.AXIS;"PERCENT OF THE";
        PRINT " HORIZONTAL AXIS"
        PRINT "X= ";X.AXIS;"  Y= ";Y.AXIS
        KEY% = PAUSE

REM MENTION "GRAPHIC CLOSE" STATEMENT

GCLOSE: CLEAR
        GRAPHIC PRINT AT (0,.9): "GRAPHIC CLOSE STATEMENT"
        GRAPHIC PRINT AT (0,.5): "GRAPHIC CLOSE HAS NO DEMO"
        KEY% = PAUSE

REM ILLUSTRATE GRAPHIC INPUT VIA CURSOR POSITIONING

GIN:    CLEAR
        SET WINDOW 0,100,0,100
        SET CHARACTER HEIGHT 0
        GRAPHIC PRINT AT (0,80): "GRAPHIC INPUT STATEMENT"
        GRAPHIC PRINT AT (0,25): "OPTION 1 ."
        SET COLOR 2
        GRAPHIC PRINT AT (0,50): "OPTION 2 ."
        SET COLOR 3
        GRAPHIC PRINT AT (0,75): "OPTION 3 ."
        GRAPHIC INPUT X.AXIS,Y.AXIS,A$
        N = INT((Y.AXIS+5)/25)
```

Listing A-1. (continued)

```
        IF N = 0 THEN N = 1      REM NO OPTION ZERO
        IF N > 3 THEN N = 3      REM ONLY THREE OPTIONS
        PRINT "THE CURSOR WAS POSITIONED AT: "; X.AXIS,Y.AXIS
        PRINT "YOU SELECTED OPTION:          "; N
        PRINT "THE TERMINATING KEY WAS:      "; A$
        KEY% = PAUSE

REM MENTION "GRAPHIC OPEN" STATEMENT

GOPEN:  CLEAR
        SET COLOR 1
        SET CHARACTER HEIGHT 0
        GRAPHIC PRINT AT (0,90): "GRAPHIC OPEN STATEMENT"
        GRAPHIC PRINT AT (0,50): \
                    "THE GRAPHIC OPEN HAS NO DEMONSTRATION"
        KEY% = PAUSE

REM DEMONSTRATE CENTERING AND JUSTIFICATION

GPRT:   CLEAR
        SET WINDOW 0,1,0,1
        SET CHARACTER HEIGHT 0
        GRAPHIC PRINT AT (0,.9): "GRAPHIC PRINT STATEMENT"
        SET JUSTIFY 0,0
        GRAPHIC PRINT AT (.5,.5): "BEGINS AT CENTER"
        KEY% = PAUSE
        SET JUSTIFY .5,0
        GRAPHIC PRINT AT (.5,.3): "THIS IS CENTERED"
        KEY% = PAUSE
        SET JUSTIFY .5,.5
        GRAPHIC PRINT AT (.5,.3): "THIS IS CENTERED"
        KEY% = PAUSE
        SET JUSTIFY 1.0,1.0
        GRAPHIC PRINT AT (.5,.5): "ENDS AT CENTER"
        KEY% = PAUSE
JUST:   CLEAR
        SET JUSTIFY 0,0
        SET WINDOW 0,100,0,100
        SET CHARACTER HEIGHT 0
        GRAPHIC PRINT AT (0,90): "JUSTIFY STATEMENT"
        PLOT (20,80),(20,20),(80,20)
        PLOT (15,40),(20,40)
        PLOT (15,60),(20,60)
        PLOT (15,80),(20,80)
        PLOT (40,15),(40,20)
        PLOT (60,15),(60,20)
        PLOT (80,15),(80,20)
        SET JUSTIFY 1,.5
        GRAPHIC PRINT AT (14,20): "20"
        GRAPHIC PRINT AT (14,40): "40"
        GRAPHIC PRINT AT (14,60): "60"
        GRAPHIC PRINT AT (14,80): "80"
```

Listing A-1. (continued)

```
        SET JUSTIFY .5,1
        GRAPHIC PRINT AT (20,14): "20"
        GRAPHIC PRINT AT (40,14): "40"
        GRAPHIC PRINT AT (60,14): "60"
        GRAPHIC PRINT AT (80,14): "80"
        KEY% = PAUSE

REM EXHIBIT VARIATION OF LINE STYLE

STYL:   CLEAR
        SET JUSTIFY 0,0
        SET WINDOW 0,1,0,1
        GRAPHIC PRINT AT (0,.9): "LINE STYLE STATEMENT"
        SET LINE STYLE 3
        SET JUSTIFY 1,0
        GRAPHIC PRINT AT (.5,.5): "Sign here"
        PLOT (0.5,0.5),(0.8,0.5)
        KEY% = PAUSE

REM ILLUSTRATE VARIATION IN SIZE OF MARKERS

MHIGH:  CLEAR
        SET WINDOW 0,1,0,1
        SET CHARACTER HEIGHT 0
        SET LINE STYLE 1
        SET JUSTIFY 0,0
        GRAPHIC PRINT AT (0,.9): "MARKER HEIGHT STATEMENT"
        DIM MX(5)
        DIM MY(5)
        MX(0) = .3 : MY(0) = .7
        MX(1) = .7 : MY(1) = .7
        SET MARKER HEIGHT .1
        MAT MARKER 1: MX,MY
        SET WINDOW 0,100,0,100
        MX(0) = 30 : MY(0) = 50
        MX(1) = 70 : MY(1) = 50
        SET MARKER HEIGHT 15
        MAT MARKER 1: MX,MY
        SET MARKER HEIGHT 0
        ASK MARKER HEIGHT MK
        PRINT "MINIMUM MARKER HEIGHT IS = "; MK
        KEY% = PAUSE

REM DEMONSTRATE ALL MARKER SHAPES

MTYPE:  CLEAR
        SET WINDOW 0,1,0,1
        SET MARKER HEIGHT 0
        GRAPHIC PRINT AT (0,.9): "MARKER TYPE STATEMENT"
        MX(0) = .5 : MY(0) = .7
```

Listing A-1. (continued)

```
        FOR I.INT% = 1 TO 5
                SET MARKER TYPE I.INT%
                MAT MARKER 0: MX,MY
                MY(0) = MY(0) - .1
        NEXT I.INT%
        KEY% = PAUSE

REM DEMONSTRATE FILLED POLYGON

MFILL:  CLEAR
        SET LINE STYLE 1
        SET JUSTIFY 0,0
        GRAPHIC PRINT AT (0,.9): "MAT FILL STATEMENT"
        SET WINDOW 0,100,0,100
        SET CHARACTER HEIGHT 0
        SET COLOR 1
        DIM X.ARRAY(10)
        DIM Y.ARRAY(10)
        X.ARRAY(0) = 40 : Y.ARRAY(0) = 10
        X.ARRAY(1) = 35 : Y.ARRAY(1) = 25
        X.ARRAY(2) = 50 : Y.ARRAY(2) = 40
        X.ARRAY(3) = 65 : Y.ARRAY(3) = 25
        X.ARRAY(4) = 60 : Y.ARRAY(4) = 10
        MAT FILL 4: X.ARRAY,Y.ARRAY
        KEY% = PAUSE

REM ILLUSTRATE POSITIONING OF MARKERS VIA AN ARRAY

MMARK:  CLEAR
        SET WINDOW 0,100,0,100
        GRAPHIC PRINT AT (0,90): "MAT MARKER STATEMENT"
        SET MARKER HEIGHT 0
        SET MARKER TYPE 1
        SET COLOR 1
        MAT MARKER 4: X.ARRAY,Y.ARRAY
        KEY% = PAUSE

REM DEMONSTRATE DRAWING POLYGON OUTLINE VIA AN ARRAY

MPLOT:  CLEAR
        GRAPHIC PRINT AT (0,90): "MAT PLOT STATEMENT"
        SET COLOR 1
        SET WINDOW 0,1,0,1
        SET CHARACTER HEIGHT 0
        FOR I.INT% = 0 TO 4
                X.ARRAY(I.INT%) = .01 * X.ARRAY(I.INT%)
                Y.ARRAY(I.INT%) = .01 * Y.ARRAY(I.INT%)
        NEXT I.INT%
        X.ARRAY(5) = .40 : Y.ARRAY(5) = .10
        SET BEAM "OFF"
        MAT PLOT 4: X.ARRAY,Y.ARRAY
        KEY% = PAUSE
```

Listing A-1. (continued)

```
        CLEAR
        MAT PLOT 5: X.ARRAY,Y.ARRAY
        KEY% = PAUSE

REM DO POLYGON VIA "PLOT" STATEMENTS

PLT:    CLEAR
        SET WINDOW 0,100,0,100
        SET CHARACTER HEIGHT 0
        SET COLOR 1
        GRAPHIC PRINT AT (0,90): "PLOT STATEMENT"
        PLOT (40,10),(35,25);
        SET COLOR 2
        PLOT (35,25),(50,40);
        SET LINE STYLE 2
        PLOT (50,40),(65,25);
        SET LINE STYLE 1
        SET COLOR 3
        PLOT (65,25),(60,10),(40,10)
        KEY% = PAUSE

REM EXERCISE ARBITRARY POSITIONING OF GRAPHIC BEAM

POSIT:  CLEAR
        GRAPHIC PRINT AT (0,90): "POSITION STATEMENT"
        SET BEAM "OFF"
        SET POSITION 50,50
        SET POSITION 50,100
        SET BEAM "ON"
        SET POSITION 0,0
        SET POSITION 50,50
        KEY% = PAUSE

REM SHOW ALL LINE STYLES

STCNT:  CLEAR
        GRAPHIC PRINT AT (0,90): "STYLE COUNT STATEMENT"
        SET WINDOW 0,100,0,100
        SET CHARACTER HEIGHT 0
        ASK STYLE COUNT ST%
        PRINT "THE NUMBER OF LINE STYLES IS: "; ST%
        FOR I.INT% = 1 TO ST%
                SET LINE STYLE I.INT%
                SET BEAM "OFF"
                PLOT (10*I.INT%,10),(10*I.INT%,90)
        NEXT I.INT%
        KEY% = PAUSE
        SET LINE STYLE 1
```

Listing A-1. (continued)

```
REM DEMONSTRATE ROTATION OF TEXT

ANGEL:  CLEAR
        GRAPHIC PRINT AT (0,90): "TEXT ANGLE STATEMENT"
        SET WINDOW 0,1,0,1
        SET CHARACTER HEIGHT 0
        PI = 3.1415926
        RAD = PI*2
        DEG = RAD/360
                FOR I.INT% = 90 TO 360 STEP 90
                SET TEXT ANGLE I.INT%*DEG
                GRAPHIC PRINT AT (.5,.5): "ROTATE ME"
        NEXT I.INT%
        KEY% = PAUSE
        SET TEXT ANGLE 0

REM ILLUSTRATE EFFECT OF VARYING VIEWPORT

VYOU:   CLEAR
        GRAPHIC PRINT AT (0,.9): "VIEWPORT STATEMENT"
        X.ARRAY(0) = 0   : Y.ARRAY(0) = 0
        X.ARRAY(1) = 0   : Y.ARRAY(1) = 100
        X.ARRAY(2) = 100 : Y.ARRAY(2) = 100
        X.ARRAY(3) = 100 : Y.ARRAY(3) = 0
        X.ARRAY(4) = 0   : Y.ARRAY(4) = 0
        SET VIEWPORT 0,1,0,1
        SET WINDOW 0,100,0,100
        SET CHARACTER HEIGHT 0
        MAT PLOT 4: X.ARRAY,Y.ARRAY
        SET VIEWPORT .1,.9,.1,.9
        MAT PLOT 4: X.ARRAY,Y.ARRAY
        SET VIEWPORT .2,.8,.2,.8
        MAT PLOT 4: X.ARRAY,Y.ARRAY
        SET VIEWPORT .3,.5,.3,.5
        MAT PLOT 4: X.ARRAY,Y.ARRAY
        SET VIEWPORT .5,.7,.5,.7
        MAT PLOT 4: X.ARRAY,Y.ARRAY
        KEY% = PAUSE

REM ILLUSTRATE EFFECT OF VARYING WINDOW

WINDW:  CLEAR
        SET VIEWPORT 0,1,0,1
        SET WINDOW 0,100,0,100
        SET CHARACTER HEIGHT 0
        GRAPHIC PRINT AT (0,90): "WINDOW STATEMENT"
        PLOT (0,0),(60,60),(60,0),(0,0)
        SET WINDOW 0,200,0,200
        SET CHARACTER HEIGHT 0
        PLOT (0,0),(60,60),(60,0),(0,0)
```

Listing A-1. (continued)

```
        SET VIEWPORT 0,.5,.5,1.0
        PLOT (0,0),(60,60),(60,0),(0,0)
        KEY% = PAUSE

REM FINISH DEMONSTRATION AND END PROGRAM

FIN:    CLEAR
        SET WINDOW 0,100,0,100
        SET CHARACTER HEIGHT 0
        SET VIEWPORT 0,1,0,1
        SET COLOR 1
        FOR I.INT% = 1 TO CT%
                SET COLOR I.INT%
                MAT PLOT 5: X.ARRAY,Y.ARRAY
                SET VIEWPORT .01*I.INT%,1-(I.INT%*.01), \
                        .01*I.INT%,1-(I.INT%*.01)
        NEXT I.INT%
        SET JUSTIFY .5,.5
        SET COLOR 1
        SET VIEWPORT 0,1,0,1
        GRAPHIC PRINT AT (50,50): "THANKS FOR THE VIEWING"
        KEY% = PAUSE
        STOP
        END
```

End of Appendix A

Index

Reader Comment Card

We welcome your comments and suggestions. They help us provide you with better product documentation.

Date _____ First Edition: May 1983

1. What sections of this manual are especially helpful?

2. What suggestions do you have for improving this manual? What information is missing or incomplete? Where are examples needed?

3. Did you find errors in this manual? (Specify section and page number.)

CBASIC Compiler™ Language Graphics Guide

COMMENTS AND SUGGESTIONS BECOME THE PROPERTY OF DIGITAL RESEARCH.

From: _____

_____

_____

_____

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS / PERMIT NO. 182 / PACIFIC GROVE, CA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL RESEARCH™

P.O. Box 579
Pacific Grove, California
93950

Attn: Publications Production