I N T E G E R G R A P H I C S L I B R A R Y

I G L

R E F E R E N C E M A N U A L

FEBRUARY, 1982 REV. B

TABLE OF CONTENTS

1. INTRODUCTION

The Integer Graphics Library (IGL) is an enhancement Library for the MTU BASIC Language. It provides a simple set of commands which is sufficient for many graphics applications. The main features of IGL are its small size and high speed. In most cases it will only subtract 3/4 K from BASIC's program space when it is loaded. To achieve its speed and small size, the IGL Library uses the philosophy that X,Y coordinates used in the commands will refer to actual coordinates on the 480 X 255 pixel array screen. This philosophy allows the IGL Library to directly access the high speed drawing routines in the GRAPHDRIVER.Z program. This philosophy also makes the IGL Library simple to use.

When IGL is "linked" into BASIC (see the LIB command description in the MTU BASIC Reference manual), simple BASIC-like statements can be used to perform drawing operations on the screen. For example, the statements:

```
SMOVE 200,100
SDRAW I+200,INT(D(I))+100
```

will draw a solid line from a point near the middle of the screen (X=200, Y=100) to a point determined by the current values of I and D(I). Note that IGL will accept as parameters any valid numerical constant, variable, or expression which greatly simplifies its use. Other commands allow you to draw dotted lines, erase lines, and even "flip" lines (white-to-black where drawn or visa-versa). IGL commands may be executed in "immediate mode" as well as from a program.

IGL also allows you to annotate your drawings with text that may be located anywhere on the screen, not just at character row and column positions. For example, the statements:

```
SMOVE 200,86
LABEL "Frequency"
```

will plot the label "Frequency" starting at X=200 and Y=86. String variables as well as the character representation of numeric constants and variables may also be plotted directly by the LABEL command.

IGL even allows BASIC programs to use the light pen or a special interactive function called "GRIN" to quickly and accurately get position data from the user. The light pen is capable of instantly returning the X and Y coordinates of the light it saw, not just the fact that it saw light.

2. LOADING INSTRUCTIONS

To load the IGL Library, simply include the name "IGL" as a file name in a LIB command. For example, you could use:

```
LIB "IGL"
```

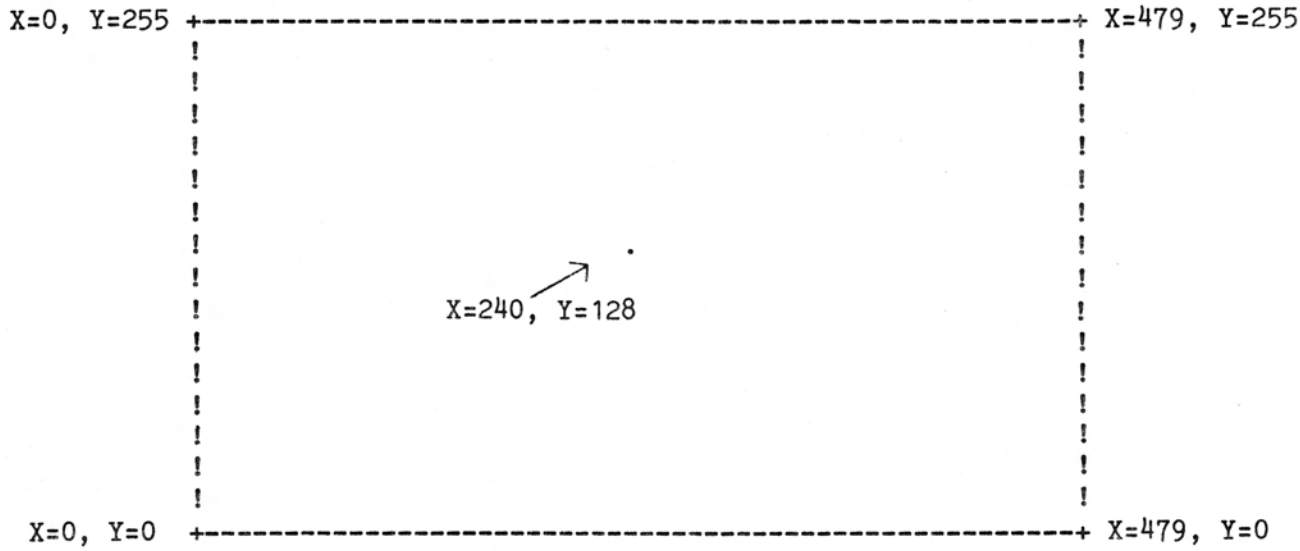
The IGL Library program is contained in the disk file IGL.Z. The ".Z" part of the name is assumed by the LIB command. If the IGL Library is not on the default drive (usually 0), then you should specify the drive number as below:

```
LIB "IGL:1"
```

for example. Actually, there are many copies of the program in the file, each assembled to run at a different location. This is done to give the LIB command a choice of where it may load the IGL Library so that it will not conflict with libraries already in memory.

3. DRAWING COMMANDS

Before beginning the descriptions of the various drawing commands, a discussion about X and Y coordinates is in order. The CRT screen, as it pertains to graphics, is organized as a 480 by 256 dot array. Each dot in the array may be referenced using an X,Y coordinate pair. The dot at coordinates 0,0 is found in the lower left corner of the screen. Thus the coordinates 479,255 refer to the dot in the upper right corner. In the remainder of this discussion, coordinates which refer to an actual pixel on the CRT display will be called absolute screen coordinates.



The X,Y coordinates specified in the drawing commands described below will be interpreted as absolute screen coordinates. This may differ from other libraries where X,Y coordinates undergo computations before absolute screen coordinates are arrived at. This implies that the X coordinate should be in the range 0 to 479, and the Y coordinate in the range 0 to 255. If a coordinate exceeds its range, the nearest limit is used instead. For example, if an X coordinate of -30 was used, 0 would be substitued since -30 is outside the valid range. Or, if a Y coordinate of 300 was used, 255 would be substituted.

A few of the drawing commands use relative coordinates instead of absolute screen coordinates. In this case the absolute screen coordinates are computed by adding the relative coordinates to the current screen coordinates. The range for relative coordinates is -128 to +127 for both X and Y. If a relative coordinate exceeds this range, an ILLEGAL QUANTITY ERROR is given.

3.1 THE DASHPAT COMMAND

PURPOSE: To set the dash pattern used when drawing dashed lines.

SYNTAX: DASHPAT byte , byte

ARGUMENTS:

byte = an expression which evaluates to a number between 0 and 255.

DISCUSSION:

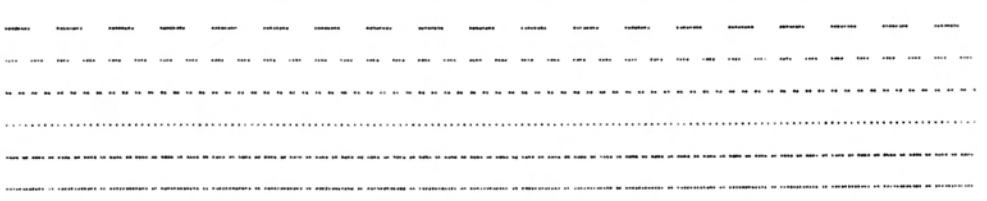
The DASHPAT command is used to set the dash pattern used when drawing dashed lines. The command sets the two bytes in memory which make up the dash pattern. The value of the first argument in the command is stored in the first byte, and the value of the second argument stored in the second byte.

For information on using the dash pattern, refer to the PENMODE command, and Appendix B.

EXAMPLES:

The following DASHPAT commands result in the dash pattern shown to the right of the command.

DASHPAT 255,0
DASHPAT 240,240
DASHPAT 204,204
DASHPAT 170,170
DASHPAT 246,246
DASHPAT 255,246



NOTES:

1. You may also PEEK and POKE the dash pattern directly. The first byte of the dash pattern is found at decimal location 523. The second byte is found at decimal 524.

3.2 THE LABEL COMMAND

PURPOSE: To print a string of characters beginning at the current screen coordinates.

SYNTAX: LABEL "⟨expression⟩"

ARGUMENTS:

"⟨expression⟩" = any valid numeric or string expression.

EXAMPLES:

```
30 LABEL "TIME"
```
will print the characters "TIME" starting at the current screen coordinates.

```
40 LABEL A$
```
will print the current content of the string A$ starting at the current screen coordinates.

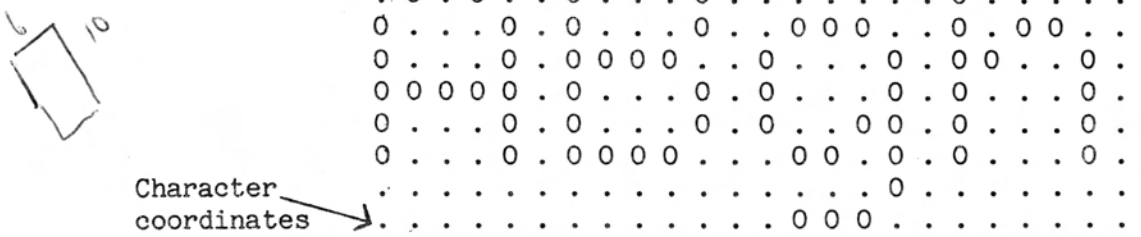
```
10 LABEL 100
```
will print the characters " 100" starting at the current screen coordinates.

```
20 LABEL 20*XF+5
```
will evaluate the expression and print the result at the current screen coordinates. note that the result my appear in exponential notation.

NOTES:

1. If the expression evaluates to a number, it is automatically converted to a string before it is printed. In this case, remember that positive numbers will have a leading blank and that this blank will erase whatever previously plotted lines it covers.

2. Characters are drawn using a 5 wide by 9 high dot matrix in a cell 6 units wide by 10 units high. The lower left corner of the first character drawn is at the current screen coordinates as shown below:



3. All characters except the underline ("_") will erase the 6 wide by 10 high cell before being drawn.

4. The current X coordinate is incremented by 6 after each character is drawn.

5. If the text runs off the right edge of the screen, the excess characters will not be drawn at all.

Does not handle null string

3.3 THE PENMODE COMMAND

PURPOSE: To set the drawing mode to be used by the SPEN and SRPEN commands.

SYNTAX: PENMODE mode

ARGUMENTS:

mode = any numeric expression which evaluates to a number from 0 to 7. This number will be interpreted as a drawing mode according to the following table:

0	MOVE	4	MOVE
1	DRAW	5	DRAW DASHED
2	ERASE	6	ERASE DASHED
3	FLIP	7	FLIP DASHED

DEFINITIONS:

MOVE - means that subsequent SPEN or SRPEN commands will move to the specified coordinates without drawing anything or shifting the dash pattern.

DRAW - means that subsequent SPEN or SRPEN commands will draw white lines regardless of the background color.

ERASE - means that subsequent SPEN or SRPEN commands will draw black lines regardless of the background color. If this black line precisely overlays a previously DRAWn line, the effect will be to erase it.

FLIP - means that subsequent SPEN or SRPEN commands will draw lines opposite in color to the background on a point-by-point basis. If this flipped line precisely overlays a previously flipped line, the effect will be to erase it and restore the background background color.

DASH - means that lines DRAWn, ERASEd, or FLIPped will be dashed rather than solid, i.e., only some of the points along the line will be affected. The default dash pattern (4 on and 4 off) may be altered by POKEing into location 523 and 524 (decimal). Each of the 16 bits represents a dot in the pattern. If the bit is a zero, the dot will not be plotted. If it is a one, the dot will be plotted. Some useful patterns are:

DASHPAT 255,0
DASHPAT 240,240
DASHPAT 204,204
DASHPAT 170,170
DASHPAT 246,246
DASHPAT 255,246

EXAMPLES:

```
10 PENMODE 5: DASHPAT 255,246
```

will prepare for drawing dashed lines using the bottommost example pattern above.

NOTES:

1. The PENMODE does not affect the SDRAW and SRDRAW commands

2. See Appendix B for more information about dashed lines.

3.4 THE SDRAW COMMAND

PURPOSE: To draw a solid white line from the current screen coordinates to a specified point.

SYNTAX: SDRAW ⟨x-coord⟩ , ⟨y-coord⟩

ARGUMENTS:

⟨x-coord⟩ = X screen coordinate, any numerical expression between 0 and 479.
⟨y-coord⟩ = Y screen coordinate, any numerical expression between 0 and 255.

EXAMPLES:

```
10 SDRAW 200,150
```

will draw a solid white line from the current screen coordinates to X=200 and Y=150 and then set the current coordinates to X=200 and Y=150.

```
30 SDRAW XF*RND(TT),YF*Y
```

will draw from the current screen coordinates to a random X coordinate and Y=YF*Y and then set the current coordinates to the new point.

NOTES:

1. The point drawn to becomes the new current screen coordinates when drawing is completed.

2. If the specified point is off the screen, then the nearest point on the edge of the screen is used. Note that this is not "image clipping" in the usual sense because the angle of lines to off-screen points may be altered.

3.5 THE SFILL COMMAND

PURPOSE: To fill a rectangular area according to the current drawing mode.

SYNTAX: SFILL ⟨xmin⟩ , ⟨xmax⟩ , ⟨ymin⟩ , ⟨ymax⟩

ARGUMENTS:

⟨xmin⟩ = X coordinate of the left edge of the rectangle, any numerical expression between 0 and 479.
⟨xmax⟩ = X coordinate of the right edge of the rectangle, any numerical expression between 0 and 479 that is greater than ⟨xmin⟩.
⟨ymin⟩ = Y coordinate of the bottom edge of the rectangle, any numerical expression between 0 255.
⟨ymax⟩ = Y coordinate of the top edge of the rectangle, any numerical expression between 0 255.

DISCUSSION:

The SFILL command "fills" a specified rectangle. The "filling" occurs by drawing horizontal lines starting at the bottom of the rectangle and moving up one pixel as each line is drawn. This continues until the top of the rectangle is reached.

Each of the horizontal lines is drawn using the current drawing mode. Depending on the drawing mode, the FILL command can have different effects. If the draw solid line mode is set (PENMODE 1), all the dots in the rectangle will be set to the "on" state. If the erase line mode is set (PENMODE 2), the rectangle will be cleared. If the dashed line mode is set (PENMODE 5), the pattern in the rectangle will vary depending on the width of the rectangle and the dash pattern.

If any arguments specified in the SFILL command are outside their legal range, they will be replaced with the nearest limit.

EXAMPLES:

```
SFILL 0,10,0,25
```

will "fill" the rectangle with corners at 0,0; 10,0; 10,25; and 0,25.

```
SFILL X,X+DX,Y,Y+DY
```

will "fill" the rectangle with the lower left corner at coordinate at X,Y; with width and heigth of DX and DY, respectively (DX and DY assumed to be positive).

NOTES:

1. The SFILL command permits one exception to the syntax given above. It permits the ymin and ymax arguments to be switched. If the first argument is found to be greater than the second argument, the horizontal lines will be drawn starting at the top of the recangle and will proceed downward.

2. Using the SFILL command will have no effect on the current screen coordinates.

3.6 THE SMOVE COMMAND

PURPOSE: To establish new current screen coordinates.

SYNTAX: SMOVE <x-coord>,<y-coord>

ARGUMENTS:

<x-coord> = X screen coordinate, any numerical expression between 0 and 479.
<y-coord> = Y screen coordinate, any numerical expression between 0 and 255.

EXAMPLES:

```
10 SMOVE 100,100
```
will establish the current screen coordinates at X=100 and Y=100.

```
20 SMOVE X1,Y1+20
```
will establish the current screen coordinates at X=X1 and Y=Y1+20

```
30 SMOVE XF*SIN(TH),YF*Y
```
will establish the current screen coordinates at X=XF*SIN(TH) and Y=YF*Y

NOTES:

1. If X <0, then X is set to 0. If X> 479, then X is set to 479.

2. If Y <0, then Y is set to 0. If Y> 255, then Y is set to 255.

3.7 THE SPEN COMMAND

PURPOSE: To draw a line from the current screen coordinates to a point using the current drawing mode as set by PENMODE.

SYNTAX: SPEN <x-coord> , <y-coord>

ARGUMENTS:

<x-coord> = x coordinate of point to draw to (see SDRAW command).
<y-coord> = y coordinate of point to draw to (see SDRAW command).

EXAMPLES:

```
10 SPEN 100,100
```
Will move the pen from the current screen coordinates to X=100 and Y=100 drawing the type of line specified by the current pen mode as it moves.

NOTES:

1. If the pen mode specifies a dashed line, the dash pattern will "recirculate" as the drawing is done. This allows dashed curves with dash continuity around vertices to be easily drawn. The pattern may be re-initialized by a new DASHPAT command.

2. If a line is drawn using FLIP mode and is later erased by FLIPping it again, then any other lines it may have crossed will be restored.

3. To successfully erase a dashed line, the initial state of the dash pattern must be the same as it was when the line was first drawn.

3.8 THE SRDRAW COMMAND

PURPOSE: To draw a solid line from the current screen coordinates a point relative to the current screen coordinates.

SYNTAX: SRDRAW ⟨x-distance⟩ , ⟨y-distance⟩

ARGUMENTS:

⟨x-distance⟩ = X distance to the endpoint of the line, any numerical expression between -128 and +127.
⟨y-distance⟩ = Y distance to the endpoint of the line, any numerical expression between -128 and +127.

EXAMPLES:

```
10 SRDRAW 10,-10
```

will draw a downward sloping line from the current screen coordinates to a point 10 units to the right and 10 units below the current screen coordinates and then update the current screen coordinates to this point.

NOTES:

1. If the specified point is off the screen, then the nearest point on the edge of the screen is used. Note that this is not "image clipping" in the usual sense because the angle of lines to off-screen points may be altered.

2. Note 2 for SRMOVE also applies here.

3.9 THE SRMOVE COMMAND

PURPOSE: To establish new screen coordinates from which further drawing is to occur, relative to the current screen coordinates.

SYNTAX: SRMOVE ⟨x-distance⟩ , ⟨y-distance⟩

ARGUMENTS:

⟨x-distance⟩ = distance from current X coordinate to new X coordinate, any numerical expression between -128 and +127.
⟨y-distance⟩ = distance from current Y coordinate to new Y coordinate, any numerical expression between -128 and +127.

EXAMPLES:

```
10 SRMOVE 10,-10
```

will establish new screen coordinates at a point 10 units to the right and 10 units below the current screen coordinates.

NOTES:

1. Positive distances move to the right and up for X and Y respectively. Negative distances move to the left and down for X and Y respectively.

2. If the x-distance or the y-distance is out of range, an ILLEGAL QUANTITY ERROR is given.

3. If the new screen coordinates are off the screen, they will be forced to the nearest screen edge.

3.10 THE SRPEN COMMAND

PURPOSE: To draw from the current screen coordinates to a point relative to the current screen coordinates, using the current drawing mode as set by SPEN.

SYNTAX: SRPEN ⟨x-distance⟩ , ⟨y-distance⟩

ARGUMENTS:

⟨x-distance⟩ = X distance to the endpoint of the line (see SRDRAW command).
⟨y-distance⟩ = Y distance to the endpoint of the line (see SRDRAW command).

EXAMPLES:

```
10 SRPEN 10,-10
```

will move the pen along a downward sloping line from the current screen coordinates to a point 10 units to the right and 10 units below the current screen coordinates drawing the type line specified by the current pen mode as it moved. It then updates the current screen coordinates to this point.

NOTES:

1. Positive distances draw to the right and up for X and Y respectively. Negative distances draw to the left and down for X and Y respectively.

2. See notes 1, 2, and 3 of SPEN.

4. GRAPHICS INPUT COMMANDS

The three IGL commands described in this section input information concerning the graphics display. The first two commands, SLTPEN and SGRIN, provide means for a program to obtain a coordinate pair from the user. The third command, SQDOT, which is actually a function, returns the state of a particular pixel.

4.1 THE SGRIN COMMAND

PURPOSE: To input an X,Y coordinate pair from the user using the GRIN routine.

SYNTAX: SGRIN ⟨string variable⟩ , ⟨x-variable⟩ , ⟨y-variable⟩

AGRUMENTS:

⟨string variable⟩ = any string variable. This variable will receive the character used to terminate the input.
⟨x-variable⟩ = any numeric variable which is to receive the X-coordinate.
⟨y-variable⟩ = any numeric variable which is to receive the Y-coordinate.

EXAMPLES:

```
10 SGRIN S$,XS,YS
```

will draw a flashing cross hair the full width and heigth of the display. The cursor keys are used to move the cross hairs. If the shift key is held down in conjunction with one of the cursor keys, the speed of movement is increased. Once the crossing point has been positioned over the desired pixel, hitting a key other than a cursor key will terminate the SGRIN command. The The X and Y coordinates of the selected pixel are returned in XS and YS respectively. The character key used to terminate the command is returned in S$.

NOTES:

1. The crosshairs are drawn in "flip" mode so that they will show up regardless of the background and so that they can be non-destructively moved through any existing image content.

2. The initial position of the crosshairs is the current screen coordinates as defined by a previous move or draw command.

3. The current screen coordinates are updated to the crosshair position when the termination key is pressed.

4. If one of the non-ASCII keys is used to terminate SGRIN, the character code returned will be according to the following table where S$ represents the first parameter of the SGRIN command:

KEY	UNSHIFTED ASC(S$)	SHIFTED ASC(S$)
f1	128	144
f2	129	145
f3	130	146
f4	131	147
f5	132	148
f6	133	149
f7	134	150
f8	135	151

KEY	UNSHIFTED ASC(S$)	SHIFTED ASC(S$)
HOME	164	180
PF1	136	152
PF2	137	153
×	138	154
÷	139	155
-	140	156
+	141	157
ENTER	142	158
INSERT	166	182
DELETE	165	181

4.2 THE SLTPEN COMMAND

PURPOSE: To input an X,Y coordinate pair from the user, using the light pen.

SYNTAX: SLTPEN <flag-variable> , <x-variable> , <y-variable>

ARGUMENTS:

<flag-variable> = a numeric variable which will receive the "hit flag".
<x-variable> = a numeric variable which will receive the X coordinate.
<y-variable> = a numeric variable which will receive the Y coordinate.

EXAMPLES:

10 SLTPEN FG,X,Y

will wait up to 1/30 second for the light pen to see light. If light is seen, FG will be set to 1.0, the horizontal position of the pen will be returned in X, and the vertical position of the pen will be returned in Y. X will always be in the range of 0 through 479 and Y in the range of 0 to 255. If no light is seen within 1/30 second, FG is set to 0 and both X and Y are unchanged.

15 SLTPEN FG,X(I,J),Y(I,J)

performs as the first example except that the pen coordinates are stored directly into the two-dimensional V and W arrays.

NOTES:

1. The X and Y coordinates returned have a ± 2 pixel uncertainty. This uncertainty will be more apparant in the X coordinate.

2. The light pen is not sensitive to lines or features less than two coordinate units wide in the horizontal direction.

3. The actual time spent executing this command varies from a minimum of less than a millisecond to a maximum of 33 milliseconds. The exact time required depends on whether the pen sees light and when the command was executed relative to the 16.66 millisecond display refresh cycle.

4. Clean the face of the CRT daily prior to first use of the light pen.

4.3 THE SQDOT FUNCTION

PURPOSE: To obtain the on/off status at a particular screen coordinate.

SYNTAX: SQDOT(<x-coord> , <y-coord>)

ARGUMENTS:

<x-coord> = X screen coordinate of point to test.
<y-coord> = Y screen coordinate of point to test.

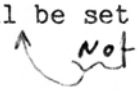
EXAMPLES:

```
10 PX=SQDOT(100,100)
```

The point (pixel) at X=100 and Y=100 is examined. If it is off (black), PX will be set to zero. If it is on, PX will be set to one.

NOTES:

1. The current screen coordinates will be set to the coordinates specified by the SQDOT command.



5. SCREEN CONTROL COMMANDS

These commands are used for clearing the screen and for setting the text window used by BASIC for all of its normal text output.

5.1 THE SCLEAR COMMAND

PURPOSE: To clear the entire screen, including the function key legends.

SYNTAX: SCLEAR

ARGUMENTS: none.

EXAMPLE:

```
10 SCLEAR
```
will clear the entire 480 by 256 screen to black.

NOTES:

1. This command will always clear the function key legends. If you wish to just clear the text window without clearing the legends, execute a PRINT CHR$(12) statement.

5.2 THE SPRINT COMMAND

PURPOSE: To make a hard copy of the image on the screen.

SYNTAX: SPRINT ⟨arg⟩

ARGUMENTS:

⟨arg⟩ = an expression which evaluates to a number from 0 to 255. It defaults to 0, if not specified.

DISCUSSION:

The SPRINT command provides a simple way of obtaining hard copy of the graphics image on the screen. The command itself executes by loading the file named SPRINT.Z from disk. It then executes the SPRINT.Z code, passing the argument value in the A register. If this command is used in conjunction with one of the MTU supplied "SPRINT" files, operation will be as follows.

The SPRINT command will make a dot for dot copy of the image on the screen to the appropriate printer. The argument controls whether or not the legends are included in the copy, and whether or not any paper is fed after the image is printed. When paper is fed, the amount fed is such that two screen images can be printed per page. The exact effect of the argument is as follows:

0 = Print screen image including legends, with paper feed.
1 = Print screen image not including legends, with paper feed.
2 = Print screen image including legends, without paper feed.
3 = Print screen image not including legends, without paper feed.

EXAMPLES:

SPRINT

will print the screen image including legends, and feed some paper.

SPRINT 3

will print the screen image not including the legends, and will not feed any paper once the image is printed.

NOTES:

1. When paper feed after print is specified (arg =0 or 2), the amount of paper feed is such that two images will fit onto an 8.5" x 11" page and the top-of-form position will not "drift" when many images are printed in succession.

2. Printing without paper feed may be used to join successive images together for greater total image area.

3. There are several SPRINT programs provided to perform dot graphics screen prints. The names of the files on disk will have appropriate letters appended to SPRINT to indicate which printer they are used with. You will have to rename the version appropriate to your printer to SPRINT.Z in order for IGL to use it. If you have another printer which is capable of dot-addressable graphics, you may assemble your own custom SPRINT.Z program by modifying one of the other source programs provided on the distribution disk.

5.3 THE TCURSOR COMMAND

PURPOSE: To establish a new character cursor position from which to start printing.

SYNTAX: TCURSOR ⟨column⟩ , ⟨row⟩

ARGUMENTS:

⟨column⟩ = the character number of the new cursor position. Can be any numerical expression. Range between 1 and 80.
⟨row⟩ = the line number of the new cursor position. Can be any numerical expression. Range between 1 and number of lines in text window.

EXAMPLES:

```
10 TCURSOR 1,1
```
will place the text cursor in the upper left corner of the text window.

```
20 TCURSOR N,LEN(A$)
```
will place the text cursor on the Nth line of the text window in the character position that would be occupied by the last character of the string A$ if it was printed in the first character position.

NOTES:

1. If you wish to position text at arbitrary positions without regard to line and column numbers, see the LABEL command.

2. The window established by TCURSOR applies to all console output from BASIC including the LIST command and user typed input.

3. Setting a new cursor position also sets the value returned by the POS() function to the specified column.

ROW > NLINES Then will rap Modulo Flines

5.4 THE TWINDOW COMMAND

PURPOSE: To establish the current text window.

SYNTAX: TWINDOW <top> , <lines>

ARGUMENTS:

<top> = the number of coordinate units down from the top of the screen that will be the top of the text window. Range is from 0 to 246. Can be any numerical expression.

<lines> = the number of text lines in the text window. Range from 1 to 24. Can be any numerical expression.

EXAMPLES:

```
10 TWINDOW 0,24
```

will specify a text window that starts at the top of the screen and is 24 lines deep. This is the usual default setting and reserves space at the bottom of the screen for the function key legends.

```
20 TWINDOW 255-YT,SC/10
```

will specify a text window whose top limit is at Y coordinate of YT and whose height is SC coordinate units.

NOTES:

1. The text window is used to restrict where printed text can appear on the screen. Positioning the cursor beyond the allowed line limit will simply wrap to to the next line within the window or will scroll the contents up one line.

2. Each line of text requires 10 coordinate units vertically.

3. The text window is always 80 characters (480 coordinate units) wide.

255 - (# pixels in text window) will set at bottom of screen

for ex 2 lines text + Legend area

20 + 15

↑ You must make it > than Legends

Legends else it writes over Legend

minimum is 9 pixels + only 1 lines window

APPENDIX A.

COMMAND SUMMARY

INTEGER GRAPHICS LIBRARY PROGRAM - IGL.Z

This enhancement library adds a simple set of graphics commands to MTU BASIC. The following is a list of the commands:

```
DASHPAT Y,Y          - Sets pattern used when drawing dashed lines.
LABEL   S$           - Print characters in S$ at current cursor location.
PENMODE M            - Set the drawing mode to M.  See below for list of modes.
SCLEAR               - Clear the entire screen.
SDRAW   X,Y          - Draw a solid line from current cursor location to X,Y.
SFILL X1,X2,Y1,Y2    - Fills rectangular area defined by values of X1,X2 and Y1,Y2.
SGRIN   S$,X,Y       - Obtain X,Y using GRIN cursor.
SLTPEN  X,Y          - Obtain X,Y using light pen input.
SMOVE   X,Y          - Move current cursor location to X,Y.
SPEN    X,Y          - Draw from current cursor location to X,Y according to mode.
SPRINT  Y            - Makes hard copy of screen image with and without legends
                       and/or with and without spacing between hard copy printing.
SQDOT   (X,Y)        - Returns the status of dot X,Y.  0 = OFF, 1 = ON.
SRDRAW  RX,RY        - Draw to current cursor location plus RX,RY.
SRMOVE  RX,RY        - Move to current cursor location plus RX,RY.
SRPEN   RX,RY        - Draw to current cursor location plus RX,RY according to mode.
TCURSOR C,R          - Set Text Cursor to column C and row R.
TWINDOW T,N          - Set Text Window Top to T'th scan line from the top of the
                       display and allow N text lines.
```

The parameters used above are defined as follows:

```
X  - an expression in the range 0 to 479
Y  - an expression in the range 0 to 255
RX - an expression in the range -128 to 127
RY - an expression in the range -128 to 127
R  - an expression in the range 1 to 24
C  - an expression in the range 1 to 80
T  - an expression in the range 0 to 255
N  - an expression in the range 1 to 24
S$ - a string variable
M  - an expression in the range 0 to 7
Graphics cursor - the current screen coordinates.
```

The modes for M are defined as follows:

0	MOVE	4	MOVE
1	DRAW	5	DRAW DASHED
2	ERASE	6	ERASE DASHED
3	FLIP	7	FLIP DASHED

The 16 bit dash pattern is found at $20B and $20C or 523 and 524 decimal.

APPENDIX B.

USING THE DASH PATTERN

If you wish to precisely control how dashed lines appear on the screen, you must take into account what happens as each pixel of the line is plotted. When you use the SPEN or SRPEN command with the mode set to 5, 6, or 7, the dash pattern will be tested as each pixel in the line is addressed to determine if that pixel should be modified. The following paragraphs explain the effect of the dash pattern on the pixels of the line.

The dash pattern consists of the 16 bits found in the bytes at locations 523 and 524 (decimal). Just before each pixel is plotted, the most significant bit of the byte in 523 is tested. If the bit is a zero the pixel is left unchanged. If it is a one, the pixel will be modified. Modification means setting the pixel to white for mode 5 (DRAW DASHED), to black for mode 6 (ERASE DASHED), and to the opposite state for mode 7 (FLIP DASHED). Before proceeding to the next pixel, the bits in the dash pattern are all shifted toward the the most significant bit just tested. The most significant bit is recirculated back the the least significant bit in the byte in 524 (decimal). This process is performed for each pixel until the entire line is drawn.

As you can see, the dash pattern will be applied to a line starting with the first pixel, and proceed in the direction that the line is plotted. For example, if you set location 523 to 255 (all one bits) and 524 to 0 (all zero bits), drawing in mode 5 would turn the first 8 pixels plotted to white. The next 8 pixels would be left unchanged, the next white, and so forth.

What the above discussion is intended to show is that the sequence in which the pixels are plotted is important in determining what will appear on the screen. Only in the case of <u>vertical lines</u> is plotting quaranteed to start with the current cursor coordinates and proceed toward the specified end point. In all other cases, plotting is guaranteed to start with the leftmost endpoint and proceed toward the other end point. This left to right direction occurs regardless of which end point is the current cursor coodinate, and which is the coordinate specified with the SPEN or SRPEN command. As an illustration of this, you might try running the following two programs.

```
10 DASHPAT 255,0
20 PENMODE 5
30 SMOVE 100,100
40 SPEN 200,200
```

```
10 DASHPAT 255,0
20 PENMODE 5
30 SMOVE 200,200
40 SPEN 100,100
```

Even though the programs will leave the current cursor coordinates at different locations, the line drawn will be exactly the same for both programs.

What this all leads up to is that in some cases it may be difficult to keep the dash pattern continuous from line to line. In cases where the dash pattern must be continuous, you must make sure the commands draw from left to right to match the plotting direction. This should be no problem for a conventionally plotted (i.e., independent variable on the horizontal axis and dependent variable on the vertical axis). graph. For vertical lines, the dash pattern always continues from the current cursor coordinates.