The 8-bit and 24-bit systems are described below.

1.1 8-Bit Model One/80

The 8-bit Model One/80, as mentioned above, has a single 8-bit-in, 24-bit-out look-up-table. The 256 addresses in the LUT are set to their default values at COLDstart and by a MODDIS 1 command. These default values are the same as the default for the Model One/25 1K addressing mode; addresses 0-63 are the same. Then, addresses 64-127, 128-191, and 192-255 repeat the same default values (the high two bits are ignored).

The LUt is set at COLDstart, with the range 0 to 63 specifying shades of red, green, and blue, from black (at 0) to white (at 63). The pixel value for any pixel drawn into image memory with the graphics primitives commands is selected with the VAL1K val command. (Note: the VALUE and VAL8 commands are provided for compatibility with other Model Ones and with the 24-bit Model One/80, but should only be used when compatibility is a requirement.)

The VAL1K command sets the current pixel value, which is then used to draw all subsequent graphics primitives. For example:

```
VAL1K 3
```

selects full-intensity blue;

```
VAL1K 63
```

selects white.

These are default values. The Model One/80's look-up-table can be changed by using the look-up-table commands.

The default look-up-table entries corresponding to the VAL1K command are:

	Color Red	Green	Blue
VAL1K 0,64,128,192	000	000	000
VAL1K 1,65,129,193	000	000	085
VAL1K 2,66,130,194	000	000	170
VAL1K 3,67,131,195	000	000	255
VAL1K 4,68,132,196	000	085	000
VAL1K 5,69,133,197	000	085	085
VAL1K 6,70,134,198	000	085	170
VAL1K 7,71,135,199	000	085	255
VAL1K 8,72,136,200	000	170	000
VAL1K 9,73,137,201	000	170	085
VAL1K 10,74,138,202	000	170	170
VAL1K 11,75,139,203	000	170	255
VAL1K 12,76,140,204	000	255	000
VAL1K 13,77,141,205	000	255	085
VAL1K 14,78,142,206	000	255	170
VAL1K 15,79,143,207	000	255	255
VAL1K 16,80,144,208	085	000	000
VAL1K 17,81,145,209	085	000	085

VALlK 18,82,146,210	085	000	170
VALlK 19,83,147,211	085	000	255
VALlK 20,84,148,212	085	085	000
VALlK 21,85,149,213	085	085	085
VALlK 22,86,150,214	085	085	170
VALlK 23,87,151,215	085	085	255
VALlK 24,88,152,216	085	170	000
VALlK 25,89,153,217	085	170	085
VALlK 26,90,154,218	085	170	170
VALlK 27,91,155,219	085	170	255
VALlK 28,92,156,220	085	255	000
VALlK 29,93,157,221	085	255	085
VALlK 30,94,158,222	085	255	170
VALlK 31,95,159,223	085	255	255
VALlK 32,96,160,224	170	000	000
VALlK 33,97,161,225	170	000	085
VALlK 34,98,162,226	170	000	170
VALlK 35,99,163,227	170	000	255
VALlK 36,100,164,228	170	085	000
VALlK 37,101,165,229	170	085	085
VALlK 38,102,166,230	170	085	170
VALlK 39,103,167,231	170	085	255
VALlK 40,104,168,232	170	170	000
VALlK 41,105,169,233	170	170	085
VALlK 42,106,170,234	170	170	170
VALlK 43,107,171,235	170	170	255
VALlK 44,108,172,236	170	255	000
VALlK 45,109,173,237	170	255	085
VALlK 46,110,174,238	170	255	170
VALlK 47,111,175,239	170	255	255
VALlK 48,112,176,240	255	000	000
VALlK 49,113,177,241	255	000	085
VALlK 50,114,178,242	255	000	170
VALlK 51,115,179,243	255	000	255
VALlK 52,116,180,244	255	085	000
VALlK 53,117,181,245	255	085	085
VALlK 54,118,182,246	255	085	170
VALlK 55,119,183,247	255	085	255
VALlK 56,120,184,248	255	170	000
VALlK 57,121,185,249	255	170	085
VALlK 58,122,186,250	255	170	170
VALlK 59,123,187,251	255	170	255
VALlK 60,124,188,252	255	255	000
VALlK 61,125,189,253	255	255	085
VALlK 62,126,190,254	255	255	170
VALlK 63,127,191,255	255	255	255

Table 1 LUT Values Corresponding to VALlK Commands

1.2 24-Bit Model One/80

The 24-bit Model One/80 can be treated as three 8-bit systems, selecting between memory units with the MEMSEL command. (NOTE: if the hardware has been connected for a true-color system, the result is three monohue systems, one red, one green, and one blue. To use the 24-bit system as three 8-bit systems, the hardware must be configured differently than for the true-color system.) In this case, the system operates as described above. More commonly, however, the 24-bit system is used for true-color applications. The commands for this are described in this section.

RGBTRU mode allows use of the 24-bit Model One/80 for true-color applications. In this mode, the value commands correspond directly to the look-up-tables by default, and there are three 8-bit-in, 8-bit-out LUTs. Each LUT corresponds to a primary color: red, green, and blue.

To enable RGBTRU mode, the RGBTRU flag command is used. flag=ON or 1 enables RGBTRU mode; flag=0 or OFF disables RGBTRU mode. Before the command is executed, the hardware must include three consecutive eight-bit memory units, the first of which is 0, 4, 8, or 12. Otherwise, the system will not work correctly. The MEMSEL command must be used to select the first memory unit of the triplet. (See the Command Reference for details of the MEMSEL command.)

Once RGBTRU mode is enabled, the value commands work as follows:

1. The VALUE command writes VALUE red green blue to the first, second, and third memory unit of the triplet, respectively.

2. The VAL8 command writes VAL8 value to the first, second, and third memory unit of the triplet.

3. The VAL16 command writes the high byte of the VAL16 value to each memory unit and discards the low byte. (Note that it acts differently when RGBTRU is OFF.)

4. The VAL1K command parses the value given as follows: the top two bits are discarded, the next two indicate the top bits for red, the next two the top bits for green, and the lowest two the top bits for blue. This results in a default value the same as those for VAL1K 0-63 for VAL1K with RGBTRU mode OFF.

2.0 The Look-Up-Tables

Six Model One commands are available to load the look-up-tables: LUTA, LUTB, LUTG, LUTR, LUT8, and LUTRMP.

LUTA index,entry changes the red, green, and blue components of the color at location index to entry, resulting in a shade of grey. The LUTA command is most useful for monochrome applications.

LUTB index, entry, LUTG index, entry, and LUTR index,entry change the Blue, Green, and Red components of the color at location index to entry, resulting in a new color for that location. Some examples will clarify this:

```
VALIK 63
FLOOD
```

will flood the screen with white; the command

```
LUTB 63,0
```

affects the blue component (by setting it to zero)--the screen then becomes yellow. Now, add the blue component back in and set the red component to zero with these commands:

```
LUTB 63,255
LUTR 63,0
```

and the screen now becomes cyan. Finally, if you add the red back in and subtract the green, you will see magenta:

```
LUTR 63,255
LUTG 63,0
```

Of course, you can use the LUTB, LUTG, and LUTR commands for any location in the look-up-table, to select the range of colors (up to 64 colors) you wish to use for your application.

LUT8 index rentry,gentry,bentry can be used to set all three components of the color at location index at once. For example, the command LUT8 63 50 100 200 changes location 63 in the look-up-table with these components determining the color: the red intensity is 50, the green intensity 100, and the blue intensity 200.

The LUTRMP num sind,eind sent,eent command is used to set a "ramp" of look-up-table values. The command's parameters are used as follows:

num indicates the components to write:
- num=1, write blue component
- num=2, write green component
- num=4, write red component
- num=7, write all components

sind,eind indicate the starting and ending locations in the look-up-table to write

sent,eent indicate the starting and ending entries to write into those locations.

For example, the command LUTRMP 4 0,63 255,255 will set the red component of all the entries in the look-up-table to full intensity. In using the LUTRMP command on the Model One/80, you should be aware that the only way to reset the look-up-table back to its default contents is a COLDstart or MODDIS 1 command, both of which erase the screen. You can also execute 256 LUT8 commands to reset each location individually to the values in Table 1; this--of course--does not require erasing the screen.

3.0 The Value Registers

The Model One uses sixteen value registers, called VREGs, to store the pixel values. Like the coordinate registers described in Coordinates and Image Memory, some value registers have predefined functions; others are available for use by the programmer. The example above, used to display the look-up table, uses one of the available value registers to store a constant value to be added to the current pixel value.

The current pixel value--the value that is used by all commands that write graphic data to the Model One--is always stored in VREG 0. Table 2 shows the value register assignments and mnemonics.

Value Register	Mnemonic	Use
VREG 0	CURVAL	The current pixel value; this is the value used by all commands that write graphic data to the Model One.
VREG 1	XHR0VAL	Crosshair 0 value.
VREG 2	XHR1VAL	Crosshair 1 value.
VREG 3	FILMSK	Fill mask used for area fills.
VREG 4-6		Reserved.
VREG 7-15		Available for temporary value storage.
VREG 16,19...40		Foreground color, alphanumeric windows 0-8.
VREG 17,20...41		Background color, alphanumeric windows 0-8.
VREG 18,21...42		Cursor color, alphanumeric windows 0-8.
VREG 43-50		Reserved.
VREG 51-63		Available for temporary value storage.

Table 2 Value Register Assignments

Five Model One commands are available to load and manipulate the contents of the value registers: VLOAD, VMOVE, VADD, VSUB, and RDPIXR.

VLOAD vreg r,g,b loads a 24-bit pixel value into any one of the sixteen value registers. Note that although the VLOAD command loads 24 bits of data, the 8-bit Model One/80 will discard the second and third byte of data. For example, VLOAD 10 1,0,0, used in Example 1, loads value register 10 with the value (1,0,0); this value is added to value register 0 by the VADD command

every time the macro is executed.

VMOVE vdst,vsrc moves the contents of one value register into another value register: VMOVE 10 11 copies the contents of value register 11 into value register 10.

VADD vsum,vreg and VSUB vdif,vreg add and subtract values between two value registers. As you saw above, VADD 0 10 adds the contents of value register 10 to the contents of value register 0; in the same way, VSUB 0 10 subtracts the contents of value register 10 from the contents of value register 0.

RDPIXR vreg places the value in image memory of the current point into the specified value register. RDPIXR 10 determines the value at the current point and then places that value into VREG 10. The RDPIXR command can be used to create a menu of colors, allowing the user to be given a choice of colors, select a color using the cursor, and that color made the current color: RDPIXR 0 makes the color at the current point the new current color.

The READVR vreg command does not affect the value within the value register: instead, the value of the specified VREG is displayed at the port that is in GRAPHICS mode (to the host if the host has sent the ENTERGRAPHICS character, to the local terminal if the local terminal has sent the ENTERGRAPHICS character).

4.0 The Pixel Processor

Whenever pixel data is written into image memory, the Model One's pixel processor is used. The pixel processor performs logic functions between incoming pixel data and the pixel values which are already in image memory. The pixel processor supports the logical and arithmetic functions INSert, SUBtract (two versions), ADD, XOR, OR, AND, PRESET and CONDITIONAL.

The pixel processor operates on data coming from the hardware vector generator (which may come from the host serial port) or from the host DMA (Direct Memory Access) port. All graphics primitives are drawn into image memory by the vector generator and are thus performed by the pixel processor.

The PIXFUN command controls the pixel processor.

The PIXFUN mode command controls the logic function to be performed by the pixel processor. PIXFUN has a single parameter, mode, which sets the pixel processor mode, as shown in Table 3. The number or the mnemonic may be used: PIXFUN 0 inserts data; PIXFUN AND ANDs incoming data with image memory.

Mode	Mnemonic	Pixel Function
0	INS	Replace image memory with incoming data
1	SUBI	Subtract image data from incoming data
2	SUBN	Subtract incoming data from image data
3	ADD	Add incoming data and image data
4	XOR	Exclusive OR incoming data values with image memory
5	OR	OR incoming data values with image memory
6	AND	AND incoming data values with image memory
7	PRESET	PRESET: write all 1's into image memory
8	CONDITIONAL	Conditional: inhibit writing of pixel values of (0,0,0)

Table 3 PIXFUN Modes and Functions

The PIXCLP flag command controls the clipping of values when values are ADDed or SUBtracted (as set with PIXFUN). flag=1 or ON enables clipping; the combined values are clipped to 255 if they add to more than 255 or to 0 if they subtract to less than 0. If clipping is disabled (flag=0 or OFF), the values are computed modulus 256: 256 is added or subtracted from the number repeatedly until a value in the range 0 to 255 is obtained (200+200=400-256=144, for example).

5.0 Write-Enable and Read-Enable Masks

In the Model One/80, the image memory planes can be selectively read-enabled and read-disabled, write-enabled and write-protected.

The Model One video output section includes an 8-bit register called a read-enable mask. The Read mask is ANDed with the data from image memory immediately before the data enters the look-up-table. If a bit in the read-mask is zero, the corresponding bit in the value is forced to zero.

The RDMASK mask command sets the read mask. For example, RDMASK 0 sets the read-mask to all zeroes and thus completely suppresses display of the image, forcing the value out of image memory to zero. This will not necessarily force the display to black; the display is dependent on the contents of the look-up-table at entry 0. The default read mask is FFH.

You should use the RDMASK command with caution because the Read mask register is logically "in front of" the look-up-table. Thus, it has the same effect on the addressing of the look-up-table as it does on image memory. NOTE: if you want to be sure you are changing only the correct look-up-table indices, you should first set the Read Mask to #FF, then issue the look-up-table commands, then reset the Read Mask to the desired value.

The WRMASK bitm,bankm command write-protects the image memory planes. The details of the settings for bitm and bankm are given in the Command Reference.

The RDMASK and WRMASK commands are used to store multiple images in image memory and select them for display.

Two other commands are available for setting the read and write masks: RMSK16 mask and WMSK16 mask. mask, which is the same for both commands, is a 16-bit mask corresponding directly to the bit planes. RDMASK and WRMASK are used for 24-bit systems; RMSK16 and WMSK16 are used for 8-bit systems.

6.0 Blinking Colors and the Blink Table

The Model One uses the look-up-table and a blink table to provide blinking colors. The blink table holds instructions for blinking of the red, green, and blue components of the value at a given look-up-table location. Up to 64 entries may be made in the look-up-table. For each index, a pair of entries is stored. The index gives the address in the look-up-table: the contents of that address is toggled between the pair of entries. The blink rate determines the amount of time each entry stays in the look-up-table.

Four commands are used for blinking colors: BLINKC, BLINKD, BLINKE, and BLINKR.

The BLINKC command clears the blink table and stops all blinking. After clearing, the first value given (entry1 of BLINKE) remains.

The BLINKD lut,index command removes a single entry from the blink table and leaves the rest of the blink table intact. For example, BLINKD 7 63 disables blinking of address 63 for all components.

The BLINKE lut,index entry1,entry2 command enables blinking of a specified index in the look-up-table, by specifying the component (red, green, blue, or all components), the index, and the intensities to blink between. For example, BLINKE 7 63 255,100 blinks all components of the color at location 63 between 255 and 100.

The BLINKR frames command sets the blink rate. The blink rate can be set to a multiple of the frame time (1/30th of a second). For example, BLINKR 30 sets the blink rate to once per second.

6.0 The CLEAR and FLOOD Commands

The CLEAR and FLOOD commands fill image memory with a uniform pixel value.

The CLEAR command fills the current clipping window (as defined by CREGs 9 and 10 or the WINDOW command) with the current pixel value. The selected pixel function (PIXFUN command) determines the actual pixel values that are left in image memory when the CLEAR is done. For example:

```
PIXFUN INS
VAL1K 63
CLEAR
```

inserts pixels with pixel value 63. This pixel value is then used to indicate the look-up-table value to be sent to the DACs for display (a 24-bit value). The CLEAR command does not affect the current point.

The FLOOD command instantly fills every displayed pixel with the current pixel value. If the screen is zoomed in when the FLOOD command is issued, only the displayed portion will be filled; FLOOD has no effect on undisplayed portions of the screen.

The VECPAT mask command can be used to change the fill pattern for a CLEAR command, by specifying a pattern of pixels to be repeated along every scan line during the execution of the CLEAR command. mask is a 16-bit parameter; for example, VECPAT #AAAA contains alternating ones and zeroes. CLEARing the image memory with VECPAT #AAAA set will create a dotted fill pattern. More information on the VECPAT command is given in Graphics Primitives.

The pixel function (PIXFUN command) does not affect the FLOOD command.

RASTER TECHNOLOGIES
MODEL ONE/80
GRAPHICS PRIMITIVES

Revision 1.0 February 27, 1984

MODEL ONE GRAPHICS PRIMITIVES
February 27, 1984

NOTICE:

The information contained in this document is subject to change without notice.

RASTER TECHNOLOGIES DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS MATERIAL (INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE), EITHER EXPRESS OR IMPLIED. RASTER TECHNOLOGIES SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM ANY ERROR CONTAINED HEREIN, INCLUDING, BUT NOT LIMITED TO, FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS MATERIAL.

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual may cause interference with regard to radio commmunications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Table of Contents

Figures

GRAPHICS PRIMITIVES

The Model One graphics primitive commands support local generation of common geometric entities: points, lines, circles, arcs, rectangles, polygons, and text. These entities are called graphics primitives and are used as building blocks for more complex images.

All commands which draw graphics primitives use the Model One's 16-bit virtual address space to define position and shape. The coordinate registers (current point, coordinate origin, and clipping window) control the placement of the graphics primitives within physical image memory, as described in Coordinates and Image Memory Addressing.

1.0 Points

The simplest graphic entity, the point, is drawn with the POINT command. POINT sets the pixel located at the current point (CREG 0) to the current pixel value (VREG 0); the current pixel value and current point are unchanged.

2.0 Lines

When drawing lines or vectors, the Model One uses the current point as the starting point for the line; a DRAW command then specifies the ending point of the line.

Five DRAW commands are available. All DRAW commands use the current pixel value to draw the line. The DRAW commands are: DRWABS, DRWREL, DRW2R, DRW3R, and DRWI. The current point is always set to the endpoint of the line after the DRAW command has executed.

The DRWABS x,y command draws a line from the current point to the given ending point. The current point is set to (x,y) after execution. For example, DRWABS 10,10 draws a line to (10,10) from the current point. After execution, the new current point is (10,10).

The DRWREL dx,dy command draws a vector from the current point to the point specified by adding dx to the X coordinate and dy to the Y coordinate. For example, DRWREL -10,-10 would draw a line to (-20,-20) from the current point of (-10,-10). The new current point will again be set to the ending point of the line: (-20,-20).

The DRW2R dx,dy and DRW3R dx,dy commands are special forms of the DRWREL command for use when the displacement to the ending point of the line is small. The DRW2R command requires only two bytes; DRW3R requires only three bytes. Details of DRW2R and DRW3R are given in the Command Reference.

The DRWI creg command draws a line from the current point to the point specified by the given coordinate register. As with the other DRAW commands, the current point is set to the endpoint of the new line after execution of the command. For example, if CREG 23 holds the value (-20,30) and the current point is (10,10), the command DRWI 23 draws a line from (10,10) to (-20,30) and sets the new current point to (-20,30).

Because all DRAW commands set the last pixel of the line to be the current point, drawing a series of lines can produce overwriting of the pixels at the beginning and ending points of the lines. This is particularly obvious if the current pixel function (PIXFUN) is ADD, SUB, or XOR. The FIRSTP flag command can be used to inhibit writing of the first pixel of each line as it is drawn. The command FIRSTP 1 inhibits writing of the first pixel; FIRSTP 0 allows writing of the first pixel. FIRSTP 0 is the default.

Lines drawn by the Model One's DRAW commands may be solid, dotted, dashed, or any repetitive pattern you can specify with a 16-bit parameter. The VECPAT mask command specifies the pattern of pixels to be repeated along every subsequent line that is drawn. VECPAT #AAAA contains alternating ones and zeroes: lines drawn with this mask are drawn with every other pixel omitted and appear dotted. VECPAT #F0F0 draws lines with four pixels drawn followed by four omitted: these lines appear dashed. Other values will create other patterns. Repeating patterns can be generated; every time a pixel is written, the vector pattern is rotated by a single bit, without regard to the starting and ending points of lines.

Note that the VECPAT command also affects filling of graphics primitives (the PRMFIL command) and the fill pattern used in CLEAR.

The default mask for VECPAT is #FFFF, so that every pixel along the line is drawn.

3.0 Circles and Arcs

Three commands can be used to draw a complete circle: CIRCLE, CIRCXY, and CIRCI.

The CIRCLE radius command draws a circle of the specified radius centered around the current point. For example, to draw a circle with a radius of 30, centered around (50,50), use these commands:

```
MOVABS 50,50
CIRCLE 30
```

The CIRCXY x,y command draws a circle centered around the current point; point (x,y) is on the circumference. (The distance from the current point to point (x,y) determines the radius.) For example, if the current point is (0,0) and the command CIRCXY 10,0 is executed, a circle centered around (0,0), with (10,0) on its circumference, will be drawn.

The CIRCI creg command draws a circle centered around the current point; the point specified by creg lies on the circumference. For example, if the current point is (10,10) and CREG 21 holds point (25,25), the command CIRCI 21 draws a circle whose center is (10,10) and with point (25,25) on the circumference. The CIRCI command can be used to specify arbitrary circles interactively, using the digitizing tablet: the cursor can be used to specify the center point, then used again to specify the radius (by specifying a point on the circumference).

A single Model One command is available for drawing arcs: ARC. The ARC rad,al,a2 command draws an arc around the current point by giving the radius (rad), the starting angle (al), and the ending angle (a2). The starting and ending angles are given in one-degree increments counterclockwise; zero degrees is the positive X-axis; ninety degrees is the positive Y-axis.

For example, ARC 10 0,90, with the current point at (0,0), draws an arc of radius 10 from (0,10) to (10,0).

4.0 Rectangles

Three commands are available to draw rectangles: RECTAN, RECREL, and RECTI.

The RECTAN x,y command draws a rectangle: the current point defines one corner, and point (x,y) defines the diagonal corner. For example, with the current point at (10,10), the command RECTAN 20,20 draws a rectangle with (10,10) as one corner and (20,20) as the diagonally opposite corner.

The RECREL dx,dy command draws a rectangle using the current point as one corner and (dx,dy) as the displacement from the current point to specify the diagonally opposite corner. For example, with the current point at (100,100), the command RECREL 10,10 draws a rectangle with (100,100) as one corner and (110,110) as the diagonal corner.

The RECTI creg command draws a rectangle, using the current point as one corner and the point specified by creg as the diagonal corner. If the current point is (100,100) and CREG 21 holds the point (10,10), the command RECTI 21 draws a rectangle with (100,100) as one corner and (10,10) as the diagonal corner.

The rectangle commands do not change the current point.

5.0 Polygons

The Model One's POLYGN command allows you to draw arbitrary polygons. The POLYGN command has the format

POLYGN npoly nverts,vert1,vert2...vertn nverts,vert1,vert2...vertn

The command is of varying length, depending on the number of polygons being drawn and the number of vertices in each polygon. The first parameter, npolys, specifies the number of polygons to be drawn, the second parameter, nverts, specifies the number of vertices in the first polygon. Then the vertices, vert1, vert2...vertn, are specified (as relative offsets to the current point). The number of vertices, nverts, in the second polygon may then be specified, and its vertices, vert1, vert2...vertn, and so on until all of the polygons have been drawn.

In using the POLYGN command, all vertices are relative to (or offset from) the current point; the current point is unchanged by the POLYGN command.

Because the POLYGN command allows specification of more than one polygon at a time, the Model One supports arbitrary polygons with interior holes. The polygons which are drawn by the POLYGN command can be filled or unfilled (see

section 7.0, below): with PRMFIL ON, the nested interior polygons (islands) will be left unfilled, while the surrounding polygon is filled, as shown in Figure 1.

Some examples will clarify the use of the POLYGN npoly nverts, vert1,vert2...vertn nverts,vert1,vert2...vertn command:

```
MOVABS 0,0
POLYGN 1 3 10,10 10,-10 -10,-10
```

will draw a triangle (a 3-vertex polygon) from (10,10) to (10,-10) to (-10,-10).

Because the POLYGN is drawn offset from the current point, you can change the current point and issue the same POLYGN command:

```
MOVABS 100,100
POLYGN 1 3 10,10 10,-10 -10,-10
```

The above commands draw the same polygon (a triangle) from (110,110) to (110,90) to (90,90).

The command sequence

```
MOVABS 20,0
POLYGN 2 3 10,10 10,-10 -10,-10 4 20,20 20,-20 -20,-20 -20,20
```

would draw a triangle and a rectangle; the triangle would be inside the rectangle. (If PRMFIL ON had been executed first, the rectangle would be filled, the triangle would not be.) Figure 1 shows the two polygons.

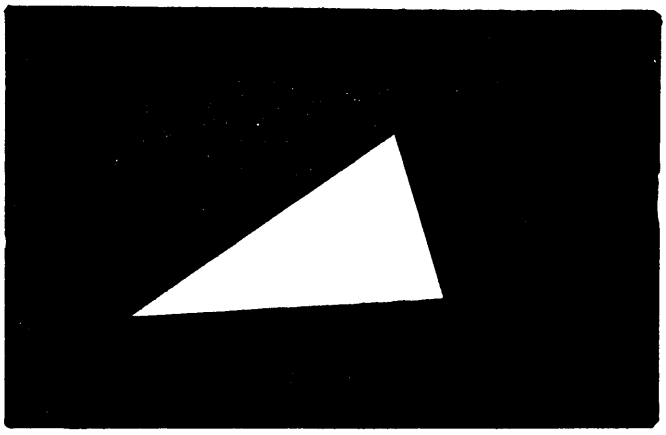

Figure 1 A Filled Polygon with a Hole

Whenever you use the POLYGN command, keep in mind that the vertices are relative to the current point. This facilitates dragging of the polygon interactively. For example, by writing macros (see Macro Programming) which set the pixel function (PIXFUN) to XOR, you could drag the polygon until the user wished to confirm it (without affecting image memory). Then, when the user confirms the location of the polygon, the PIXFUN could be set to change the value of pixel memory appropriately.

6.0 Text

The Model One supports seven text drawing commands: TEXT1, TEXT2, VTEXT1, VTEXT2, TEXTC, TEXTDN, and TEXTRE.

The TEXTC size,angle command sets the size and baseline orientation angle for subsequent text drawing commands. Text may be drawn at any angle, in one-degree increments counterclockwise; the scale factor may range from 0 to 255. A scale factor of 16 is the default; a scale factor of 32 doubles the size of the text. Details of how the angle and scale affect the drawing of text are given below.

The TEXTDN char,veclst command defines a second font (font 2) for text drawing. Font 2 is used by the TEXT2 and VTEXT2 commands. char gives the character to be defined. veclst defines the vectors that will draw the character. Details of the TEXTDN command are given in the Command Reference.

The TEXTRE command erases the definition of font 2 and allows a new font 2 to be defined if desired.

The four commands TEXT1, TEXT2, VTEXT1, and VTEXT2 are all used for drawing text. The commands TEXT1 string and TEXT2 string use font 1 and font 2 to draw text into image memory, at the angle and scale factor specified by the TEXTC command. If a particular character in font 2 has not been defined, the corresponding character in font 1 is used.

TEXT1 and TEXT2 draw horizontal text. Angles are calculated from the positive X axis.

The commands VTEXT1 string and VTEXT2 string also use fonts 1 and 2 to draw text into image memory, at the angle and scale factor given by TEXTC. The text is drawn vertically, starting at the current point and writing down. Angles are calculated from the negative Y axis counterclockwise; an angle of 90 degrees draws side-ways text along the X axis.

Figure 2 shows the various text angles and the differences betwen vertical and horizontal text.

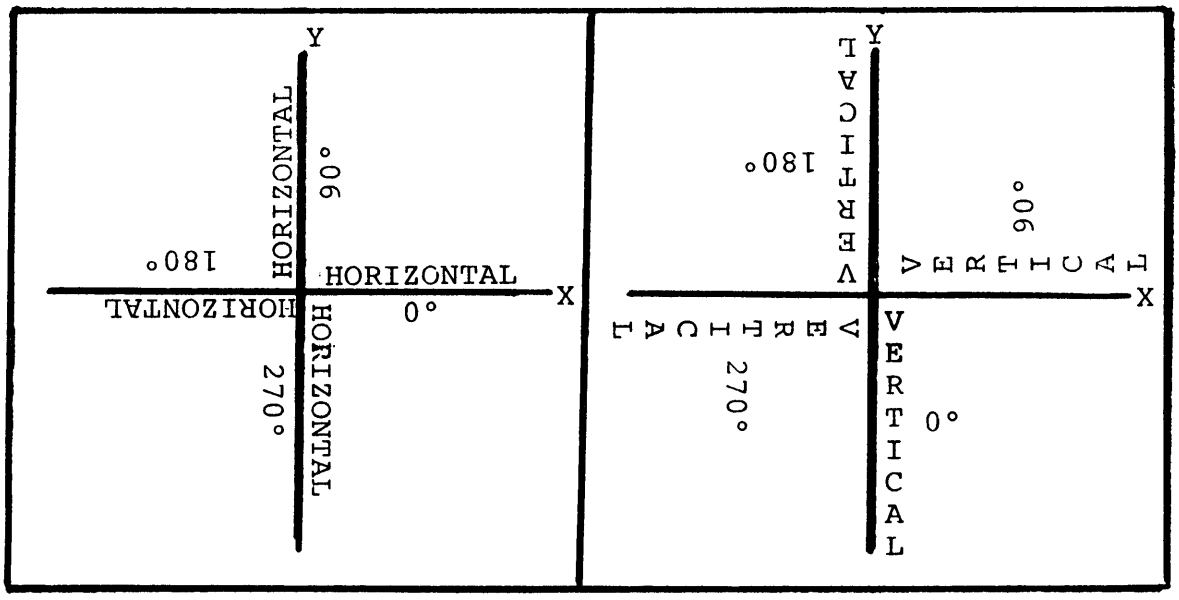

Figure 2 Horizontal and Vertical Text

Each text string is drawn with its lower-left corner placed at the current point. If the current scale factor is 16 (the default), each character uses five pixels horizontally and seven pixels vertically (excluding descenders, which take an additional two pixels).

When a TEXT command is sent from the host computer, it includes a one-byte parameter which indicates the number of characters to be drawn, followed by the 7-bit ASCII codes for each of the characters. All TEXT commands may be issued from the local terminal without counting the number of characters to be drawn. Each character after the command and delimiting blank is assumed to be part of the text string. For example:

```
TEXT1 Draw this text
```

will draw the text "Draw this text" horizontally from the current point.

```
VTEXT1 Draw vertical text
```

will draw "Draw vertical text" down from the current point.

7.0 Filled Primitives

Circles, arcs, rectangles, and polygons may be drawn filled or unfilled, using the PRMFIL flag command. When the command PRMFIL 1 or PRMFIL ON is issued, any subsequent graphics commands will draw filled graphics primitives. When a primitive is filled, the current pixel value is used for all interior pixels, as well as for the border. With PRMFIL OFF, the border only is drawn in the current pixel value.

The default setting is PRMFIL OFF.

The SETPAT pattern and PATFIL flag commands allow for patterned area and primitive fills in preset patterns. PATFIL flag enables (flag=1 or ON) and disables (flag=0 or OFF) patterned area filling; SETPAT pattern determines the pattern. pattern is a sixteen-bit pattern converted to a four-by-four repeating pattern for filling.

8.0 Seeded Area Fills

In addition to filling of graphics primitives, the Model One also supports two area fill commands. In a seeded area fill, a seed point and a boundary condition are given. The seed point designates the interior of the area to be filled; the boundary condition tells the Model One the conditions which define the boundary of the area to be filled.

The AREA1 command uses the current point as its seed point. The boundary of the region is defined as any pixel whose value is different from the current pixel value. For example:

```
VALUE 0,0,0         Set current pixel value to black (24-bit system)
VAL1K 0             Set pixel value to black (8-bit system)
CLEAR
PRMFIL OFF
VALUE 255,0,0       Red outlined shapes (24-bit system)
VAL1K 48            Red shapes (8-bit system)
MOVABS 0,0
CIRCLE 25           Draws a circle of radius 25, center (0,0)
VALUE 0,0,255       Value is now blue (24-bit system)
VAL1K 3             Blue (8-bit system)
```

AREA1

The commands above will draw a red unfilled circle. When the AREA1 command is executed, the circle is filled with blue, creating a blue disk with a red edge.

The AREA2 vreg command also uses the current point as its seed point. However, AREA2 does not stop filling an area until it encounters a pixel whose value is equal to the value stored in vreg or a pixel whose value is equal to the value stored in vreg 0. For example, for an 8-bit system:

CLEAR	
VAL8 48	Set current pixel value to 48 (default LUT value 255,0,0: red).
CIRCLE 50	Draw a red circle.
VAL8 12	Set current pixel value to 12 (default LUT value 0,255,0: green).
RECTAN 50,50	Draw a green rectangle.
VLOAD 8 12,0,0	Load value register 8 with the value in LUT index 12 (green).
VAL8 60	Set current pixel value to 60 (default LUT value 255,255,0: yellow).
MOVABS 1,1	Move into circle/rectangle overlap.
AREA1	The arc defined by the rectangle will be filled with yellow.
MOVABS 1,1	Move into circle/rectangle overlap.
VAL8 51	Set current pixel value to 51 (default LUT value 255,0,255: magenta).
AREA2	Fill the entire rectangle with magenta.

Whenever an area fill is done, using either AREA1 or AREA2, the fill mask, specified by the FILMSK rmsk,gmsk,bmsk command, is ANDed with all pixel values read from image memory. Then the resultant pixel values are tested to see if they meet the prescribed boundary condition. The fill mask allows the boundary pixel values to lie in a different bit plane or image memory bank while performing the area fill in another bit plane or memory bank. The fill mask value is stored in VREG 3; the default value is 255,255,255.

An example illustrating this is shown on the next page.

```
PRMFIL ON              Select filled primitives.
LUT8 1 255 0 0         Set color out for LUT index 1 to red.
LUT8 2 0 255 0         Set color out for LUT index 2 to green.
LUT8 3 0 0 255         Set color out for LUT index 3 to blue.
LUT8 4 255 255 0       Set color out for LUT index 4 to yellow.
LUT8 5 255 255 255     Set color out for LUT index 5 to white.
LUT8 6 255 255 0       Set color out for LUT index 6 to yellow.

MOVABS 0 0             Move current point to 0,0.
VAL8 1                 Set current pixel value to 1 (red).
CIRCLE 100             Draw a filled red circle.

MOVABS -50 0           Move current point to -50,0.
VAL8 2                 Set current pixel value to 2 (green).
WMSK16 #0200           Write enable only bit plane 1.
CIRCLE 100             Draw a filled green circle.  The intersection
                       of this circle and the red circle is blue, since
                       bit plane 0 is write protected.
MOVABS 0 75            Move current point to 0,75: in the intersection
                       of the two circles.
VAL8 4                 Set current pixel value to 4 (yellow).
WMSK16 #0500           Write enable only bit planes 0 and 2.
FILMSK 1,0,0           Check data on bit plane 1 only for area fill
                       tests.
AREA1                  Perform area fill in yellow.  Fill the overlapped
                       region and the partial red circle.  Do not fill
                       the partial green circle, because the fill mask
                       is ANDed with the green boundary value before
                       the comparison is made.
```

RASTER TECHNOLOGIES
MODEL ONE/80
DATA READ-BACK AND IMAGE TRANSMISSION

Revision 1.0 February 27, 1984

MODEL ONE DATA READ-BACK AND IMAGE TRANSMISSION
February 27, 1984

NOTICE:

The information contained in this document is subject to change without notice.

RASTER TECHNOLOGIES DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS MATERIAL (INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE), EITHER EXPRESS OR IMPLIED. RASTER TECHNOLOGIES SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM ANY ERROR CONTAINED HEREIN, INCLUDING, BUT NOT LIMITED TO, FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS MATERIAL.

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual may cause interference with regard to radio commmunications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Table of Contents

DATA READ-BACK AND IMAGE TRANSMISSION

This manual describes the Model One commands which are used to load image memory with data from the host computer, and also describes the commands which read data from the Model One back to the host computer.

1.0 Reading Back Information to the Host Computer

The Model One supports these commands for reading back data from the Model One to the host computer: RDKEYB, READER, RDMODE, READP, READCR, READVR, READW, READWE, READBU, and READF.

The RDKEYB command reads back the keyboard buffer. It sends data in binary format only, regardless of the setting of the RDMODE command. Details are given in the Command Reference.

The RDMODE and READF commands set the readback formats for the other READ commands; they are discussed at greater length below.

Whenever any of the readback commands is issued and data is sent by the Model One to the host computer, the host must acknowledge receipt of the transmitted data by sending a 7-bit ASCII ACK character (06H or 86H) back to the Model One. Any data sent from the host before the ACK is received is ignored by the Model One.

The host acknowledge character must be sent as a 7-bit ASCII ACK, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hexadecimal format.

This handshaking protocol ensures proper operation of the Model One when communicating with the host over a full-duplex serial communication line. A full duplex host computer will echo back the characters sent by the Model One; ignoring characters sent by the host until the ACK is sent alleviates this problem.

When a READ command is issued, the data requested is sent back to the port that requested the information.

When the READ command is issued from the local alphanumeric terminal, no acknowledgement is expected or required.

The RDMODE flag command sets the format for reading back data to the host computer. The default format—RDMODE 0--is ASCII decimal. RDMODE 1 selects binary format. ASCII decimal readbacks are used for serial communications. Binary readbacks are used for DMA communications.

When ASCII decimal data is sent, it is followed by a carriage return. When binary data is sent, no carriage return is included.

The READP command reads the pixel value of the current point and sends that value to whichever port (host or alphanumeric terminal) is in graphics mode. The value is sent as three ASCII decimal numbers, representing the red, green, and blue pixel values. When the pixel value is sent to the host, three three-digit integers (FORTRAN 3I3 format) are sent, if the RDMODE is 0. If

RDMODE=1, the data is returned as three binary bytes.

The READCR creg and READVR vreg commands read back the contents of the coordinate registers and value registers. The READCR command returns two 6-digit integers (2I6), representing the X and Y cooordinates; the READVR command returns three 3-digit integers (3I3) representing the red, green, and blue pixel values. Both of these commands will return binary bytes (4 bytes for READCR and 3 bytes for READVR) if the RDMODE is set to 1. For example, if CREG 23 holds the point (10,-10), the command READCR 23 returns 00010 -00010 (RDMODE 0).

The READF func command specifies the format of the pixel data that is sent to the host when a READW or READWE command is executed. (The READW and READWE commands are described in the next few paragraphs.) func=0 tells the Model One to send full 24 bit per pixel values--red, green, and blue. The bytes are sent in FORTRAN 3I3 format (RDMODE 0) or as three binary bytes (RDMODE 1). func=1, 2, or 3 selects data from a single image memory bank (red=1, green=2, blue=3) and sends the data in I3 format or as a single binary byte. func=4 instructs the Model One to send data in the same packed RGB format used to send data to the Model One in the VALlK command; the data is sent in I3 format or as a single binary byte to the host computer.

NOTE: READF 4 functions differently depending on the RGBTRU mode: with RGBTRU ON, the two high bits of each byte of the value are packed into a 6-bit value which is padded with two bits of zero. With RGBTRU OFF, the first byte is sent; the mode is identical to READF 1.

The READW nrows,ncols bf command instructs the Model One to read back a rectangular array of pixels. nrows and ncols specify the number of rows and columns to be read. bf, the blocking factor, specifies the number of pixels to be sent back to the host before waiting for the ACKnowledge character to be sent to the Model One from the host. If the end of the window is reached before the block is full, it is padded with zeroes and sent.

If, after the READW command has been issued and before it completes, the host wants the Model One to discontinue sending data blocks, the host can send the NACK (negative acknowledge) character instead of an ACK. The NACK character must be sent as a seven-bit ASCII NACK (15H or 95H); the character may be changed with the SPCHAR command. Once the NACK is sent, the command is completely aborted--in fact, the Model One leaves graphics mode.

The READW block factor, bf, prevents host input buffer overflows.

Once the window is defined by nrows and ncols, the READW command sends the pixels in the window from left-to-right and top-to-bottom. The current point is used to specify the upper-left corner of the window. For example, to read back a 512x512 square area of image memory, the current point is moved to (-256,255) Then, the command READW 512 512 bf, with the correct blocking factor, is given.

The READW command sends data to the host computer in the same format that the PIXELS and PIXEL8 commands use to send data to the Model One.

The READWE nrows,ncols bf command is a run-length encoded form of the READW command. Like the READW command, the READWE command is aborted if the host computer sends a NACK character. When the READWE command is executed, it sends back data in a form that can later be transmitted to the Model One by the RUNLEN and RUNLN8 commands: it includes a pixel value (r,g,b for RUNLEN and val for RUNLN8) and a count. The count is from 0 to 255, and indicates one less than the actual number of pixels set to that value. (Note: the count is done this way to allow 256 pixels to be set at once, if appropriate.) The count is sent in FORTRAN I3 format or as a single binary byte. As in READW, the blocking factor, bf, specifies the number of pixels to be sent before waiting for an ACK. Again, if the end of window is reached before the block is full, the block is padded with zeroes and sent.

The READBU flag,cflg command allows the host computer to determine which function buttons have been pressed at the user's workstation. Whenever a function button is pressed, the Model One makes an entry in the function button event queue; this entry includes the button number and the digitizer or joystick coordinates. The function button event queue is eight events deep. The READBU command removes one entry from the event queue and sends the data to the host. The event queue is first-in, first-out (FIFO): if more than one button is pressed, the first button is sent to the host.

The flag parameter of the READBU command indicates whether the Model One should respond immediately.

If flag=1, the Model One will wait until there is an entry in the button queue and then send the data; if the button queue is not empty, the Model One sends the data immediately. if flag=0, the Model One responds immediately: if the queue is empty, a button number of zero is sent; if there is data in the queue, the data is sent. The button number is returned as a three-digit integer (I3) or as a single binary byte. The (x,y) coordinates--whether from the digitizer or joystick-- are returned as FORTRAN 2I6 or as four binary bytes.

The cflg parameter of the READBU command indicates whether the digitizer coordinate (cflg=0) or the joystick coordinate (cflg=1) should be returned.

If, for example, the user presses button 4 twice, and you execute this command sequence, you will see:

```
READBU 1 0
 004  00010 -00015
READBU 0 0
 004  00010 -00015
READBU 1 0
```

and no value will be returned from the third READBU command until another button is pressed.

The READER command can be used to determine whether an error has occrred. READER returns a single byte (either binary or FORTRAN I3 format) giving the number of the first error since the last READER or COLDstart command. Zero will be returned if no error has occurred.

2.0 Image Transmission

Four commands are available to load a rectangular array of image memory pixels with arbitrary data specified by the host computer. PIXELS and PIXEL8 send data to the Model One in a pixel-by-pixel form. RUNLEN and RUNLN8 send pixel data in a run-length encoded format, in which each pixel value is followed by a one-byte count specifying the number of pixels (horizontally) to be set to the specified pixel value.

The transmission time for some images is reduced by using the run-length transmission format; other images may require more time to transmit because of the overhead involved in sending the one-byte count along with each pixel value. The easiest way to determine which format is more rapid for a particular type of image is to compare the required transmission time for sample images in each of the two possible formats.

The PIXELS nrows,cols r,g,b ... r,g,b command transmits 24 bits of pixel value data per pixel to fill a specified rectangle. nrows and ncols specify the number of rows--the height-- and the number of columns--the width--of the rectangular array of pixels which is to be filled. For each pixel in the rectangle, one byte each of red, green, and blue pixel value data is sent, starting at the upper-left corner and working left-to-right and top-to-bottom. The upper-left corner of the rectangle begins at the current point (CREG 0).

For example, to fill a 512x512 square of image memory, the current point is moved to (-256,255) and the command PIXELS 512 512 given, followed by 3x512x512 bytes of image memory data, used to fill the 512x512 rectangle with red, green, and blue pixel values.

The PIXELS command is analogous to issuing a series of VALUE, POINT, and MOVREL 1,0 commands to fill image memory. (Of course, the analogy breaks down at the end of the scan line, when you would have to move explicitly to the next line.)

The PIXEL8 nrows,ncols val ... val command also defines a rectangular array of pixels in the Model One's image memory. The pixel value data is given as a single one-byte value, as in the VAL8 command. The PIXEL8 command functions in the same way as the PIXELS command: nrows and ncols define the rectangle; the series of val is used to fill the rectangle with pixel values.

The PIXEL8 command is most often used to transmit 8 bit-per-pixel pseudo-color images; it is analogous to a series of VAL8, POINT, and MOVREL 1,0 commands.

The RUNLEN nrows,ncols r,g,b count command is a run-length encoded form of the PIXELS command. nrows and ncols again define the rectangle for the pixel data. r,g,b and count define the pixel value and the number of horizontal pixels to be set to the given r,g,b pixel value. count gives one less than the number of adjacent pixels to be set: for example, if count=0, one pixel is set; if count=1, two pixels are set. The maximum count of 255 sets 256 pixels to the r,g,b pixel value.

The RUNLN8 nrows,ncols val count command is the 8-bit form of RUNLEN. The pixel value is interpreted as in the VAL8 command. Like the PIXEL8 command, RUNLN8 is used in applications using less than 24 bits per pixel.

RASTER TECHNOLOGIES
MODEL ONE/80
MACRO PROGRAMMING

Revision 1.0 February 27, 1984

MODEL ONE MACRO PROGRAMMING
February 27, 1984

NOTICE:

The information contained in this document is subject to change without notice.

RASTER TECHNOLOGIES DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS MATERIAL (INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE), EITHER EXPRESS OR IMPLIED. RASTER TECHNOLOGIES SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM ANY ERROR CONTAINED HEREIN, INCLUDING, BUT NOT LIMITED TO, FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS MATERIAL.

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual may cause interference with regard to radio commmunications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.