RASTER TECHNOLOGIES
MODEL ONE
ALPHANUMERIC TERMINAL EMULATION
PROGRAMMING GUIDE

Raster Technologies

RASTER TECHNOLOGIES
MODEL ONE
ALPHANUMERIC TERMINAL EMULATION
PROGRAMMING GUIDE

Revision 2.0 June 24, 1983

ALPHANUMERIC TERMINAL EMULATION PROGRAMMING GUIDE
June 24, 1983

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.

Table of Contents

1.0 INTRODUCTION

This manual describes the alphanumeric terminal emulation commands. These commands use the Model One monitor as an alphanumeric terminal, thus allowing a Model One workstation to be operated with the Model One, a local keyboard, and a monitor. Under these circumstances, no local terminal is needed.

This manual includes 7 sections, as described below.

Section 2.0 Alphanumeric Terminal Emulation Commands

This section describes this command:

ALPHEM Enables or disables terminal emulation; routes text from the keyboard and host to the selected window.

Section 3.0 Window Definition, Selection, and Deletion

This section describes the commands to set up the scrolling text windows as desired.

DEFWIN Defines the size and position of up to 9 scrolling text windows.

SELWIN Selects a previously defined window.

DELWIN Deletes a previously defined window.

GETWIN Returns the number of the currently selected window.

Section 4.0 Moving Around the Window

This section describes the commands for moving around the selected text window:

DIRCUR Moves the cursor to a specified character position.

GETPOS Returns the character position of the cursor.

MOVCUR Moves the cursor to a specified Model One coordinate position.

GETCUR Returns the Model One coordinate position of the cursor.

HOME Moves the cursor to character position (0,0).

SETCUR Enables or disables drawing of the cursor.

Section 5.0 Window and Text Characteristics

This section describes the commands used to define the window attributes, such as color and text size:

BOLD Enables or disables drawing of bold text.

OVRSTK Enables or disables overstriking of text.

SCROLL Enables or disables text scrolling.

SETSIZ Sets x and y scaling of text.

WRAP Enables or disables text wraparound.

Section 6.0 Error Messages

This section lists the possible error messages.

Section 7.0 Alphabetical Command Reference

This section provides complete details of all commands associated with the alphanumeric windows.

2.0 ALPHANUMERIC TERMINAL EMULATION COMMANDS

A single command is used to start and stop the alphanumeric terminal emulator; it is described in this section. Use of the windows is described in detail in the following sections. When the Model One is factory-configured for a keyboard attached to the KEYBSIO port, the Model One coldstarts with window 0 defined and selected; all window attributes, such as bolding and text size, are set to their defaults. No other windows are defined. Any text that is typed at the keyboard will go to the text window.

The command ALPHEM ON is also executed during coldstart. This instructs the Model One to send all text from the keyboard to the window; if you wish to turn off the alphanumeric text emulation environment, you can use the command ALPHEM 0 or ALPHEM OFF.

3.0 WINDOW DEFINITION, SELECTION, AND DELETION

Up to nine scrolling text windows may be defined and used for alphanumeric terminal emulation. Window 0 is the hardware-scrolled window; it always uses the entire monitor area. The screen origin register (CREG 4) is used for scrolling, so that changes to CREG 4 will also affect window 0.

Up to eight software-scrolled windows--windows 1 through 8--may also be defined and used for alphanumeric text. These windows may be any size.

Each alphanumeric window is defined by number. When the window is defined, two diagonal corners are specified, thus giving the window size. (Of course, window 0 always occupies the entire screen, and so these corners are ignored for window 0.) The x and y scaling of the text for the window can also be defined.

Finally, each window has an associated write mask, which is specified the same way as with the WRMASK command.

The DEFWIN window,xl,yl,xl,y2,x size,y size,bitm,bankm command defines the text window specified by window. Any parameter with a value of zero (0) instructs the DEFWIN command to use the default or existing value. (There is no default for the window corners.) (This is shown in the examples below.) xl,yl and x2,y2 specify the corners of the text window. Windows 1 through 8 can be any size; window 0 will always use the entire screen. x size and y size specify the text size. x_size and y_size are normally set to 1,1 (2,2 for the Model One/40 and the Model One/25 in 1K mode); this is equivalent to specifying a text size of 16 with the TEXTC command. x_size and y_size of 2,2 are equivalent to a text size of 32 with the TEXTC command. x_size and y_size may be different, allowing tall skinny text or short fat text.

bitm and bankm specify the write mask and are used exactly as they are used in the WRMASK command. Note: because a value of zero for any parameter instructs the DEFWIN command to use the default or existing value, it is impossible to specify a write mask of 0,0. This means, of course, that the write mask can never be set so that no information can be written in the alphanumeric windows.

Each window has a foreground, background, and cursor color. A set of three value registers is used for each window to specify these colors. Window 0 uses value registers 16, 17, and 18; window 1 uses value registers 19, 20, and 21; and so on through window 8, which uses value registers 40, 41, and 42.

The foreground color is used for text; the background color occupies the rest of the window. The cursor is an XOR cursor, so that the color in the value register is XORed with the background color to display the cursor. For example, if the background color is red and the foreground color is black, the cursor color value register must be set to cyan for the user to see a white cursor.

For example, to define window 1 to occupy the lower area of the monitor, use all bit planes and banks for text, and have a normal text size, you would use the command:

```
DEFWIN 1 -512,-512 512,-450 2,2 0,0
```

for a Model One/40 or a Model One/25 in 1K mode. For a Model One/60, you could use this command:

```
DEFWIN 1 -389,-288 389,-240 2,2 0,0
```

Then, to set up the desired colors for background, foreground and cursor, you would set value registers 19, 20, and 21.

Now, if you wish to redefine the area of window 1, you can use the command:

```
DEFWIN 1 -100,-100 -50,-50 0,0 0,0
```

which redefines the area of the window, and sets the text size and write mask back to the default values.

Once a window has been defined, it must be selected before information can be written to the window. Only one window can be selected at a time, and selection of the window "unselects" the previously selected window. The command SELWIN window selects a window; for example, SELWIN 2 selects window 2.

DELWIN window deletes the definition of a window. The window to be deleted cannot be selected.

GETWIN returns the number of the selected window. -1 is returned if no window is selected.

4.0 MOVING AROUND THE WINDOW

The alphanumeric text windows include a cursor. This cursor can be moved to any character position--row and column--or to a Model One coordinate position. For windows 1 through 8, the Model One coordinate position is rounded to the nearest character position; in window 0, the coordinate position is not rounded.

The commands DIRCUR x,y and HOME are used to move the cursor to character positions. DIRCUR will move the cursor to an arbitrary position; if it is outside the window, it will move the cursor as far as possible in the right direction. HOME always moves the cursor to character position (0,0)--the first row and first column. GETPOS returns the character position of the cursor.

MOVCUR moves the cursor to a Model One coordinate position; GETCUR returns the Model One coordinate position of the cursor.

SETCUR enables or disables the cursor.

5.0 WINDOW AND TEXT CHARACTERISTICS

In using the scrolling text windows, the boldness of the text, the size of the text, whether the text can be overstruck, and whether the text will wrap or scroll can all be set as desired.

BOLD ON or BOLD 1 turns on bolding of text. The boldness of text is in proportion to its size, as determined by the SETSIZ or DEFWIN commands. Bold text is the default. BOLD OFF or BOLD 0 disables bolding of text.

SETSIZ xscale,yscale specifies the x and y scaling of text. The default is 1,1. The new size overrides whatever was defined by the DEFWIN command; for windows 1 through 8, the entire window is re-initialized and any text currently in the window is lost. If either xscale or yscale is zero, the current value will be used.

OVRSTK ON or OVRSTK 1 enables overstriking of characters on the line. Previous characters in the line may then be overstruck without erasing that character. OVRSTK OFF or OVRSTK 0 disables overstriking; this is the default. For windows 1 through 8, only the last character in a given position will remain after the window is scrolled.

SCROLL ON and WRAP ON enables scrolling and wraparound of text. If scrolling is off--SCROLL OFF or SCROLL 0--characters wrap from the end of the bottom line to the beginning of the top line. If wraparound is off, the cursor stays in the last character position on the line until a carriage return in typed. The default is scrolling and wraparound of text.

In addition to these commands, a set of control characters can be used to move around the window and perform line functions. These are:

[CTRL-B]	Clear entire current line
[CTRL-E]	Move cursor up one pixel (window 0 only)
[CTRL-H]	Backspace
[CTRL-I]	Horizontal tab to next multiple of 8 characters
[CTRL-J]	Linefeed
[CTRL-K]	Vertical tab one line (upward linefeed)
[CTRL-L]	Formfeed: clears entire window
[CTRL-M]	Carriage return
[CTRL-T]	Clear to end of line
[CTRL-V]	Move cursor down one pixel (window 0 only)
[CTRL-Y]	Move cursor left one pixel (window 0 only)
[CTRL-\]	Home (move to character position 0,0)
[CTRL-]]	Move cursor right one character position
[CTRL-_]	Move cursor right one pixel (window 0 only)

6.0 ERROR MESSAGES

These error messages are generated by the alphanumeric terminal emulation commands:

7. Illegal parameter: this message indicates that an illegal value for a parameter has been entered. This message will be given for a bad write mask in DEFWIN.

17. Not enough space for definition: this message will be given by DEFWIN or SETSIZ if all available window buffer space has been used. Delete unneeded windows before proceeding.

72. Window not yet defined: this message will be given by SELWIN if the user attempts to select a window that has not been defined by DEFWIN.

73. Bad character size: this message will be given by SETSIZ if an invalid size has been entered.

74. Bad window size: this message will be given by DEFWIN if an invalid size has been given by the two window corners.

75. Illegal window number: this message will be given by DEFWIN, SELWIN, and DELWIN if an invalid window number (not 0 through 8) has been entered.

76. No window selected: this message will be given by the commands ALPHEM, BOLD, DIRCUR, GETCUR, GETPOS, HOME, MOVCUR, OVRSTK, SCROLL, SETCUR, SETSIZ, and WRAP if no window has been selected and an attempt is made to execute one of these commands.

78. Cannot delete selected window: this message will be given by DELWIN if the user tries to delete the currently selected window.

7.0 ALPHABETICAL COMMAND REFERENCE

This section lists, one command to a page, every Model One alphanumeric windowing command. These subsections are included for each command:

SYNTAX: this section gives the command syntax.

FUNCTION: this section describes the function of the command.

PARAMETERS: this section lists the parameters for the command, gives their ranges, and supplies any other needed information on the command parameters.

HOST BINARY COMMAND STREAM: this section gives the host binary command stream and the hexadecimal, octal, and decimal opcodes for the command.

FORTRAN CALL: this section describes the FORTRAN call for the command, including the variable names.

EXAMPLE: this section gives an example of how the command is used.

RELATED COMMANDS: this section lists any related commands.

The following pages present the Model One commands in alphabetical order.

A L P H E M

SYNTAX

ALPHEM flag

FUNCTION

The ALPHEM command turns the alphanumeric terminal emulator on or off. When the alphanumeric terminal emulator is on, the selected window of the display monitor is used as a scrolling text window. Any characters typed at the Raster Technologies keyboard are sent to the window. The window must be defined (see DEFWIN) and selected (SELWIN) before it can be used.

PARAMETERS

flag flag=1 or ON enables alphanumeric terminal emulator; flag=0 or OFF disables alphanumeric terminal emulator

HOST BINARY COMMAND STREAM

[C2H] [flag] (2 bytes)
C2H=302(octal)=194(decimal)

FORTRAN SUBROUTINE CALL

CALL ALPHEM (FLAG)

EXAMPLE

```
DEFWIN 2 110,110 220,220 1,1 0,0
                         Defines window 2 with default write mask
SELWIN 2                 Selects window 2
ALPHEM ON                Turn on alphanumeric terminal emulator.
                         Subsequent characters typed at the local
                         keyboard are sent to window 2
```

RELATED COMMANDS

all alphanumeric terminal emulation commands

B O L D

SYNTAX

BOLD flag

FUNCTION

The BOLD command controls the drawing of text. BOLD ON or BOLD 1 (the default) draws bold text, in proportion to the scaling factor. BOLD OFF or BOLD 0 draws normal (thin) text.

PARAMETERS

flag flag=1 or ON draws bold text; flag=0 or OFF draws normal text

HOST BINARY COMMAND STREAM

[CCH] [flag] (2 bytes)
CCH=314(octal)=204(decimal)

FORTRAN SUBROUTINE CALL

CALL BOLD (flag)

EXAMPLE

BOLD ON Enables drawing of bold text

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

D E F W I N

SYNTAX

DEFWIN window,x1,y1,x2,y2,x_size,y_size,bitm,bankm

FUNCTION

The DEFWIN command is used to define the parameters for the alphanumeric windows: the corners, the size of the text within the window, and the writemask. window gives the window number. window=0 specifies the hardware-scrolled window, which uses the entire screen of the monitor. This window uses the screen origin register for scrolling; changes to creg 4 will affect window 0. In addition, eight other windows are available (window=1 to 8). These windows are software-scrolled, and may be any size.

The corners are defined by the Model One coordinate pair (x1,y1) and (x2,y2), where the two points specify diagonally opposite corners. For window 0, these corners are ignored, as the window always occupies the full screen. Windows may overlap and the selected window (for windows 1 through 8) will always be on top of the other windows.

x size and y size specify the x and y scaling of all text for that window. The scale is equivalent to that specified with the SETSIZ command.

bitm and bankm set the write mask for the specified window. Details of these parameters can be found in the WRMASK command. The default write mask is all bit planes of all banks.

If the size or write mask parameters have a value of zero, the default value is used.

Three value registers are associated with each window: the first specifies the text color, the second the background color, and the third the cursor color. Note that this cursor color is then XORed with the background color to produce the color that is actually seen; for example, if the background is red and the user wants a white cursor, the value register for the cursor color would hold cyan. Window 0 uses value registers 16, 17, and 18; window 1 uses value registers 19, 20, and 21; and so on through window 8, which uses value registers 40, 41, and 42.

For windows 1 through 8, any changes to attributes of the window do not affect the entire window until the window is scrolled. However, changes to the attributes will take place on the current line immediately. For example, a change in the value register for the background color will not change the window until the window is scrolled.

PARAMETERS

window	the window number; the range is 0 to 8.
x1,y1	the first corner of the window (ignored for window 0); must be given in Model One coordinates.
x2,y2	the diagonally opposite corner of the window (ignored

for window 0); must be given in Model One coordinates.

x_size,y_size the x and y scaling of text within the window; range is from 1,1 (where each character uses a 16-pixel square). The default is 1,1 for a 512x512 or 768x576 system, and 2,2 for a 1024x1024 system (including 1K mode).

bitm,bankm specify the write mask, as detailed in the WRMASK command.

HOST BINARY COMMAND STREAM

[C0H][window][highx1][lowx1][highy1][lowy1][highx2][lowx2][highy2][lowy2]
[x_size][y_size][bitm][bankm] (14 bytes)
C0H=300(octal)=192(decimal)

FORTRAN SUBROUTINE CALL

```
CALL DEFWIN (WINDOW,X1,Y1,X2,Y2,X_SIZE,Y_SIZE,BITM,BANKM)
```

EXAMPLE

```
DEFWIN 1 20,20 50,50 1,1 0,0
```

This command defines window 1 with a lower-left corner of (20,20), an upper right corner of (50,50), a text size of 1,1 and the default write mask of all bit planes and banks.

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

D E L W I N

SYNTAX

DELWIN window

FUNCTION

The DELWIN command deletes a previously defined window, freeing the buffer space for use. The currently active window cannot be deleted.

PARAMETERS

window the window number, from 0 to 8

HOST BINARY COMMAND STREAM

[C3H] [window] (2 bytes)
C3H=303(octal)=195(hexadecimal)

FORTRAN SUBROUTINE CALL

CALL DELWIN (WINDOW)

EXAMPLE

```
SELWIN 2     Select window 2
DELWIN 3     Deletes window 3
DELWIN 2     Window 2 cannot be deleted; it is selected
Error 075
DELWIN: Illegal window number
```

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

D I R C U R

SYNTAX

DIRCUR x,y

FUNCTION

The DIRCUR command moves the cursor to character position (x,y) within the alphanumeric window. The character positions are defined as (0,0) in the upper-left corner of the window, moving positive in x and y toward the lower-right corner, as shown:

Note that the first row is row 0 and the first column in column 0.

If an attempt is made to move the cursor beyond the window boundaries, the cursor will be moved to the edge of the window but not beyond. The largest character position on the Model One/20 (assuming a full-screen window) is (84,50); for the Model One/40 and Model One/60, it is (169,101).

PARAMETERS

x,y x,y specify the character position within the window.

HOST BINARY COMMAND STREAM

[C4H][x][y] (3 bytes)
C4H=304(octal)=196(decimal)

FORTRAN SUBROUTINE CALL

CALL DIRCUR (X,Y)

EXAMPLE

DIRCUR 3,4 Moves the cursor to character position (3,4), which is the fourth row, fifth column.

RELATED COMMANDS

all alphanumeric terminal emulation commands

G E T C U R

SYNTAX

GETCUR

FUNCTION

The GETCUR command returns the Model One coordinate position of the cursor within the currently selected window. Note that the Model One coordinate position is rounded to the nearest character position for windows 1 through 8.

HOST BINARY COMMAND STREAM

[C9H] (1 byte)
C9H=311(octal)=201(decimal)

FORTRAN SUBROUTINE CALL

CALL GETCUR (IX,IY)

EXAMPLE

```
MOVCUR 100,100      Moves cursor to coordinate position (100,100)
GETCUR              Gets cursor coordinate position
00100 00100         Current coordinate position
```

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

G E T P O S

SYNTAX

GETPOS

FUNCTION

The GETPOS command returns the character position of the cursor within the currently selected window.

HOST BINARY COMMAND STREAM

[C5H] (1 byte)
C5H=305(octal)=197(decimal)

FORTRAN SUBROUTINE CALL

CALL GETPOS (ICOL,IROW)

EXAMPLE

```
DIRCUR 3,4          Moves cursor to character position (3,4)
GETPOS              Gets cursor character position
003 004               Current character position
```

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

G E T W I N

SYNTAX

GETWIN

FUNCTION

The GETWIN command returns the number of the currently selected alphanumeric window. The value (-1) is returned if no window is active.

HOST BINARY COMMAND STREAM

[CEH] (1 byte)
CEH=316(octal)=207(decimal)

FORTRAN SUBROUTINE CALL

CALL GETWIN (NUM)

EXAMPLE

```
GETWIN          Returns number of selected window
-001          No window is active
SELWIN 7
GETWIN
007           Window 7 is now active
```

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

H O M E

SYNTAX

HOME

FUNCTION

The HOME command moves the cursor within the alphanumeric window to character position (0,0), which is the upper-left corner of the window. Note that the command DIRCUR 0,0 and HOME are equivalent.

HOST BINARY COMMAND STREAM

[CFH] (1 byte)
CFH=317(octal)=207(decimal)

FORTRAN SUBROUTINE CALL

CALL HOME

EXAMPLE

```
DIRCUR 1,1        Moves cursor to character position (1,1)
HOME              Moves cursor to character position (0,0)
```

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

M O V C U R

SYNTAX

MOVCUR x,y

FUNCTION

The MOVCUR command moves the alphanumeric window cursor to Model One coordinate (x,y), within the boundaries of the window. For windows 1 through 8, the cursor position will be rounded to the nearest character position; for window 0, the precise Model One coordinate is used.

PARAMETERS

x,y the Model One (x,y) coordinate

HOST BINARY COMMAND STREAM

[C8H] [highx] [lowx] [highy] [lowy] (5 bytes)
C8H=310(octal)=200(decimal)

FORTRAN SUBROUTINE CALL

CALL MOVCUR (X,Y)

EXAMPLE

MOVCUR 100,100 Moves the cursor to Model One coordinate (100,100)

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

O V R S T K

SYNTAX

OVRSTK flag

FUNCTION

The OVRSTK command enables or disables overstriking of text. If overstriking is enabled, previous characters in the line may be overstruck without being erased. flag=1 or ON will enable overstriking; flag=0 or OFF (the default) disables overstriking.

In windows 1 through 8, only the last character in a given position will remain after the window is scrolled.

PARAMETERS

flag flag=0 or OFF disables overstriking; flag=1 or ON enables overstriking

HOST BINARY COMMAND STREAM

[CDH][flag] (2 bytes)
CDH=315(octal)=207(decimal)

FORTRAN SUBROUTINE CALL

CALL OVRSTK (FLAG)

EXAMPLE

OVRSTK ON Enables overstriking of characters

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

S C R O L L

SYNTAX

SCROLL flag

FUNCTION

The SCROLL command controls whether the window scrolls when the text reaches the bottom of the window. If flag=1 or ON (the default), scrolling is done; if flag=0 or OFF, scrolling is not done.

When scrolling is disabled, characters wrap from the end of the bottom line to the beginning of the top line.

PARAMETERS

flag flag=1 or ON enables scrolling; flag=0 or OFF disables scrolling

HOST BINARY COMMAND STREAM

[CAH][flag] (1 byte)
CAH=312(octal)=202(decimal)

FORTRAN SUBROUTINE CALL

CALL SCROLL (FLAG)

EXAMPLE

SCROLL ON Enables scrolling of the text window

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

S E L W I N

SYNTAX

SELWIN window

FUNCTION

The SELWIN command selects the current window for use as a scrolling text window. If the alphanumeric emulator is in use, all alphanumeric data is sent to the currently selected window. The window must be defined (see DEFWIN) before it can be selected. SELWIN redraws the entire window, for windows 1 through 8, leaving the selected window on top of the other windows.

PARAMETERS

window window number: range is 0 to 8.

HOST BINARY COMMAND STREAM

[ClH] [window] (2 bytes)
ClH=301(octal)=193(decimal)

FORTRAN SUBROUTINE CALL

CALL SELWIN (WINDOW)

EXAMPLE

```
DEFWIN 7 100,100 150,150 1,1 0,0
             Defines window number 7 using the default write mask
SELWIN 7     Selects window number 7 as the current window
```

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

S E T C U R

SYNTAX

SETCUR flag

FUNCTION

The SETCUR command determines whether or not the cursor will be displayed.

PARAMETERS

flag — flag=1 or ON displays the cursor; flag=0 or OFF disables the cursor.

HOST BINARY COMMAND STREAM

[C7H][flag] (2 bytes)
C7H=307(octal)=199(decimal)

FORTRAN SUBROUTINE CALL

CALL SETCUR (FLAG)

EXAMPLE

SETCUR ON Cursor is displayed

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

S E T S I Z

SYNTAX

SETSIZ xscale,yscale

FUNCTION

The SETSIZ command sets the width and height of all text written into the alphanumeric windows. xscale determines the width (scaling in the x dimension); yscale determines the height.

If a value of zero is given for either parameter, the current value is retained. The default value for the Model One/25 in 512 mode and for the Model One/60 is (1,1); otherwise, the default is (2,2), for 1K mode and the Model One/40. An xscale and yscale of (1,1) create a 10-pixel-high by 6-pixel-wide character.

For windows 1 through 8, this command will reinitialize the entire window, erasing the text buffer and placing the cursor in character position (0,0).

PARAMETERS

xscale determines the width of the text

yscale determines the height of the text

HOST BINARY COMMAND STREAM

[C6H][xscale][yscale] (3 bytes)
C6H=306(octal)=198(decimal)

FORTRAN SUBROUTINE CALL

CALL SETSIZ (XSCALE,YSCALE)

EXAMPLE

SETSIZ 3,3 Sets the x and y text scaling to three times normal size

RELATED COMMANDS

all alphanumeric terminal emulation commands

W R A P

SYNTAX

WRAP flag

FUNCTION

The WRAP command controls whether the cursor will wrap around to the next line when the end of the line is reached. flag=1 or ON (the default) enables wraparound; flag=0 or OFF disables wraparound.

If wrap is disabled, the cursor stays in the last character position on the line until a carriage return is typed, so that the last character is overwritten repeatedly.

PARAMETERS

flag flag=1 or ON enables wraparound; flag=0 or OFF disables wraparound

HOST BINARY COMMAND STREAM

[CBH][flag] (2 bytes)
CBH=313(octal)=203(decimal)

FORTRAN SUBROUTINE CALL

CALL WRAP (FLAG)

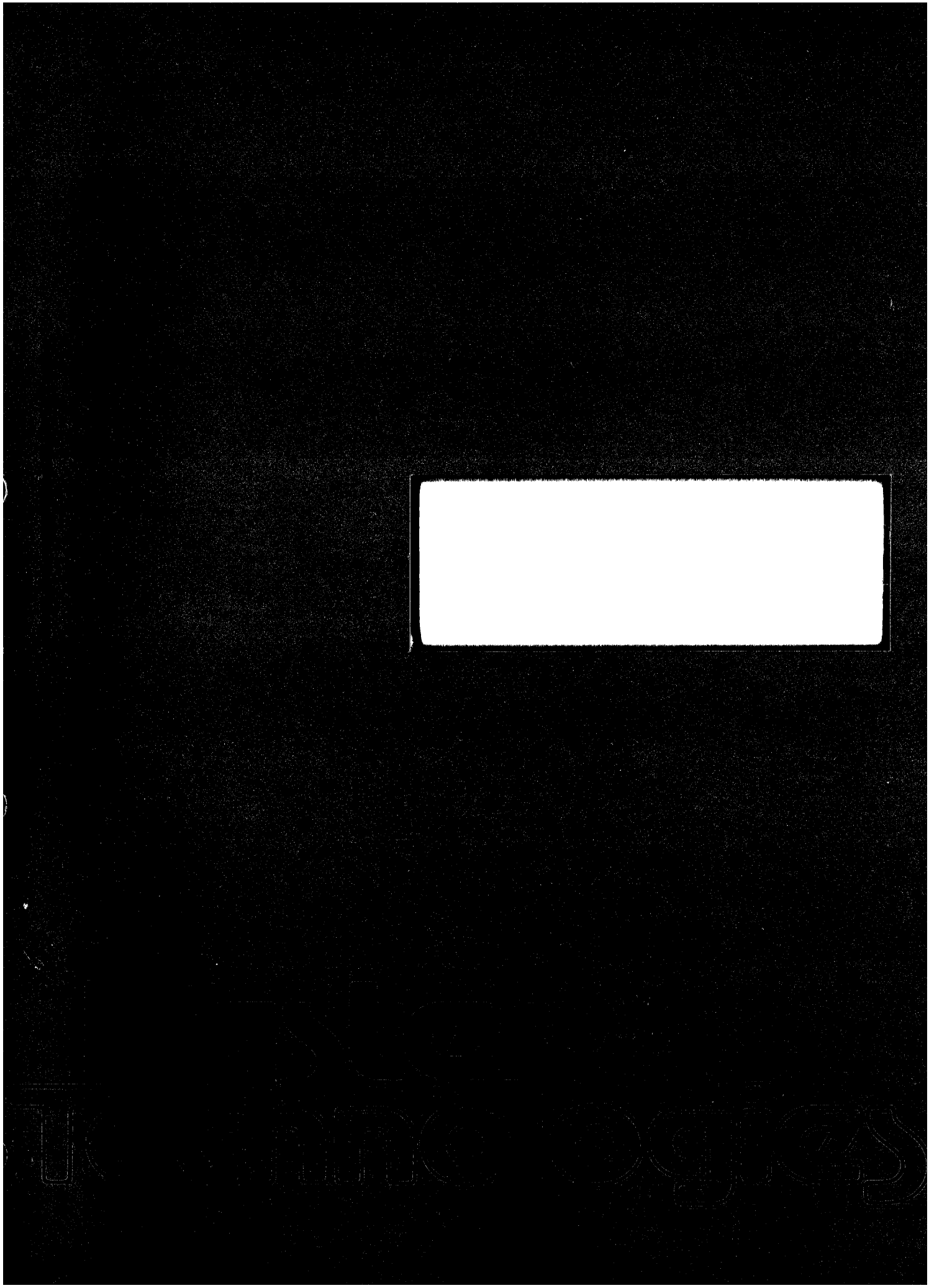
EXAMPLE

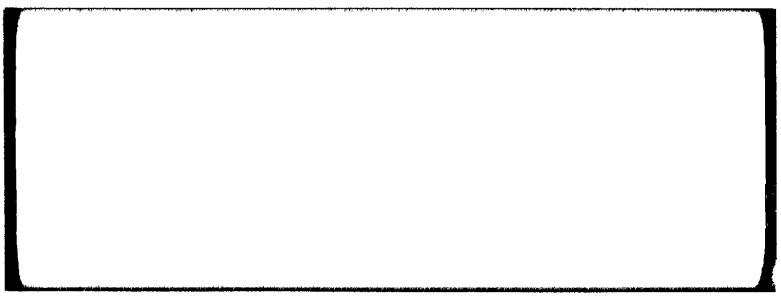
WRAP OFF Disables wrapping of the cursor when the end of the text line is reached

RELATED COMMANDS

all alphanumeric terminal emulation commands
ALPHEM
BOLD
DEFWIN
DELWIN
DIRCUR
GETCUR
GETPOS
GETWIN
HOME
MOVCUR
OVRSTK
SCROLL
SELWIN
SETCUR
SETSIZ
WRAP

8.0 INDEX





RASTER TECHNOLOGIES
MODEL ONE
DISPLAY LIST FIRMWARE
PROGRAMMING GUIDE

Revision 1.0 May 17, 1983

DISPLAY LIST FIRMWARE PROGRAMMING GUIDE
May 17, 1983

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.

Table of Contents

1.0 INTRODUCTION

This manual describes the Display List Firmware for the Model One/25, Model One/40, and Model One/60. The same Firmware can be used on any Model One; no changes are necessary to move the firmware between Model Ones.

There are seven sections in this manual:

1.0 Introduction

2.0 Creating and Updating Segments: this section describes the command used for creating and updating segments.

3.0 View Specification: this section describes and explains the DEFVW command.

4.0 Segment Attributes: this section describes the possible attributes of segments and the commands used to set those attrbutes.

5.0 Picking Graphics Primitives: this section describes the commands used to pick graphics primitives. In addition, the commands used to manipulate picked primitives are explained.

6.0 Alphabetical Command Reference: this section lists the available Display List Firmware commands in alphabetical order, providing a complete reference.

7.0 Quick Command Reference: this section lists and very briefly describes the available Display List Firmware commands.

2.0 CREATING AND UPDATING SEGMENTS

This section outlines the Display List Firmware commands used for creating and updating segments.

The command SEGINI initializes the Display List Firmware and should be used whenever an entirely new set of segments is to be defined. SYSTAT returns the Display List Firmware memory usage and availability.

2.1 The World Coordinate System

Segments are composed of graphics primitives which are defined in a 16-bit World Coordinate System ranging from (-32767,-32767) to (32766,32766). The graphics primitives are defined by the Model One command set, including MOVEs, DRAWs, CIRCLEs, ARCs, POLYGONs, and so on.

Views are defined by the DEFVW command (see section 3); views specify the portion of the World Coordinate System (WCS) which is currently visible and how that portion is displayed.

2.2 Defining Segments

A segment is a collection of graphics primitives which are stored and referenced together. The command SEGDEF begins definition of a segment; each segment may include:

1. Model One graphics primitives: MOVE commands, DRAW commands, TEXT command, CIRCLE, ARC, POLYGN, RECTAN, etc.

2. Changes to the current pixel value: VALUE, VAL8, VAL1K, etc.

3. Changes to primitive generation attributes: PRMFIL, VECPAT, etc.

To end segment definition, the SEGEND command is used.

2.3 Segment Manipulation

Once defined, a segment may be deleted (SEGDEL), renamed (SEGREN), copied (SEGCOP), or expanded (SEGAPP--segment append). To end an append to a segment, the SEGEND command is also used.

To nest a segment within another segment--up to eight levels deep-- the command SEGREF is used.

To change the current pick identification number of a segment (see section 4 for details), the SEGPID command is used.

3.0 VIEW SPECIFICATION

Up to eight views into the World Coordinate System may be defined and displayed. These are defined with the DEFVW command; the view can then be drawn with the REDRAW command.

Each view is defined by:

1. its window coordinates in the World Coordinate System

2. the viewport coordinates in the Display Coordinate System. The Display Coordinate System is defined by the monitor window: for example, the Model One/40 has a maximum display viewport from (-512,-512) to (511,511).

3. the UP vector: this vector is defined by its two endpoints, and may be at any angle. It specifies the rotation of the view as it is displayed on the screen.

4. the transformation type: four types of transformation may be done. These are: no transformation; translation and scaling; rotation only; and rotation, translation, and scaling.

5. the background color for the view.

6. the write mask for the view.

7. the highest level segment to be displayed.

The REDRAW command is used to display a window after it is defined; you can erase the viewport to the background color if desired.

4.0 SEGMENT ATTRIBUTES

Each segment has a set of attributes associated with it; these attributes may be used when picking or editing segments.

A pick identification number can be associated with primitives within a segment. The SEGPID command assigns a pick identification number which is assigned to all the graphics primitives until a SEGEND command is executed or another SEGPID command is encountered, whichever comes first.

The SETATR command is used to set the visibility or pickability of an entire segment; these are useful for the PICKCR and REDRAW commands.

The SET command is used to set the search attributes for the PICKCR command. See section 5 for details of picking.

The command SEGINQ returns information about the attributes of a given segment; the pickability and visibility are indicated.

5.0 PICKING GRAPHICS PRIMITIVES

In PICKING, an (x,y) coordinate in the Display Coordinate System is given. This coordinate is then used to find both the segment number and the graphics primitive which intersects that point. The error margin (aperture) can be set as desired.

The PICKCR command initiates the pick search. PICKCR also specifies the coordinate register for picking.

When the PICKCR command is executed, the PIDREG (pick identification number register) is updated, as is the SEGREG (segment register). The PIDREG and SEGREG are set with the value of the picked primitives. The PIDREG and SEGREG may be set with the SET command, as well. (See DELPID and HILITE for more details of how this would be used.) PICKCR uses the picking error margin (PICKAP) set with the SET command.

The HILITE and DELPID commands can be used to highlight or delete primitives with a specified pick identification number (as found in PIDREG) or segment number (SEGREG).

6.0 ALPHABETICAL COMMAND REFERENCE

This section lists, one command to a page, every Model One Display List Firmware command. These subsections are included for each command:

SYNTAX: this section gives the command syntax.

FUNCTION: this section describes the function of the command.

PARAMETERS: this section lists the parameters for the command, gives their ranges, and supplies any other needed information on the command parameters.

HOST BINARY COMMAND STREAM: this section gives the host binary command stream and the hexadecimal, octal, and decimal opcodes for the command.

FORTRAN CALL: this section describes the FORTRAN call for the command, including the variable names.

EXAMPLE: this section gives an example of how the command is used.

RELATED COMMANDS: this section lists any related commands.

The following pages present the Model One commands in alphabetical order.

D E F V W

SYNTAX

DEFVW view wcsregl,wcsreg2 dcsregl,dcsreg2 upregl,upreg2 rotate
xform backvreg bitm,bankm hiseg

FUNCTION

The DEFVW command defines the view into the Display List world coordinate system (WCS) and how it is displayed in the display coordinate system (DCS). The UP vector, rotation, and transformation are defined. The background color for the view is specified, as are the writemask and the highest level segment to be included in the view.

To explain the DEFVW command further: the user defines graphics primitives in a 64,000 by 64,000 world coordinate system (WCS). The view specifies the portion of the WCS that is to be displayed, and describes the display in detail. Up to eight views--numbered from 0 to 7--can be simultaneously defined and displayed. Two coordinate registers wcsregl and wcsreg2 specify the corners of the window within the world coordinate system.

Each view includes a window of the WCS that is to be displayed on the screen. This window specifies a rectangle which may present a squeezed or stretched view of the WCS window. The screen display system is called the device coordinate system (DCS); the DCS has coordinates corresponding to the screen size. The Model One/25 has a window from (-256,-256) to (255,255); the Model One/40 DCS goes from (-512,-512) to (511,511) as does the Model One/60 image memory, although the actual display window is from (-384,-288) to (383,287). The two coordinate registers dcsregl and dcsreg2 specify the display window.

To specify rotations of the WCS as it is displayed in the DCS window, a vector which defines UP is used. The two endpoints of this UP vector are defined by the registers upregl and upreg2. rotate then gives the center around which the window is rotated. rotate is a coordinate register containing an x,y point.

xform gives the type of transformation that is desired when the WCS window is mapped into the DCS window. xform=0 specifies that no transformation is done for display; this is especially useful for menus and other primitives that should not be changed before display. xform=1 indicates that translation and scaling should be done; the UP vector is ignored. xform=2 specifies that only rotation should be done. xform=3 specifies translation, rotation, and scaling should be done. The default value is xform=1.

A background color for the window can be specified, using backvreg. When REDRAWs are done, using the REDRAW command, the window can be cleared to this background value first, if desired.

bitm and bankm specify the writemask for the image memory. These two parameters function exactly as they do in the WRMASK command and should be used in the same way.

hiseg specifies the number of the highest level segment that is included in the view. For a top-level view, for example, the user would define the view to reference the highest level segment only. Then, unwanted detail from lower-level segments could be controlled with the SETATR command, which can make selected segments "invisible".

PARAMETERS

view the view number; range is 0 to 7.

wcsreg1,wcsreg2
this pair of coordinate registers specifies the diagonal corners of the WCS window. The range is 0 to 63 (although some cregs are reserved).

dcsreg1,dcsreg2
this pair of coordinate registers specifies the diagonal corners of the DCS window. The range is 0 to 63 (although some cregs are reserved).

upreg1,upreg2
this pair of coordinate registers specifies the two ends of the UP vector.

rotate the number of the coordinate register containing the WCS x,y center of rotation

xform transformation type; the range is 0 to 3, as follows: 0: no transformation;1: translate and scale; 2: rotate;3: translate, rotate, and scale.

backvreg the number of the value register containing the background color.

bitm,bankm
the writemask for the view; see the WRMASK command for details.

hiseg the highest level of nested segments to be included in the view.

HOST BINARY COMMAND STREAM

[EBH] [view] [wcsreg1] [wcsreg2] [dcsreg1] [dcsreg2] [upreg1] [upreg2]
[rotate] [xform] [backvreg] [bitm] [bankm] [hiseg] (14 bytes)
EBH=353(octal)=235(decimal)

FORTRAN SUBROUTINE CALL

```
CALL DEFVW (VIEW,WCSREG1,WCSREG2,DCSREG1,DCSREG2,UPREG1,UPREG2,
            ROTATE,XFORM,BACKVREG,BITM,BANKM,HISEG)
```

EXAMPLE

```
DEFVW 2 22,23 24,25 26,27 28 3 14 255,7 3
```

This command defines view 2 with WCS window corners specifed by cregs 22 and 23, DCS window corners specified by cregs 24 and 25, UP vector corners given by cregs 26 and 27, the rotation center given by creg 28, the transformation type of 3, a background value given by vreg 14, a writemask (bitm,bankm) of 255,7 (write all banks, all bit planes), and a high segment number of 3.

RELATED COMMANDS

all Display List commands

D E L P I D

SYNTAX

DELPID

FUNCTION

The DELPID command deletes all graphics primitives with a pick identification number equal to PIDREG and a segment number equal to SEGREG. The graphics primitives are deleted from the current pickid to the next pickid or to the end of the segment, whichever comes first.

PARAMETERS

None.

HOST BINARY COMMAND STREAM

[ECH] (1 byte)
ECH=354(octal)=236(decimal)

FORTRAN SUBROUTINE CALL

CALL DELPID

EXAMPLE

```
SET SEGREG 1        Set the segment register id
SET PIDREG 15       Set the pickid to 15
DELPID              Delete all primitives with pickid of 15 in segment 1
```

RELATED COMMANDS

all Display List commands

H I L I T E

SYNTAX

HILITE view,flag,vreg

FUNCTION

The HILITE command highlights all graphics primitives with the current pickid and segment number, using PIDREG and SEGREG (see the SET command). The view to be highlighted is given by view; the highlight color is given by value register vreg. If flag=1 or ON, the corresponding primitives are highlighted; if flag=0 or OFF, the primitives are drawn normally. This option can be used to unhighlight previous highlighted segments.

PARAMETERS

view	the view number (see DEFVW)
flag	if flag=1 or ON, graphics primitives are highlighted in color specified by vreg. If flag=0 or OFF, graphics primitives are drawn normally.
vreg	value register containing the highlight color. The default is value register 0 (the current color).

HOST BINARY COMMAND STREAM

[DFH][view][flag][vreg] (4 bytes)
DFH=337(octal)=223(decimal)

FORTRAN SUBROUTINE CALL

CALL HILITE (VIEW,FLAG,VREG)

EXAMPLE

```
SET SEGREG 1        Set the segment register id
SET PIDREG 15       Set pickid to 15
HILITE 1 1 4        Highlight all primitives with pickid of 15
                    in segment 1, view 1, using the color in vreg 4
HILITE 1 0          Returns highlighted primitives to original color.
                    Note that vreg is not needed here.
```

RELATED COMMANDS

all Display List commands

P I C K C R

SYNTAX

PICKCR view dcsreg searchflag

FUNCTION

The PICKCR command is used in pick searches of a view. view gives the view number to be picked. dcsreg specifies the coordinate register containing the DCS coordinates for the pick. searchflag specifies the conditions of the search. If searchflag is ON or 1, the search is continued from the current location (allowing continuing searches through a tree to pick among multiple overlapping objects: see the RDTREE command). If searchflag is OFF or 0, the search is started from the top of the segment.

PARAMETERS

view the view number (see DEFVW); range is 0 to 7.
dcsreg the coordinate register containing the DCS coordinates for the pick.
searchflag
If 1 or ON, pick search is from current location; if 0 or OFF, pick search is from top of segment.

HOST BINARY COMMAND STREAM

[E3H] [view] [dcsreg] [searchflag] (4 bytes)
E3H=343(octal)=227(decimal)

FORTRAN SUBROUTINE CALL

CALL PICKCR (VIEW,WCSREG,SEARCHFLAG)

EXAMPLE

PICKCR 2 24 ON Performs search from current location of view 2, using coordinate register 24 for DCS coordinates.

RELATED COMMANDS

all Display List commands

R D P I D

SYNTAX

RDPID

FUNCTION

The RDPID command reads and returns the graphics primitives with a pickid equal to PIDREG and a segment number equal to SEGREG. A list of primitives is returned: the format is (n,nray). n gives the number of words; nray is a vector of n words.

For RDPID, a graphics primitive is defined as all the graphics primitives between two SEGPID commands (see SEGPID for more details). n thus gives the number of words; nray gives the words defining the graphics primitives in raw form--opcodes and parameters.

PARAMETERS

None.

HOST BINARY COMMAND STREAM

```
[EDH]                (1 byte)
EDH=355(octal)=237(decimal)
```

FORTRAN SUBROUTINE CALL

CALL RDPID (BYTESRAY(m,n))

BYTESRAY is an array of size m,n where m is the number of bytes and n is the raw data.

EXAMPLE

```
SEGDEF 1              Define segment 1
  :
SEGPID 4
MOVABS 10,-10
SEGPID 5
  :
SEGEND
  :
SET SEGPID 1        Set SEGPID to 1
SET PIDREG 4        Set PIDREG to 4
RDPID
00003
00004 00010 -00010 00000
                    (Indicates command MOVABS 10,-10)
```

RELATED COMMANDS

all Display List commands

R D R E G

SYNTAX

RDREG

FUNCTION

The RDREG command returns the current segment number and the current pick identification number, as set by the SET SEGREG and SET PIDREG commands.

PARAMETERS

None.

HOST BINARY COMMAND STREAM

[E0H] (1 byte)
E0H=340(octal)=224(decimal)

FORTRAN SUBROUTINE CALL

CALL RDREG (SEGMENT,PICKID)

EXAMPLE

```
SET PIDREG 46          Set current pickid to 46 (load PID register)
SET SEGREG 10          Set current segment to 10 (load segment register)
RDREG
 00010 00046           Segment is 10, pickid is 46
```

RELATED COMMANDS

all Display List commands

SYNTAX

RDTREE

FUNCTION

The RDTREE command returns the hierarchical history generated by the PICKCR command. The nesting level (plus 1) and an 9x2 array of the segment and pickid numbers are returned.

PARAMETERS

None.

HOST BINARY COMMAND STREAM

```
[EEH]                (1 byte)
EEH=356(octal)=239(decimal)
```

FORTRAN SUBROUTINE CALL

CALL RDTREE (NESTING,SEGPID)

SEGPID is an 9x2 array of segment and pickid numbers.

EXAMPLE

```
SEGDEF 5            Begin view data structure definition
  :
SEGPID 1
SEGREF 1
  :
SEGEND
SEGDEF 1
SEGPID 0
SEGREF 13

  :
SEGEND
SEGDEF 13
  :
SEGPID 100
DRWREL 100 100
SEGEND              End view definition
                    CREG 2 contains the DCS coordinates of a point on
                    or near (see SET PICKAP) the DRWREL 100,100 vector
PICKCR 0,2,0        Set up search through view 0
RDTREE              Execute RDTREE command
00003
00005 00001
00001 00000
```

```
00013 00100
00000 00000
00000 00000
00000 00000
00000 00000
00000 00000
00000 00000
00000 00000
```

RELATED COMMANDS

all Display List commands

REDRAW

SYNTAX

REDRAW view,flag

FUNCTION

The REDRAW command redisplays all segments associated with the specified view. If the flag is ON or 1, the window of the display is cleared to the background color before the display is done. All views can be erased by giving a view of -1.

PARAMETERS

view	the view number; range is -1 to 7. If the view is -1, all views are redisplayed.
flag	flag=ON or 1, erase before display; flag=OFF or 0, do not erase before display.

HOST BINARY COMMAND STREAM

[E2H][view][flag] (3 bytes)
E2H=342(octal)=226(decimal)

FORTRAN SUBROUTINE CALL

CALL REDRAW (VIEW,FLAG)

EXAMPLE

REDRAW 2 ON Redraw view 2; erase before redrawing.
REDRAW -1 OFF Redraw all views; do not erase first.

RELATED COMMANDS

all Display List commands

S E G A P P

SYNTAX

SEGAPP segment

FUNCTION

The SEGAPP command is used to append graphics primitive commands to an existing segment definition. The specified segment is reopened; after all desired commands are added, the SEGEND command closes the opened segment.

PARAMETERS

segment the segment to which commands will be appended

HOST BINARY COMMAND STREAM

[DBH][segment] (2 bytes)
DBH=333(octal)=219(decimal)

FORTRAN SUBROUTINE CALL

CALL SEGAPP (SEGMENT)

EXAMPLE

```
SEGDEF 10          Begin definition of segment 10
  :
  :                Model One graphics primitives commands
  :
SEGEND             End definition of segment 10
SEGAPP 10          Re-open segment 10 to append commands
  :
  :                Additional Model One commands
  :
SEGEND             Re-close segment 10 after appending commands
```

RELATED COMMANDS

all Display List commands

S E G C O P

SYNTAX

SEGCOP segment2, segmentl

FUNCTION

The SEGCOP command copies segmentl into segment2, leaving segmentl unchanged. If segment2 already exists, it is overwritten.

PARAMETERS

segmentl the segment to be copied; range is 0 to 32,767.
segment2 the name of the segment into which segmentl is copied.

HOST BINARY COMMAND STREAM

[EAH][segment2][segmentl] (3 bytes)
EAH=352(octal)=234(decimal)

FORTRAN SUBROUTINE CALL

CALL SEGCOP (SEGMENT2,SEGMENTl)

EXAMPLE

```
SEGDEG 10          Begin definition of segment 10
  :
  :                Model One graphics primitive commands
  :
SEGEND
SEGCOP 12 10       Copy segment 10 into segment 12
```

RELATED COMMANDS

all Display List commands

SEGDEF

SYNTAX

SEGDEF segment

FUNCTION

The SEGDEF command begins the definition of a Display List segment. segment gives the segment number; while its range is from 0 to 32,767, only roughly 500 segments may be simultaneously defined. It may be useful, however, to number the segments in increments of ten, like program statements, to allow room for expansion.

A Display List segment may only include these commands:

1. Graphics primitive commands: the move commands, the draw commands, POLYGN, the text commands, the circle commands, the rectangle commands, and other graphics primitives.

2. The pixel value commands: VALUE, VAL1K, VAL8. Note that each byte of a command takes up Display List space, so it may be desirable to use VAL1K or VAL8 commands if minimal space use is desired. VMOVE may be used for dynamic color control.

3. Changes to primitive generation attributes: VECPAT, PRMFIL, etc.

PARAMETERS

segment the segment number; range is from 0 to 32,767.

HOST BINARY COMMAND STREAM

[DCH] [segment] (2 bytes)
DCH=334(octal)=220(decimal)

FORTRAN SUBROUTINE CALL

CALL SEGDEF (SEGMENT)

EXAMPLE

```
SEGDEF 10       Begin definition of segment 10.
  :
  :             Model One graphics primitive commands
  :
SEGEND          End definition of segment 10
```

RELATED COMMANDS

all Display List commands

S E G D E L

SYNTAX

SEGDEL segment

FUNCTION

The SEGDEL command deletes the specified Display List segment. segment gives the number of the segment.

PARAMETERS

segment Display List segment number

HOST BINARY COMMAND STREAM

[DEH] [segment] (2 bytes)
DEH=336 (octal)=222 (decimal)

FORTRAN SUBROUTINE CALL

CALL SEGDEL (SEGMENT)

EXAMPLE

SEGDEL 10 Delete the definition of segment 10.

RELATED COMMANDS

all Display List commands

S E G E N D

SYNTAX

SEGEND

FUNCTION

The SEGEND command ends the definition of a Display List segment. See SEGDEF for details of segment definition.

PARAMETERS

None.

HOST BINARY COMMAND STREAM

[DDH] (1 byte)
DDH=335(octal)=221(decimal)

FORTRAN SUBROUTINE CALL

CALL SEGEND

EXAMPLE

```
SEGDEF 10        Begin definition of segment 10.
  :
  :              Model One graphics primitive commands
  :
SEGEND           End definition of segment 10.
```

RELATED COMMANDS

all Display List commands

S E G I N I

SYNTAX

SEGINI words

FUNCTION

The SEGINI command initializes the Display List Firmware and deletes any existing segments and view definitions. words assigns the number of data words to each segment block. The default is 254 words (invoked by setting words=0); however, if the segments will be small, it may be desirable to define a smaller block size.

PARAMETERS

words — minimum number of data words per segment block; the segment may be continued from block to block

HOST BINARY COMMAND STREAM

[E1H][words] (2 bytes)
E1=341(octal)=225(decimal)

FORTRAN SUBROUTINE CALL

CALL SEGINI (WORDS)

EXAMPLE

SEGINI 128 — Initializes Display List Firmware with a segment block size of 128 data words

RELATED COMMANDS

all Display List commands

SEGINQ

SYNTAX

SEGINQ segment

FUNCTION

The SEGINQ command returns the attributes of the specified segment. One word is returned (16 bits) with the two least significant bits (LSBs) indicating the attributes of the segment. The least significant bit indicates whether the visibility of the segment is ON (1) or OFF (0); the next least significant bit indicates whether the pickability of the segment is ON (1) or OFF (0). Thus, a returned value of zero indicates that both are OFF; 1 indicates visibility is ON; 2 indicates pickability is ON; and 3 indicates both are ON.

PARAMETERS

segment the segment number whose attributes are being queried; range is 0 to 32,767.

HOST BINARY COMMAND STREAM

[E5H][segment] (2 bytes)
E5H=345(octal)=229(decimal)

FORTRAN SUBROUTINE CALL

CALL SEGINQ (SEGMENT,STATUS)

EXAMPLE

```
SETATR 10 VIS ON     Set segment 10 visibility ON
SETATR 10 PICK OFF   Set segment 10 pickability OFF
SEGINQ 10            Query attributes of segment 10
 00001               Value of 1 indicates visibility ON, pickability OFF
```

RELATED COMMANDS

all Display List commands

S E G P I D

SYNTAX

SEGPID pickid

FUNCTION

The SEGPID command defines the pick identification number for the graphics primitives--vectors, circles, rectangles, and so on--immediately following the SEGPID command until another SEGPID command is reached or until the end of the segment (SEGEND) is reached.

All Model One graphics primitives will be assigned that pickid. Thus, a RDPRM, HILITE, or DELPID command will find all the primitives associated with the pickid. (Note: these commands use a pick identification number register set with SET PIDREG to determine which primitives to pick.)

PARAMETERS

pickid pick identification number; range is 0 to 32,767.

HOST BINARY COMMAND STREAM

[D9H][pickid] (2 bytes)
D9H=331(octal)=217(decimal)

FORTRAN SUBROUTINE CALL

CALL SEGPID (PICKID)

EXAMPLE

```
SEGDEF 10          Begin definition of segment 10
  :                Model One graphics primitives commands
SEGPID 46          Assign pick identification number of 46 to all
CIRCLE 50          following graphics primitives
RECTAN 0,0
SEGPID 47          Change to pick identification number of 47
TEXT1 Pickid is 47
SEGEND
```

Note that the circle and the rectangle are both included in pickid number of 46; the text has a pickid number of 47.

RELATED COMMANDS

all Display List commands

S E G R E F

SYNTAX

SEGREF segment

FUNCTION

The SEGREF command nests the specified segment within another segment. Segments may be nested up to eight levels deep. There is no practical limit to the number of segment references at any given level. For example, segment definition 1 can reference any number of other segments; if one of those segments in turn references another segment, the nesting level is two. The viewing transformation applied is unaffected by the nesting of segments. When nesting segments, the user should be careful to use relative moves and draws in nested segments; in addition, the nested segments should restore the current point before returning to the calling segment.

PARAMETERS

segment the segment to be nested; range is 0 to 32,767.

HOST BINARY COMMAND STREAM

[D8H] [segment] (2 bytes)
D8H=330(octal)=216(decimal)

FORTRAN SUBROUTINE CALL

CALL SEGREF (SEGMENT)

EXAMPLE

```
SEGDEG 10      Begin definition of segment 10
  :
SEGREF 12      Nest segment 12 within segment 10
  :
SEGEND         End definition of segment 10
```

RELATED COMMANDS

all Display List commands

S E G R E N

SYNTAX

SEGREN segment2 segment1

FUNCTION

The SEGREN command is used to rename a specified segment. segment1 gives the segment to be renamed; segment2 gives the new name.

Note that references to the old segment (SEGREF command) are not changed when the segment is renamed.

PARAMETERS

segment1 the segment to be renamed; range is 0 to 32,767.
segment2 the new segment name; range is 0 to 32,767.

HOST BINARY COMMAND STREAM

[DAH][segment2][segment1] (3 bytes)
DAH=332(octal)=218(decimal)

FORTRAN SUBROUTINE CALL

CALL SEGREN (SEGMENT2,SEGMENT1)

EXAMPLE

```
SEGDEF 10        Begin definition of segment 10
  :
  :              Model One graphics primitives commands
  :
SEGEND
SEGREN 12 10     Rename segment 12 to segment 10
```

RELATED COMMANDS

all Display List commands

S E T

SYNTAX

SET global value

FUNCTION

The SET command is used to set the value of globally used parameters: the pick error margin aperture, the current pickid value, and the current segment number. These parameters are used by other commands, such as PICKCR, HILITE, RDPRM, and DELPID. (Note that SET PIDREG and SEGPID are different commands; SET PIDREG sets the pick identification number for picking, while SEGPID sets the pick identification number for a given segment.)

The possible forms of the SET command are:

1. SET PICKAP aperture
   The pick error margin aperture is specified in DCS units.

2. SET PIDREG value
   The pickid register is set to value.

3. SET SEGREG value
   The segment number register is set to value.

Note that setting SEGREG and PIDREG invalidates the current pick history.

In summary, the SET command works as follows:

SET global value

where global may be specified with the parameter (0 for PICKAP; 1 for PIDREG; 2 for SEGREG) or with the character string (in local mode only).

PARAMETERS

global — represents a possible global value that may be set by this command; possible values are PICKAP (0), PIDREG (1), and SEGREG (2) (see above for details).

value — for PICKAP: aperture error margin in DCS units; for PIDREG: pickid number, with a range of 0 to 32,767; for SEGREG: segment number, with a range of 0 to 32,767.

HOST BINARY COMMAND STREAM

[46H] [global] [highval] [lowval] (4 bytes)
46H=106 (octal)=70 (decimal)

FORTRAN SUBROUTINE CALL

CALL SET (GLOBAL,VALUE)

EXAMPLE

```
SET PICKAP 10       Set the pick aperture error margin to 10 DCS units
SET PIDREG 104      Set the pickid register to 104
SET SEGREG 15       Set the segment number register to 15
```

RELATED COMMANDS

all Display List commands

S E T A T R

SYNTAX

SETATR segment attribute flag

FUNCTION

The SETATR command sets the visibility or pickability of a segment. The two possible attributes are VIS and PICK. These may be entered locally as character strings; from the host, the codes are 0 for VIS and 1 for PICK. A segment number of -1 sets the attribute for all segments.

PARAMETERS

segment	the segment number; range is 0 to 32,767. A value of -1 sets attributes for all defined segments.
attribute	the attribute to be set: choices are VISibility (0) or PICKability (1).
flag	the attribute flag; ON or 1 enables visibility or pickability; OFF or 0 disables the attribute.

HOST BINARY COMMAND STREAM

[E6H] [segment] [attribute] [flag] (4 bytes)
E6H=
346(octal)=236(decimal)

FORTRAN SUBROUTINE CALL

CALL SETATR (SEGMENT,ATTRIBUTE,FLAG)

EXAMPLE

```
SETATR 10 VIS ON      Set segment 10 visibility ON
SETATR 10 PICK OFF    Set segment 10 pickability OFF
SEGINQ 10             Query attributes of segment 10
 00001                Value of 1 indicates visibility ON, pickability OFF
```

RELATED COMMANDS

all Display List commands

SYNTAX

SYSTAT

FUNCTION

The SYSTAT command returns system memory usage and availability. A single 16-bit word is returned giving the number of free memory blocks of the size defined with the SEGINI command.

PARAMETERS

None.

HOST BINARY COMMAND STREAM

[E4H] (1 byte)
E4H=344(octal)=228(decimal)

FORTRAN SUBROUTINE CALL

CALL SYSTAT (FREEBLOCKS)

EXAMPLE

SYSTAT	Query system memory availability
00117	117 free blocks of memory

RELATED COMMANDS

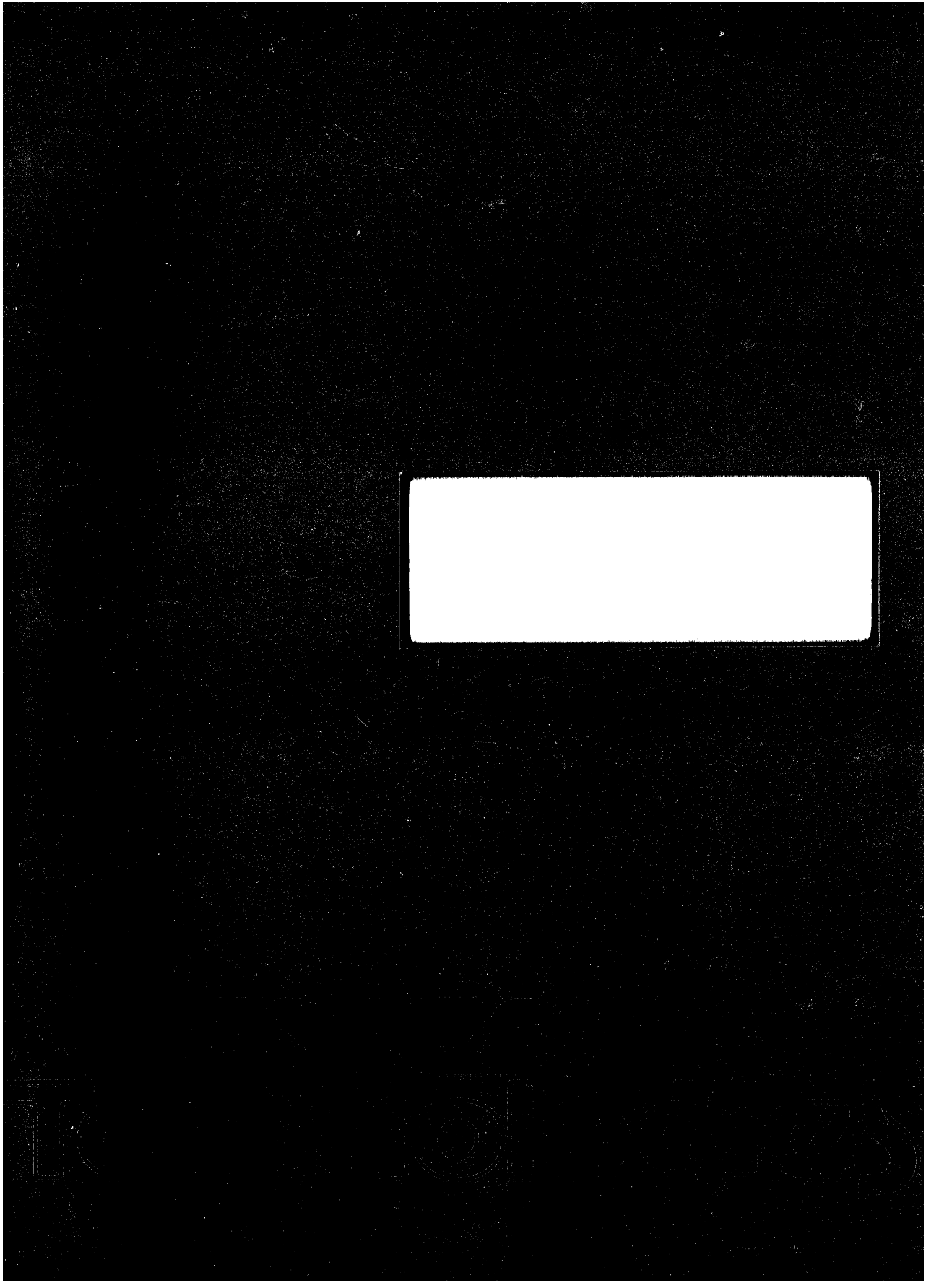
all Display List commands

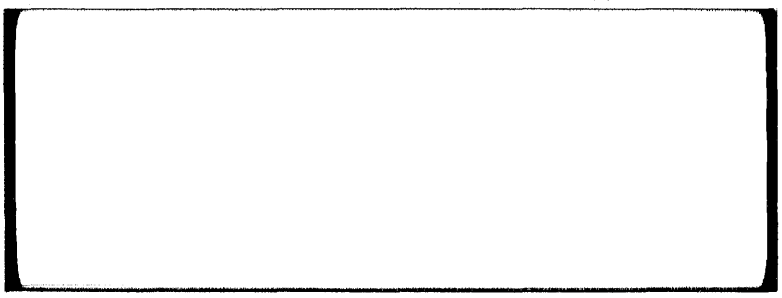
7.0 QUICK COMMAND REFERENCE

This section provides a quick reference to the Display List Firmware commands.

DEFVW view,wcsregl,wcsreg2 dcsregl,dcsreg2,upregl upreg2,rotate,xform, backvreg,bitm,bankm, hiseg	Defines the view into the World Coordinate System. view gives the view number; wcsregl and wcsreg2 define the WCS corners; dcsregl and dcsreg2 define the display viewport; upregl and upreg2 give the ends of the UP vector; rotate defines the WCS window rotation center; xform specifies the transformation type; backvreg gives the background color; for bitm and bankm, see the WRMASK command; hiseg specifies the segment nesting. [EBH]
DELPID	Delete primitives with PIDREG and SEGREG. [ECH]
HILITE view,flag,vreg	Highlight view in the color specified by vreg. flag=1 or ON highlights primitives; flag=0 or OFF draws primitives normally. PIDREG and SEGREG (see SET) are used. [DFH]
PICKCR view,dcsreg,searchflag	Perform pick search; view is picked, using the dcsreg coordinates. searchflag=1 or ON, search from current tree location; searchflag=0 or OFF, search from the top of the segment. [E3H]
RDPID	Reads and returns the graphics primitives with PIDREG and SEGREG (see SET). [EDH]
RDREG	Reads and returns the current SEGREG and PIDREG (see SET and PICKCR). [E0H]
RDTREE	Returns the PICKCR hierarchical history. [EEH]
REDRAW view,flag	Redisplay view; flag=1 or ON clears the window to the background color before display. view=-1 redisplays all views. [E2H]
SEGAPP segment	Open segment to append graphics primitive commands. SEGEND ends the append. [DBH]
SEGCOP segment2,segment1	Copies segment1 into segment2. [EAH]
SEGDEF segment	Begins definition of segment. [DCH]
SEGDEL segment	Deletes segment. [DEH]
SEGEND	End segment definition begun with SEGDEF or SEGAPP. [DDH]
SEGINI words	Initializes Display List Firmware; words gives the number of data words per segment block. [E1H]
SEGINQ segments	Returns the attributes of segment. (See SETATR). [E5H]
SEGPID pickid	Defines the pickid for primitives within segment until next SEGPID or SEGEND command. [D9H]
SEGREF segment	Nest specified segment within current segment. [D8H]
SEGREN segment2,segment1	Renames segment1 to segment2. [DAH]
SET global,value	Set PICKAP aperture; set PIDREG value; set SEGREG value. [46H]
SETATR segment,attribute,flag	Sets VISibility or PICKability of segment. flag=1 or ON enables attribute (VIS or PICK); flag=0 or OFF disables attribue. [E6H]

SYSTAT	Returns system memory usage and availability. [E4H]





INTRODUCTION TO THE RASTER TECHNOLOGIES MODEL ONE

Revision 5.0 May 18, 1983

Introduction to the Raster Technologies Model One
May 18, 1983

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.

Table of Contents

List of Figures

List of Tables

ABOUT THIS MANUAL

The Raster Technologies Model One family of color graphic display controllers includes the Model One/25, the Model One/40, and the Model One/60. This manual introduces the Model One family, providing an overview of the Model One system architecture and a summary of the features and specifications of the display system. Section 1.0 provides an overview of the Model One family of graphics systems. Section 2.0 describes the Model One's standard firmware command set; section 3.0 describes the host FORTRAN subroutine library. Sections 4.0 through 11.0 describe the functional blocks of the Model One. An appendix provides a complete list of the firmware command set.

1.0 INTRODUCTION

The Model One/25, the Model One/40, and the Model One/60 fully support general purpose graphics, interactive CAD, simulation, and imaging applications. The Model One display controller forms the heart of a computer graphics workstation and provides the local intelligence to off-load interactive tasks from the host computer system. The workstation consists of a Model One display controller, a high-resolution video monitor (either 512x512, 768x576, or 1024xl024), and local interactive devices such as a keyboard or digitizer. Figre 1.1 shows such a workstation.

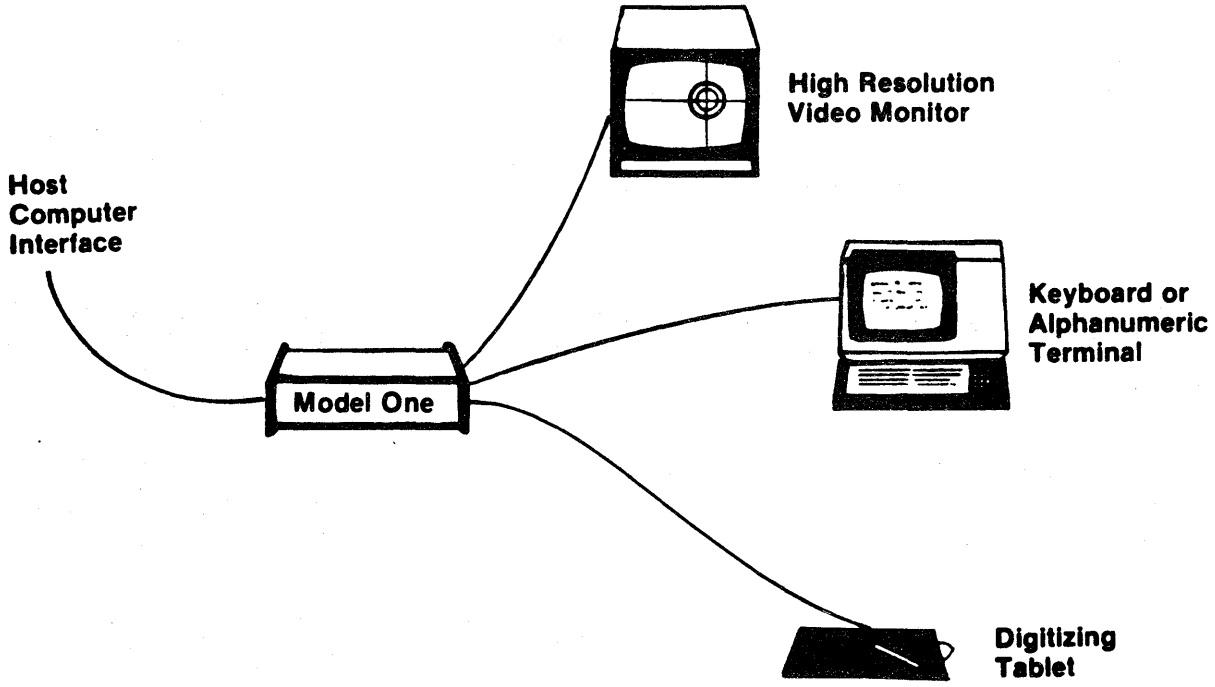


Figure 1.1 Interactive Graphics Workstation

The Model One/25 is designed for general purpose graphics and full-color imaging applications. It includes up to 24 bit planes of image memory. Its standard RS-170 red, green, blue outputs allow it to drive a 512x512 monitor. The Model One/25 is also available in a broadcast-compatible 512x484 configuration, which may be NTSC-encoded for video taping and video mixing. The Model One/25 includes three 8-bit-in, 8-bit-out programmable Look-Up-Tables, allowing over 16 million simultaneously displayable colors.

The Model One/40 is designed for ultra-high resolution applications; it includes up to six bit planes of image memory, with a 1024xl024 visible display. The Model One/40 provides standard RS-343 red, green, blue outputs. The video Look-Up-Table for the Model One/40 is 6-bit-in, 24-bit-out, allowing up to 64 simultaneously displayable colors, selected from a palette of 16.8 million colors.

The Model One/60 is a flicker-free, 60 Hz refresh high-resolution display system with a visible display window of 768x576 and a total image memory of 1024x1024. The off-screen image memory may be used for symbols, fonts, menus, etc. With up to 6 bit planes of image memory and a 6-bit-in, 24-bit-out video Look-Up-Table, the Model One/60 allows up to 64 simultaneously displayable colors. The video output is RS-343 standard red, green, blue.

1.1 Overview of the Model One Family

In operation, the Model One controller is sent commands from the host computer to update the displayed image and communicate with the local interactive devices at the workstation. The Model One is delivered with an RS-232 standard host interface, which operates at speeds ranging from 75 baud to 38.4 Kbaud. Two optional higher-speed host interfaces are available: IEEE-488 and a Direct Memory Access port for high-speed transfer of commands and image data between the host computer and the Model One.

Multiple pipelined processors within the Model One are used to interpret and execute the host command stream at the fastest possible rate. The 16-bit central processor, hardware vector generator, and pixel processor, all standard hardware features of the Model One, provide the fastest possible execution of graphics commands.

The Model One offers an integrated set of over 150 firmware-based graphics commands, compatible with the full product line. The command set includes:

* Graphics Primitives: lines, polygons, circles, arcs, rectangles, horizontal and vertical text, and points, are available with firmware commands. The standard command set includes several approaches to drawing primitives, including relative and absolute addressing.

* Area Fills: two kinds of area fills, as well as flooding and clearing of the display, are provided.

* Display Control: zooming, panning, read and write masks, clipping window, and crosshairs are among the display control commands.

* Look-Up-Table Control: a complete set of commands is available for programming the video look-up-tables.

* Data Read-Back: read-back of image memory windows, as well as pixel values, current point, and so on, is provided with straightforward firmware commands.

* Image Transmission: several formats for transmission of images from the host to the Model One are supported.

* Interactive Device Support: a set of commands, such as blinking and button support, is provided for interactive device support.

* Macro Programming: firmware macro definition commands are supplied, simplifying application development.

* Register Operations: commands for manipulation of the Model One's coordinate and value registers are supplied.

* Software Development: the applications developer has a full set of commands to aid software development, including a command stream translator and a local debugger.

* Alphanumeric Terminal Emulation: these commands allow the monitor to be used as an alphanumeric terminal with attached keyboard. Up to nine independent scrolling windows can be specified and used simultaneously; these windows can be any size and placed whereever desired.

* Display List Firmware: local scaling, rotation, and translation can be handled locally. The user works in a 64,000 by 64,000 world coordinate system and may define up to eight simultaneous views into the world coordinate system.

The Model One firmware commands are described in more detail in Section 2.0 of this manual; Appendix I provides a complete listing.

A complete library of FORTRAN-callable subroutines is available with the Model One for host-level access to all firmware commands.

The Model One is contained in a rack-mountable 5.25"-high enclosure which houses the power supply, connectors, and four Model One printed circuit cards. The Model One may be located at the graphics workstation or rack-mounted with the host computer. All available hardware options fit within the Model One's standard 5.25" enclosure.

1.2 Applications Development Features

The Model One includes extensive features to speed development of applications programs. The Model One development tools are designed specifically to minimize typical problems encountered in applications development and support, as shown in the chart below.

Software Development Problems	Model One Approach
Off-loading graphics tasks from the host program: clipping, scrolling windows, rubber-banding, dragging	Broadest command set available: allows the Model One to perform operations previously requiring host intervention
Training new graphics programmers for the Model One	English-like command mnemonics; local mode allows commands to be tested locally, without writing host programs
Testing new graphics concepts and sequences without writing or rewriting host programs	Local mode
Cost of programming microprocessor intelligence for graphics use	Macro command sequences
Application menu management and on-going support	Macro command sequences
DMA application debugging	Command stream translator allows local examination of the high-speed command stream.
Break-pointing new application programs during development: stepping through programs under local control	Complete local debugger allows breakpointing, stepping through programs, local command execution, and listing of defined macros
Local management of cursors, buttons, and other interactive devices	Macro command sequences

Table 1.1 The Model One Answers to Software Development Problems

The Model One firmware includes support for named, storable sequences of commands, called macros. Macros are a series of graphics commands which may be defined and stored for repeated execution with a single command. Up to 256 macros may be stored at any one time.

The Model One's local debugger allows the programmer to step through program execution, list defined macros, execute graphics commands locally, and return to normal execution when desired. The command stream translator is especially useful when combined with the local debugger. The command stream translator, when invoked, translates the command stream from the host into human-readable mnemonics (the same mnemonics used when entering commands locally) and prints those mnemonics and their associated parameters at the local terminal. The command stream translator can be used with the serial, IEEE-488, and DMA interfaces. Once application development is complete, the command stream

translator and local debugger can be disabled; of course, they remain available for diagnostic use. Table 1.2 shows sample output from the Model One's command stream translator.

Command Stream From Host	Local ALPHA Output
..0100040006..	MOVABS 0004 0006
..0600FFFF..	VALUE 00 255 255
..0E002D..	CIRCLE 100
..0200E300D4..	MOVREL 100 105
..9005C8E5ECECEF..	TEXT1 Hello

Table 1.2 Sample Command Stream Translator Output

1.3 System Architecture Overview

The hardware architecture of the Model One includes these functional blocks:

- -central processor
- -hardware vector generator
- -pixel processor
- -image memory
- -Look-Up-Tables
- -video output
- -optional DMA interface
- -optional pixel mover

Figure 1.2 shows the data paths and relationships between these functional blocks.

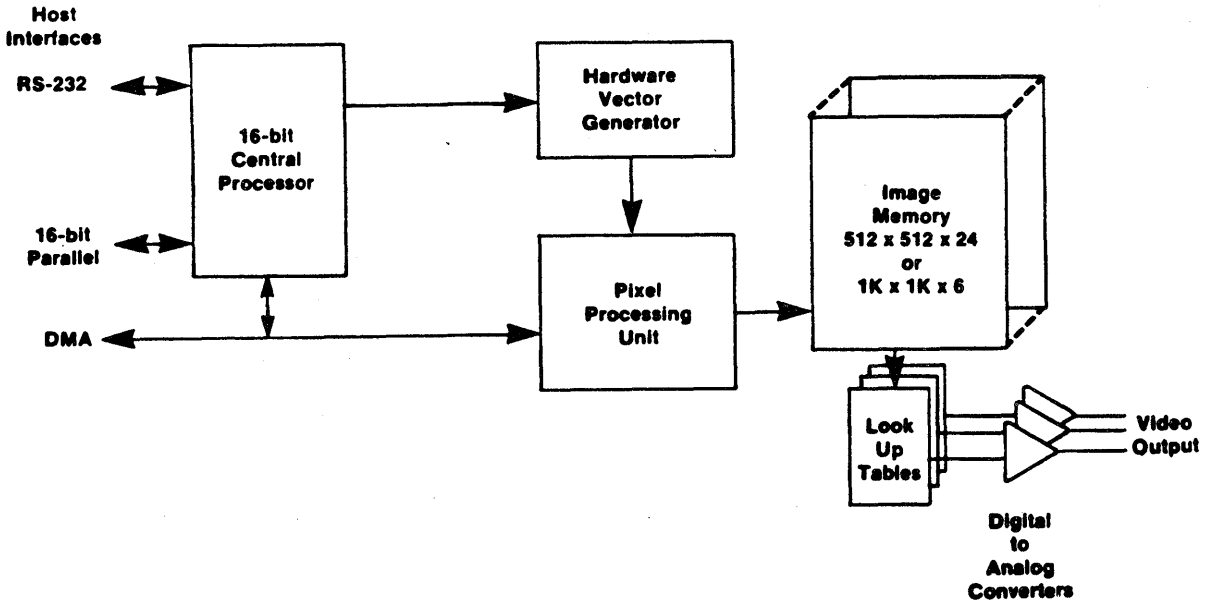


Figure 1.2 Data Paths and Functional Blocks

The Model One architecture includes an easy-to-use powerful microprocessor, coupled with hardware accelerators for high performance.

The Model One central processor is a general purpose 16-bit microcomputer whose principal function is to interpret and decode graphics commands received by the Model One, then direct the hardware vector generator and pixel processor to execute those commands.

The hardware vector generator is a custom processor which executes a digital line generating algorithm at high speed. The central processor continues processing of incoming commands while the vector generator makes real-time updates to image memory.

The pixel processor performs logic operations on pixel values which are being placed into image memory. The Model One/25 also allows arithmetic functions. The pixel processor can be used in CAD applications to provide a write-through mode for non-destructive dragging of objects in image memory. The pixel processor also supports operations on images which are transferred over the Model One's high-speed DMA port.

The optional DMA interface performs high-speed DMA transfers of both commands and image data between the Model One and the host computer. The optional pixel mover performs high-speed moves of rectangular blocks of pixels.

1.4 Model One/25, Model One/40, and Model One/60 Comparison

The Model One/25, Model One/40, and Model One/60 all offer a complete integrated graphics controller, including the standard firmware set, applications development environment, central processor, vector generator, and pixel processor. The primary differences between the three models arise from the configuration of image memory and the use of the video Look-Up-Tables.

The Model One/25 is a full-color graphics controller, offering up to 24 bit planes of 512x512 pixels. Three 8-bit-in, 8-bit-out Look-Up-Tables, one for each color bank (red, green, and blue), are supported. These can be used for double-buffering or animation of locally-stored images or as a single 8-bit-in, 24-bit-out Look-Up-Table for pseudo-color imaging.

The Model One/25 also offers a 1024x1024 addressing mode, which is fully software compatible with the Model One/40. In this mode, the Model One/25 appears as a 1024x1024 system with six bit planes, allowing 64 simultaneous colors. For display, four pixels are intensity-averaged to one pixel for display on a standard 512x512 monitor. This produces automatic hardware anti-aliasing. When the image is zoomed in, it is no longer averaged, and a 512x512 window of the 1024x1024 array is displayed.

Programs written for the Model One/25's 1024x1024 addressing mode may be run without alteration on a Model One/40. This allows use of the less-expensive medium-resolution monitor with the same programs as those developed for a more expensive high-resolution monitor.

The Model One/40 offers 1024x1024 resolution, with up to six bit planes of image memory. The Model One/40 supports a single 6-bit-in, 24-bit-out Look-Up-Table for display of up to 64 simultaneous colors, selectable from a palate of over 16 million colors. The Model One/40 is fully compatible with the Model One/25's 1024x1024 addressing mode, and with the Model One/60.

The Model One/60 offers a 60 Hz refresh rate for flicker-free operation, with a high-resolution monitor. The displayed window is 768x576 pixels of an image memory of 1024x1024 with up to six bit planes. The off-screen memory may be displayed by panning, or used for off-screen storage of menus, fonts, and symbols, if desired. Like the Model One/40, the Model One/60 is fully compatible with the rest of the Model One family--the Model One/25's 1024x1024 addressing mode and the Model One/40.

Table 1.3 compares the Model One/25, the Model One/40, and the Model One/60.

Feature	Model One/25 512 mode	Model One/25 1K mode	Model One/40	Model One/60
Display Window (Pixels)	512x512	1024x1024	1024x1024	768x576
Total Image Memory	512x512x24	1024x1024x6	1024x1024x6	1024x1024x6
Bit Planes	24	6	6	6
Refresh Rate	30 Hz	30 Hz	30 Hz <2>	60 Hz
Maximum Simultaneous Colors	16.7 M	64	64	64
Programmable Look-Up-Table	3 8-bit-in,8-bit-out	<1>	6-bit-in, 24-bit-out	6-bit-in, 24-bit-out
Option Card				
Pixel Mover	yes	yes	yes	yes
DMA	yes	yes	yes	yes
Overlay Planes	yes	-	-	-
Firmware Set	Advanced Graphics Application Development Firmware			

<1> The Model One/25's 1024x1024 addressing mode's Look-Up-Table cannot be reset.
<2> The Model One/40 can also display a 512x512 quadrant of the 1024x1024 image at 60 Hz when zoomed in.

Table 1.3 Comparison of Model One/25, Model One/40, and Model One/60

2.0 MODEL ONE STANDARD FIRMWARE

The Model One standard firmware, a rich command set which includes over 100 commands, is the control program run by the Model One's central processor.

The firmware can be divided into two major functions: communications control, and graphic command interpretation and processing. Communication control supports serial and parallel I/O with the host computer and serial I/O with the local alphanumeric terminal and graphic input devices. The graphics command interpreter and processor provides a rich command set, designed for efficiency, versatility, interactivity, and ease of applications development.

2.1 Communications Control

The Model One provides six RS-232C serial ports, which are firmware-configurable. A set of firmware commands allows the user to configure the host port, alphanumeric port, and port for error message transmission. Each port may be configured for RTS, CTS, parity, baud rate (from 75 baud to 38.4 Kbaud), queue size, XON/XOFF protocol, and acceptance of control characters. All input and output through these ports is buffered and interrupt-driven for maximum processor utilization.

The optional IEEE-488 port is also firmware configurable.

Use of the optional 16-bit DMA interface is user-transparent, requiring no special commands.

2.2 Alpha and Graphics Modes

The host and local alphanumeric ports are always in graphics mode or alpha mode; the mode of the port determines whether a character received at the port is interpreted as alphanumeric data or as graphics commands. Only one of the Model One's ports may be in graphics mode at any one time.

To enter graphics mode, the ENTERGRAPHICS control character is sent over the port. The default ENTERGRAPHICS control character is [CTRL-D], although a firmware command (SPCHAR) allows the user to select an alternate character. Once a port enters graphics mode, all subsequent data is interpreted by the graphics command interpreter.

Graphics commands are composed of a one-byte opcode followed by a string of bytes which are operands. The opcode specifies the graphics command--0EH specifies the CIRCLE command--and the operand or operands supplies the data needed for command execution--the radius for a CIRCLE, for example.

2.3 Host Serial Communications Command Format

Commands sent over the host serial interface are normally sent in 8-bit binary format. In binary format, the opcode for each command takes one byte and is sent in the first character, with each byte of operand sent in subsequent characters.

For those host computers which cannot be programmed to transmit 8-bit binary charactes over their serial lines, the Model One can be configured to accept ASCII hexadecimal format. In ASCII hexadecimal format, each 8-bit binary character is replaced by two hexadecimal digits, sent in two successive ASCII hexdecimal characters.

The 8-bit binary format is twice as efficient as ASCII hexadecimal format and should be used for any host computer capable of transmitting 8-bit binary characters.

2.4 DMA Command Format

The DMA port is used with the optional host DMA interface and supports parallel transfers from the host computer. Each transfer includes two bytes of data, a high byte and a low byte. The Model One interprets the high byte of data before the low byte, in exactly the same manner as two successive 8-bit binary characters sent over the serial interface. (The Model One can be configured to interpret the low byte first, if necessary.)

Hence, the data which passes from the host computer is independent of which interface is used--serial or DMA--and is therefore transparent to the applications program.

2.5 Local Alphanumeric Command Format

During program development, it is often helpful to be able to issue graphics commands locally, which allows the applications programmer to test a sequence of commands without writing and compiling a host computer program to generate the commands. The Model One, by providing one port which accepts local alphanumeric input, allows the user to do this simply and rapidly.

When the ENTERGRAPHICS character is typed locally and sent to the local alphanumeric port, the Model One responds with a graphics command prompt [!] to the local terminal. From then until the user types the command QUIT, he can talk directly to the Model One command interpreter instead of the host computer. The Model One expects commands over the local alphanumeric port to be entered in a special English-like mnemonic form, rather than in binary or hexadecimal form. The complete command set is shown in section 11; briefly, the Model One's mnemonics includes such commands as CIRCLE, POINT, MOVREL, and TEXT. The operands for commands may be entered locally in hexadecimal (base 16) or decimal (base 10).

2.6 Diagnostic and Development Features

The Model One's binary transmission format and 16-bit parallel transfers maximize the performance of the Model One when communicating with the host. This gives the Model One an advantage for interactive applications.

Because binary characters and 16-bit parallel transfers are not (usually) human-readable, special debugging features are included to aid the applications programmer in developing and debugging applications. A local debugger and a command stream translator are available; in addition, error messages can be sent to the host or to the local alphanumeric display whenever an error occurs. A firmware REPLAY command allows the user to replay the last

32 characters that were sent over the host interface.

The local debugger can be entered at any time, through typing a single control character at the local terminal. The application programmer can then step through the commands being sent from the host, list all defined macros, list the contents of a specific macro, enable execution of the macro at location zero of the button table, and exit the local debugger.

The Model One's command stream translator allows the user to set the Model One to disassemble the command stream from the host automatically; human readable command mnemonics and parameters are sent to the local alphanumeric terminal. The command stream translator, when combined with the local debugger, thus allows the applications programmer to see the actual commands as they are executed.

The command stream translator and local debugger can be used without regard to the type of host-Model One interface (serial or DMA) in use.

2.7 Coordinate System

The default coordinate system for all graphics commands has (0,0) in the center of image memory, with the positive X axis to the right, and the positive Y axis above. The locations of the four corners of image memory are shown in Table 2.1.

	Lower-left	Upper-left	Lower-Right	Upper-Right
Model One/25				
512 mode	-256,-256	-256,255	255,-256	255,255
1K mode	-512,-512	-512,511	511,-512	511,511
Model One/40	-512,-512	-512,511	511,-512	511,511
Model One/60				
Display	-288,-384	-288,383	287,-384	287,383
Memory	-512,-512	-512,511	511,-512	511,511

Table 2.1 Image Memory Corners for Model One Family

A coordinate origin register, which may be changed with the firmware command CORORG, may be used to change the default coordinate system to any desired orientation. Coordinates are supplied as parameters as two's-complement 16-bit numbers ranging from -32,768 to 32,767. Relative moves and draws which would cause a coordinate to be out of this range are clipped to these values.

2.8 Coordinate Registers

A set of sixty-four coordinate registers are used to manipulate coordinate data in the Model One. Each coordinate register has a 16-bit X-component and a 16-bit Y-component. Some of these registers are assigned to specific functions. Others are available to the user for temporary coordinate storage.

A set of firmware commands allow the user to load coordinate registers (CLOAD), and to add (CADD), subtract (CSUB), and copy/move (CMOVE) between pairs of coordinate registers.

The coordinate registers are defined in Table 2.2.

Coordinate Register	Function
0	Current point: used as a reference point by graphics commands. The current point is modified by MOVE and DRAW commands.
1	Joystick or trackball location, updated automatically by the Model One every 1/30th second.
2	Digitizing tablet cursor location, updated automatically by the Model One every 1/30th second.
3	Coordinate origin: used to position physical image memory within the virtual address space. The coordinate origin is modified by the CORORG command.
4	Screen origin: specifies the point which appears at the center of the screen. The screen origin is changed by the SCRORG command. CREG 4 is used for horizontal and vertical panning. It is also used to scroll alphanumeric window zero.
5	Crosshair 0 current location: changes to this register move crosshair 0. The crosshair is enabled/disabled using the XHAIR command.

Table 2.2 Coordinate Register Assignments
(continued on next page)

Table 2.2 Coordinate Register Assignments (continued)

Coordinate Register	Function
6	Crosshair 1 current location: changes to this register move crosshair 1. The crosshair is enabled/disabled using the XHAIR command.
7	Crosshair 2 (on overlay plane 0) current location; available only for Model One/25 Option Card users.
8	Crosshair 3 (on overlay plane 1) current location; available only for Model One/25 Option Card users.
9	Clipping window, lower-left corner.
10	Clipping window, upper-right corner.
11,12	For Option Card users only: diagonal corners for source window for PIXMOV command.
13	For Option Card users only: PIXMOV destination window.
14	For Option Card users only: pixel writing direction for PIXMOV destination window.
15-19	Reserved.
21-63	Available for use by applications programmer.

Table 2.2 Coordinate Register Assignments

The coordinate registers provide a great deal of flexibility for applications programming for the Model One. One of the most powerful uses is in passing coordinate parameters in macro calls.

2.9 Clipping Window

The Model One supports a clipping window for clipping of vectors, graphics primitives, and images to any prescribed window in image memory. Figure 2.1 illustrates the use of the clipping window to keep vectors from "spilling over" into neighboring areas of the image.

The clipping window corners may be defined by direct modification of the appropriate coordinate registers, or with the firmware command WINDOW.

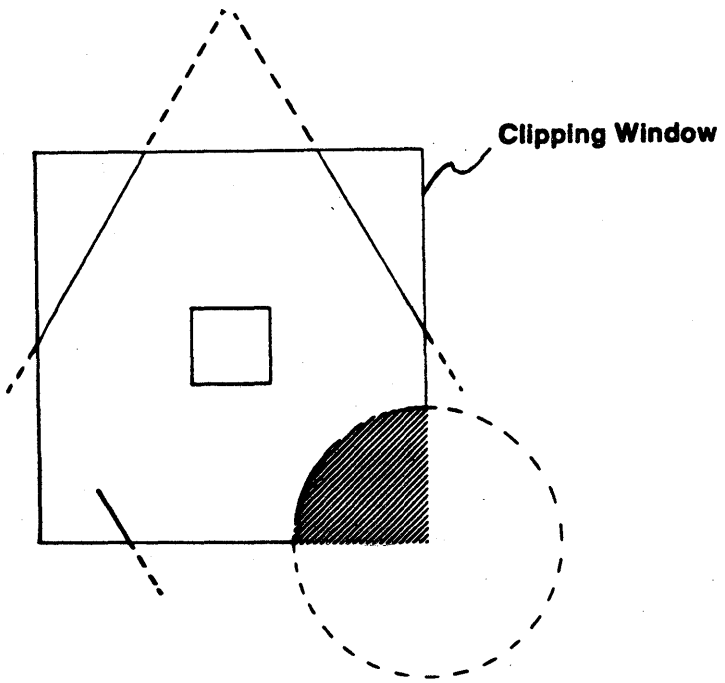


Figure 2.1 Use of the Clipping Window

2.10 Graphic Primitive and Area Fills

The Model One supports filling of graphics primitives and two kinds of area fills. A single command, PRMFIL, allows the user to select whether subsequent graphics primitives should be drawn filled or unfilled. The vector pattern register (as controlled by the VECPAT command) also controls filling of graphics primitives. Figure 2.2 shows filling of a polygon (drawn with the Model One's POLYGN command).

In an area fill, a seed point and a boundary condition are defined. The two firmware commands AREA1 and AREA2 expect differing types of boundaries. Every pixel within the boundary is set to the desired pixel value when the area fill is performed. Figure 2.3 shows an area fill.

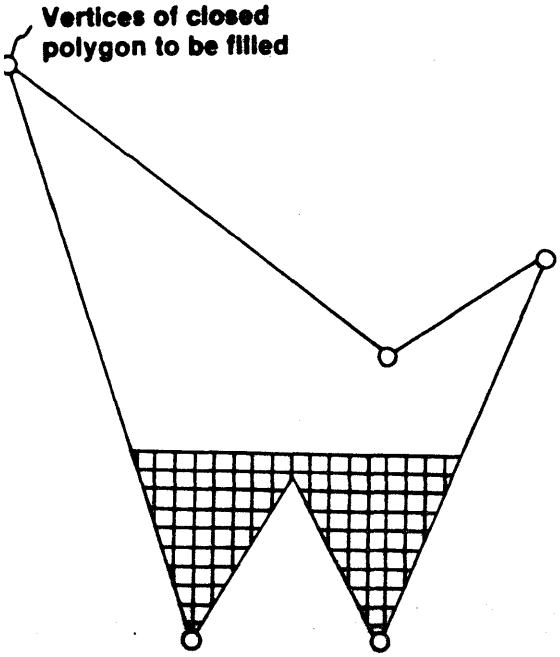


Figure 2.2 Polygon Fill

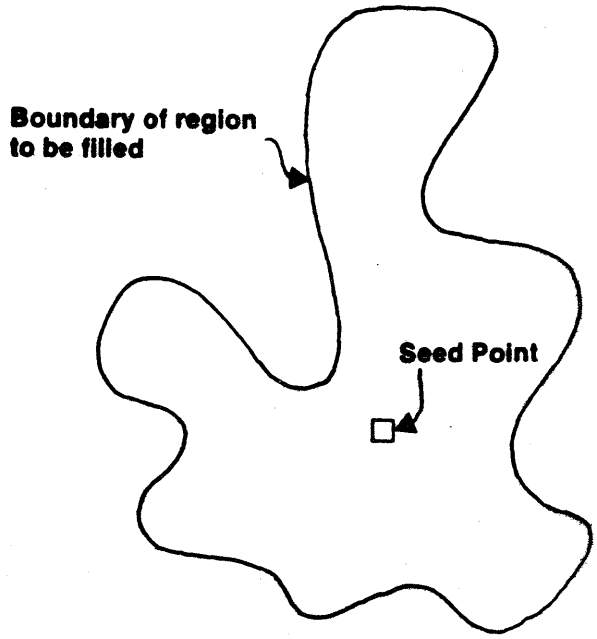


Figure 2.3 Area Fill

2.11 Pixel Value Registers

The Model One has sixty-four pixel value registers (VREGs) for the manipulation of image data values. Some of these registers are reserved for Model One use; the rest are available for programmer use. A set of firmware commands is available for loading the value registers (VLOAD), and for adding (VADD), subtracting (VSUB), and copying/moving (VMOVE) between pairs of value registers.

Table 2.3 summarizes the use of the value registers.

Value Register	Use
VREG 0	The current pixel value; this is the value used by all commands that write graphic data to the Model One.
VREG 1	Crosshair 0 pixel value.
VREG 2	Crosshair 1 pixel value.
VREG 3	Fill mask used for seeded area fills.
VREG 4*	Color assignment for overlay plane 0.
VREG 5*	Color assignment for overlay plane 1.
VREG 6	Reserved.
VREG 7-15	Available for temporary value storage.
VREG 16-42	Foreground, background, and cursor colors for alphanumeric windows.
VREG 43-50	Reserved.
VREG 51-63	Available for temporary value storage.

*For Model One/25 Option Card users only.

Table 2.3 Value Register Assignments

Like the coordinate registers, the pixel value registers allow a great deal of flexibility in applications programming for the Model One. One of the most powerful uses is in passing value parameters in macro calls.

2.12 Macro Commands: Definition and Execution

The Model One includes macro programming capability. A group of commands may be defined and stored in the Model One, to be executed whenever desired. Up to 256 macros may be stored at any time. A macro is created with the firmware command MACDEF, followed by a string of Model One commands. Any Model One command--including defining or executing another macro--can be included in a macro, with the exception of the two commands ASCII and QUIT. A macro is ended with the command MACEND.

Macros can be nested up to eight deep.

Using the Model One's BUTTBL command, it is possible to set up macros to be executed in response to buttons on the cursor that are pushed by the user. The local debugger allows macros to be downloaded from the host and then rewritten locally for testing, if desired.

Explicit MACRO commands, sent from the host, from the local alphanumeric terminal, or from another macro, can also be used to execute macros. Parameters can be passed between macros by using the Model One's coordinate and pixel value registers for storage of coordinates and pixel values.

2.13 Interactive Operations

Many interactive applications, such as dragging, rubber-banding, and menuing, require a group of commands to be executed repeatedly, or for commands to be executed in response to a function button of some kind. The Model One's button table supports these applications.

The Model One button table has 64 entries, from 0 to 63. Each entry indicates a macro to be executed in response to a function button pressing or in response to an explicit BUTTON command. If no button is pressed, the entry at button table location 0 is executed; this macro will execute every 1/30th second and can be used for such functions as cursor tracking.

The button table entries can be modified by macros executed in response to button pressing, allowing virtually unlimited flexibility in interactive support. For example, a tree of menus can be set up easily, since macros can also define other macros and execute them.

When combined with the Model One's macro command facilities, the button table allows many applications that used to require host intervention to be done entirely within the Model One, thus off-loading the host computer.

2.14 Alphanumeric Scrolling Windows

The Model One Advanced Graphics Application Development Firmware includes a set of firmware commands for emulation of multiple-windowed editing terminals. Up to five independently-scrolling windows are provided. The size of text within the window, cursor position, scrolling, wraparound, bolding of text, and overstriking of text are all firmware controllable.

2.15 Display List Firmware

The otptional Display List Firmware offers a fully compatible two-dimensional display list management capability for the Model One. The Display List Firmware package performs local scaling, rotation, and translation of a two-dimensional graphics database which includes primitives such as circles, polygons, and lines.

The user works in a 64,000 by 64,000 world coordinate system with a segmented display structure. Up to eight views into the world coordinate system can be simultaneously defined and displayed; each has its own viewpoint and independent scaling, translation, and rotation. Pickability and visibility of segments are easily controlled.

3.0 FORTRAN LIBRARY

The Model One host FORTRAN library, called ONELIB, gives the programmer access to all of the Model One commands through subroutine CALLs from the host application program.

To send any command from the host to the Model One, the programmer issues a CALL to a ONELIB subroutine. All ONELIB command subroutines have the same name as the local command mnemonic. For example, these FORTRAN lines:

```
CALL MOVABS (0,0)
CALL CIRCLE (100)
CALL DRWABS (20,50)
```

are identical to typing these commands at the local terminal:

```
MOVABS 0,0
CIRCLE 100
DRWABS 20,50
```

The FORTRAN library contains several levels of subroutines. The subroutines which generate Model One commands are called by the application program; those subroutines in turn call low-level subroutines to perform I/O between the host and the Model One.

Each Model One command has an equivalent FORTRAN subroutine. For example, the FORTRAN call for the MOVABS command is:

```
CALL MOVABS (IX,IY)
```

IX and IY are INTEGER*2 variables; the order for parameters is the same as for locally-typed commands.

For all FORTRAN subroutines, these conventions are used: parameters are given in the same order as for locally-typed commands; they are always INTEGER*2 (ranging from -32,768 to 32,767); and they are never changed by the FORTRAN call.

4.0 CENTRAL PROCESSOR

The Model One central processor is a Z8002 16-bit microprocessor. The Z8002 processor is a powerful general purpose microprocessor with 16 and 32 bit logic and arithmetic capabilities, including multiply and divide instructions<1>.

The central processor runs a ROM firmware program which:

1. decodes incoming graphics commands,
2. provides interactive device supports,
3. maintains input/output buffers, and
4. controls display attributes, such as zoom and pan.

The central processor reads from and writes to image memory through memory-mapped registers which lie in its address space. Up to six serial ports, an optional IEEE-488 port, and an optional DMA port are provided for external communications. A set of firmware commands is used to set up these ports as desired for host communications, local terminal use, interactive device support, and error communications.

Figure 4.1 diagrams the central processor.

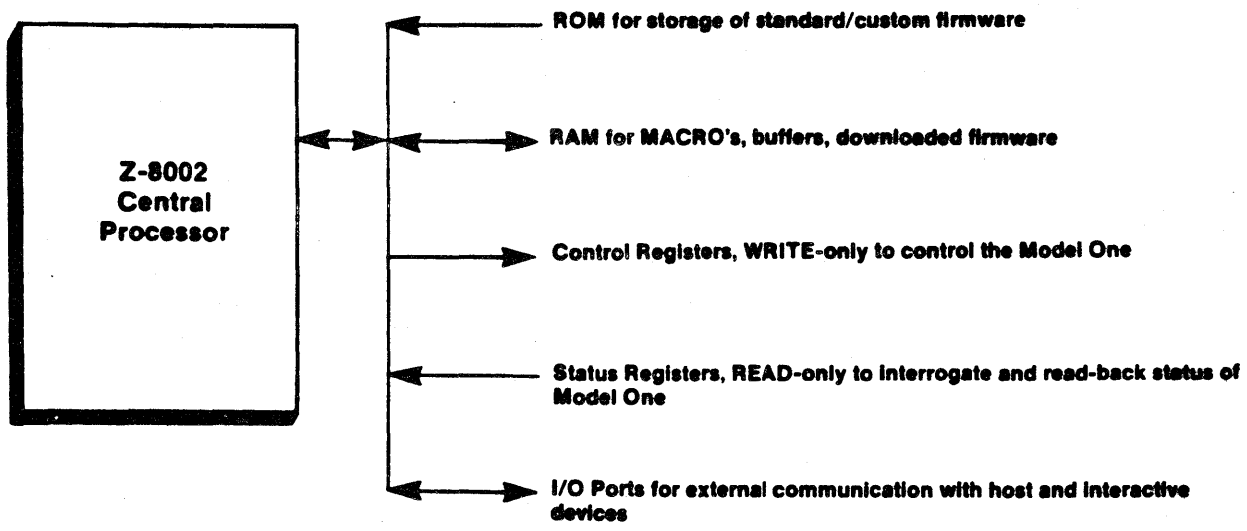


Figure 4.1 Central Processor Addressing and I/O

<1>A detailed description of the Z8002 instruction set and features can be found in AmZ8001/2 Processor Instruction Set, available from Advanced Micro Devices.

4.1 Processor Memory

The Z8002 has a direct addressing range of 64K bytes and a memory mapping scheme for access to a maximum of 128K bytes of physical memory. This expanded memory space is available to users for macro storage, configurable buffers, and user-written firmware. Its memory map is divided into three parts: ROM, RAM, and Control/Status registers.

4.2 Input/Output Ports

The central processor communicates with external devices through six serial ports, an optional IEEE-488 interface, and an optional DMA interface. These ports are configured, using firmware commands, for host communications, local terminal support, interactive device support (such as a tablet), and error message communications.

Table 4.1 summarizes the available ports.

Port	Type	Description
1-6	RS-232C	Host, alphanumeric terminal, tablet, keyboard--serial ports to 38.4KBaud; one of these ports is factory-selectable to RS-232C, RS-422, or TTL (for keyboard support). All ports configured with firmware commands.
HOSTGPIB	IEEE-488	Optional host byte-parallel interface to 20 Kbytes/second.
HOSTDMA		Optional host DMA port.

Table 4.1 Central Processor I/O Port Assignments

All serial ports are firmware configurable for: RTS, CTS, parity, baud rate (from 75 baud to 38.4 Kbaud), queue size, XON/XOFF protocol, and control character acceptance. In the standard firmware, all serial input and output is interrupt-driven, with firmware configurable buffers to maximize processor utilization.

The optional IEEE-488 interface implements all of the listener functions, as described in IEEE standard 488-1978 Digital Interface for Programmable Instrumentation.

The optional host DMA interface is available for DEC DR11V, DRW11B, and other host computers.

4.3 Control and Status Registers

The central processor uses Control/Status registers for its internal communications with the Model One. Control registers are write-only locations that cannot be read back; Status registers are read-only locations and cannot be changed by a write operation.

The Control registers are grouped into three major subsystems: Vector Generator, Pixel Processor, and Display Control. Three groups of Status registers interrogate each subsystem and monitor its operation.

5.0 VECTOR GENERATOR

A hardware vector generator is used to off-load digital line generation from the central processor. The Model One's vector generator is a custom dedicated processor which performs vector-to-raster conversion. Using control registers, the central processor sends the vector generator the start-point, end-point, and pixel value information. The start-point and end-point give the address of the pixels in image memory which are to be connected with a digital line; the pixel value data indicates the value to be placed in image memory at every pixel along the line. Each vector requires 1.6 microseconds of hardware set-up time. Each pixel along a vector takes as little as 400ns to compute and write into image memory, giving the Model One exceptional high-speed vector writing capability.

The vector generator has the image memory available to it approximately 27% of the time. As a result, the vector generator has an average pixel writing rate of roughly 1.45 microseconds, for a total of 700,000 pixels per second. With the screen blanked (using the firmware command BLANK), the pixel writing rate is about one-third higher, for a total of over one million pixels per second.

The central processor requires about 70 microseconds to execute a DRAW command. This time is often completely overlapped with vector drawing by using the hardware vector generator. The bottleneck in vector drawing thus depends on the average length of each vector being drawn; in any case, the Model One can process over 12,000 "average" vectors per second.

5.1 Vector Queue

A hardware first-in, first-out (FIFO) buffer queue resides between the central processor and the hardware vector generator. This queue stores up to 240 vectors waiting to be drawn by the vector generator. The Model One's performance is enhanced by this queue, which increases the utilization of the central processor and the vector generator. In addition, this vector queue keeps vector set-up time to a minimum. Vector generator status registers give the central processor the ability to detect queue-empty and queue-full conditions.

For added flexibility, the central processor can inhibit the flow of vectors out of the vector queue under program control, using the firmware command VGWAIT. This capability is useful in real-time applications, where it can be used to synchronize vector writing with vertical blanking interrupts to eliminate tearing while moving objects.

5.2 Pattern Register

The vector generator provides patterned lines, such as dashed or dotted lines. The firmware command VECPAT accesses a 16-bit control register which is used to provide the line patterns. The same line pattern can also be used for primitive and area fills.

At every pixel along the register, the pattern register is left-shifted by one bit. If the shift causes a carry to occur, the pixel currently being processed is written into image memory normally; if no carry occurs, the pixel is inhibited. The carry bit is shifted into the rightmost bit of the

pattern register so that the bit pattern recycles every sixteen pixels. Because the pattern register is not reinitialized after each vector, the pattern will continue properly from one vector to the next.

5.3 Concatenated Vectors

The vector generator writes digital lines from the start-point to the end-point, including the points at both ends. If a series of vectors represents concatenated line segments, the common point (the start of the new line and the end of the old line) of the two segments will be written twice into image memory. If the pattern register (VECPAT command) or pixel processor (PIXFUN command) is being used, this may create undesired patterns. To prevent these undesired effects, the firmware command FIRSTP can be used to inhibit writing of the first pixel (start-point) of any vector.

6.0 PIXEL PROCESSOR

Many applications require arithmetic and logical operations to be performed between incoming image data values and those which are already in the image memory. For example, the pixels of an incoming image can be added together with the pixel values which comprise a displayed image, thus adding the first image to the second.

In many graphics systems, these pixel functions must be performed in program software, adding significant overhead. In the Model One, a hardware pixel processor is standard, providing the most useful of these pixel functions at high speed, with no central processor overhead. Both the vector generator and host DMA port can write into image memory using the pixel processor.

6.1 Pixel Processor Operation

The vector generator or the DMA port give the pixel processor an image memory address and a pixel value as input. The pixel processor uses the input address to read from image memory; the value read from image memory and the input data value are then input to ALUs (Arithmetic/Logic Units) which perform the desired pixel function or logic operation.

The output from the ALUs is written into image memory on the next available image memory cycle, for a total processing time of 800ns.

Figure 6.1 illustrates the data path within in the pixel processor.

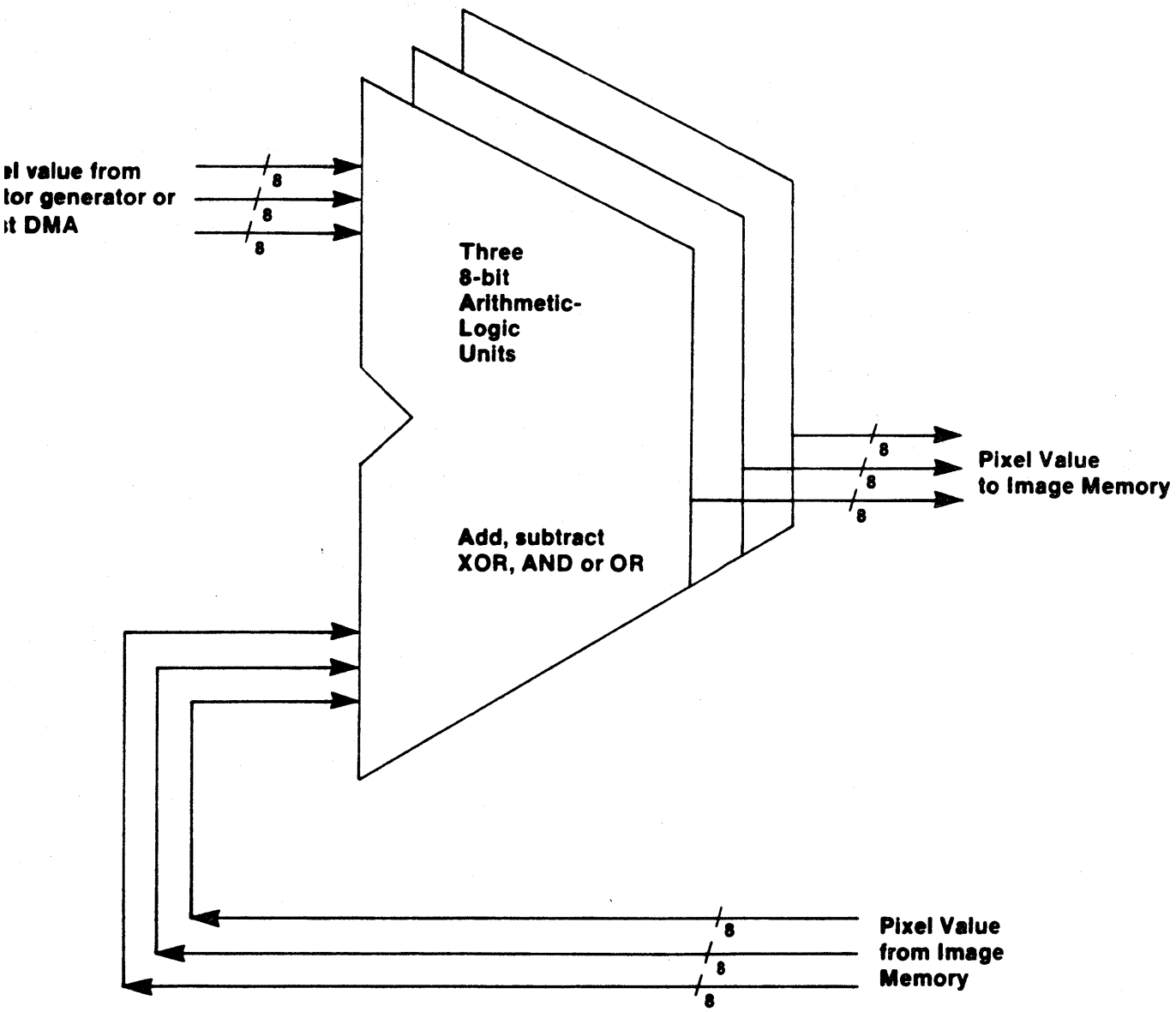


Figure 6.1 Pixel Processor Data Paths

6.2 ALU Functions

The pixel processor ALUs support logical XOR, AND, and OR functions, and PRESET (write all 1's), and CONDITIONAL (do not write values of 0,0,0) functions for the Model One/25, Model One/40, and Model One/60. For the Model One/25 only, addition and subtraction functions are supported as well, in the 512x512 addressing mode. (In the 1024x1024 addressing mode, the arithmetic functions are not available.)

The logic functions provide easy solutions to many CAD-related problems. For example, the XOR mode can be used to provide a "write-through" function for rapid non-destructive dragging of sub-pictures and objects. The arithmetic functions are most useful in imaging applications to add and subtract intensity values between incoming data and data already in image memory.

The pixel processor function is selected through the firmware command PIXFUN.

6.3 Data Value Clipping

When using the Model One/25 in add or subtract mode, an overflow or underflow from the ALU is possible. The firmware command PIXCLP allows the user to select between wrap-around or clipping of the ALU output on overflow and underflow.

7.0 IMAGE MEMORY

The image memory of the Model One consists of up to 768K bytes of dynamic RAM, available in different configurations. The Model One/25 is available with up to 24 512x512 bit planes; the Model One/40 and Model One/60 are available with up to 6 1024x1024 bit planes. The Mod[illegible]25 of course, can be addressed in 1K mode, where the memory is [illegible]1024x1024 bit planes.

The Model One uses 64K RAM chips to allow a[illegible] contained on a single printed circuit board. Image m[illegible] minimum of 128K bytes (4 bits per pixel at 512x5[illegible] and with a minimum of 256K bytes (2 bits per pixel) [illegible] and Model One/60. The maximum image memory is 768K [illegible] at 512x512, 6 bits per pixel at 1024x1024).

BILL HERBLIN GLENOLDEN 237-7705

7.1 Image Memory Organization

In the Model One/25, image memory is or[illegible]r 1024x1024x6 array, selectable with the firmware co[illegible]age array is divided into three banks, each of which [illegible]el in 512 mode or 2 bits per pixel in 1K mode. The[illegible]d to store the red, green, and blue components of a fu[illegible]el image, or to store multiple independent pseudo-color

In the Model One/40 and the Model One/60, [illegible]s organized as 1024x1024x6 array, allowing 6 bits per pixel, for up [illegible] colors. In the Model One/40, all of image memory can be displayed; in the Model One/60, a portion of image memory is off-screen and may be used to store symbols, fonts, etc.

7.2 Write-Enable Masks

Each bit-plane and each bank of image memory can be selectively write-protected or write-enabled by using write-enable masks. A set of control registers gives the central processor access to the write-enable masks. The firmware command WRMASK allows the programmer to modify the write masks.

7.3 Read Masks

The output from image memory passes through a set of read masks which can force the output of any bit plane to zero. These masks, which may be modified with the firmware command RDMASK, are useful in double-buffering and other real-time applications.

7.4 Memory Bandwidth

Each memory cycle is under 400 ns. During this time, any location in image memory may be written into or read from. In normal operation, approximately 73% of all image memory cycles is used for screen refresh, leaving 27% for use by the vector generator, pixel processor, and host DMA. The memory bandwidth with normal screen refresh enabled is thus about 2.4M bytes per second and about 7.5M bytes per second with the screen blanked.

The firmware command BLANK allows the programmer to enable and disable screen refresh as desired.

8.0 IMAGE MEMORY OUTPUT

In the Model One/25 using 512x512 addressing mode, the output of image memory is routed into three color Look-Up-Tables (LUTs). The LUTs are used to drive eight-bit digital-to-analog converters (DACs) for each primary color, as shown in Figure 8.1.

A set of firmware commands is available to control the output from the LUTs: LUTR, LUTG, and LUTB control the red, green, and blue LUTs; LUTA modifies all three LUTs; LUTRTE can be used to modify the routing into the Look-Up-Tables; and LUTRMP can be used to modify the default linear ramping of the Look-Up-Tables.

In the Model One/25's 1024x1024 addressing mode, no Look-Up-Tables are used, and two bits per bank are used to drive the DACs directly, giving 64 simultaneously displayable colors.

In the Model One/40 and the Model One/60, the output of image memory is six bits, which is routed into a single Look-Up-Table with sixty-four entries. This Look-Up-Table can be modified with the same commands used for the Model One/25; the LUTRTE command is not applicable, however.

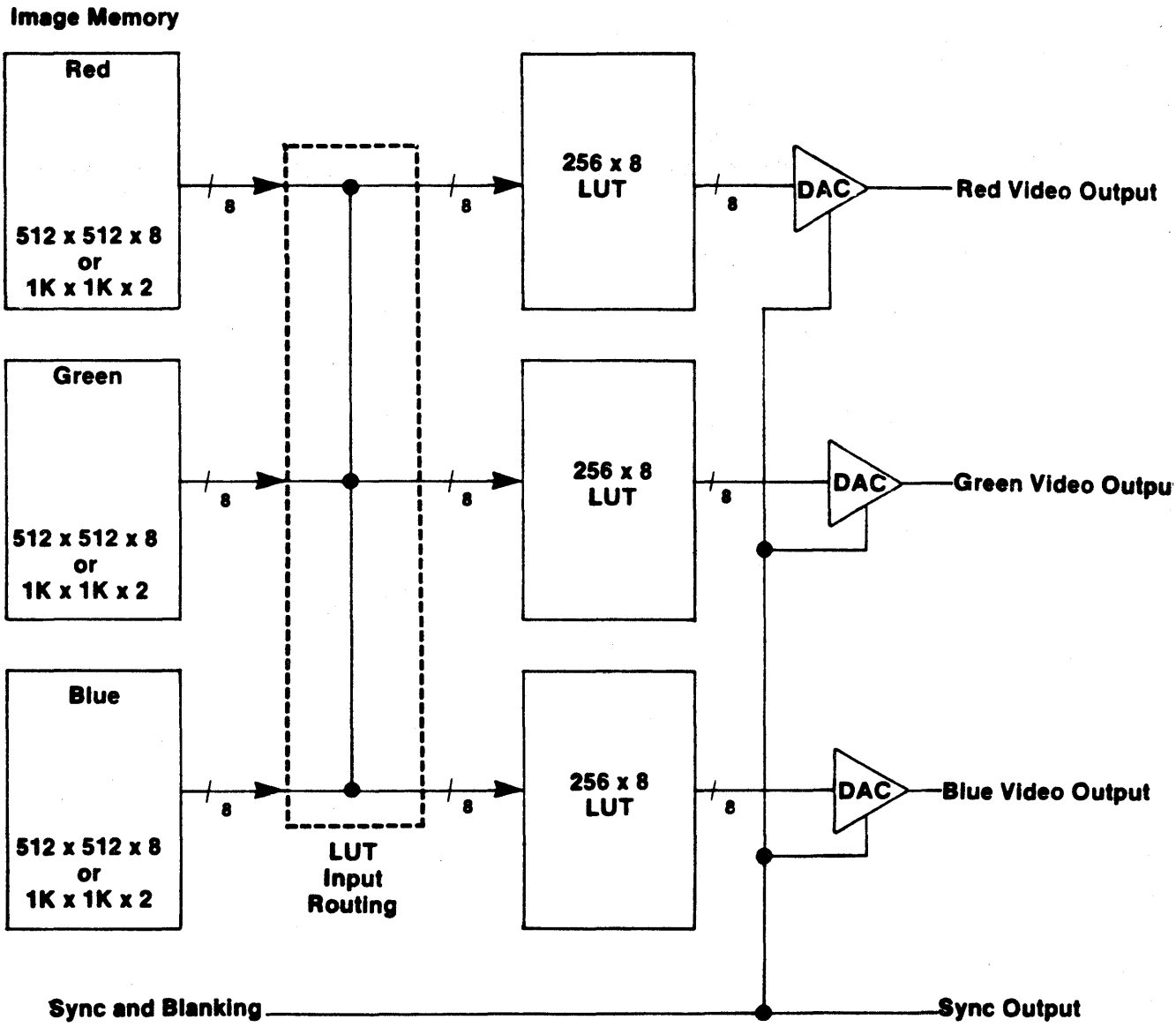


Figure 8.1 Image Memory Output Diagram for the Model One/25

8.1 Look-Up-Tables

In the Model One/25, three 8-bit-in, 8-bit-out LUTs (256x8) are used to drive the video DACs. In a 24 bit per pixel full-color imaging application, the LUTs can be used to provide linearity and color correction. In an 8 bit per pixel pseudo-color application, the inputs to all three LUTs would come from the same image memory bank; alternatively, the 24 bit plane system could be divided into three 8 bit per pixel pseudo-color images, for pseudo-color animation.

In the Model One/40 and Model One/60, the single Look-Up-Table is a 6-bit-in, 24-bit-out Look-Up-Table, with all six bits coming from a single image memory bank.

8.2 Video Amplifiers

A separate video amplifier stage is used for each of the three primary colors: red, green, and blue. A summing amplifier for each channel adds blanking and sync pulses to the video signal and drive a 75 ohm output buffer. The output of this buffer is connected to external BNC connectors for input to a standard color video monitor. Since sync and blanking signals have been added to all three video signals, any or all of the video outputs can be used to drive a monochrome video monitor. In addition, separate composite sync output is provided.

8.3 LUT Input Routing

In the Model One/25, a central processor control register determines the image data routing into the three LUTs. This register may be modified using the firmware command LUTRTE.

By using this register, any bank of image memory can drive no LUT, one LUT, two LUTs, or all three LUTs. This provides greater control for the user in image memory configurations with less than 24 bits per pixel, in addition to providing flexibility for real-time operations such as double-buffering and movie-loop animation.

9.0 DISPLAY CONTROL

The Model One provides pan, zoom, and control of several other display attributes in hardware. The firmware then supports these features through control registers which are accessed by the central processor.

9.1 Screen Origin

A pair of control registers are used to preset the screen refresh address counters in the Model One. These counters control the placement of the screen origin in image memory and are used for panning. The firmware command SCRORG is used to modify these control registers. In the Model One/60, modification of the screen origin can be used to display off-screen memory.

The screen origin can be placed on four pixel boundaries horizontally and two pixel boundaries vertically in the Model One/25; in the Model One/40, sixteen pixel poundaries are used horizontally and two pixel boundaries vertically. Finally, the Model One/60 uses sixteen pixel horizontal boundaries and one pixel vertical boundaries.

Changes to the screen origin are performed only during vertical blanking to eliminate tearing.

9.2 Display Scaling

The Model One supports display scale factors of 1, 2, 4, and 8. These correspond to display windows as shown in Table 9.1.

Scale Factor	Model One/25		Model One/40	Model One/60
	512 mode	1K mode		
1	512x512	1024x1024	1024x1024	768x576
2	256x256	512x512	512x512	384x288
4	128x128	256x256	256x256	192x144
8	64x64	128x128	128x128	96x72

Table 9.1 Display Scale Factors and Resultant Windows

In the Model One/25's 1024x1024 addressing mode (1K mode), each 2x2 array of image memory pixels is intensity averaged to form each screen pixel. This averaging is performed automatically by the Model One/25's hardware. Once the screen is zoomed to a scale factor of two, pixel averaging is no longer necessary.

For the Model One/40 and the Model One/60, no pixel averaging is necessary.

The display scale factor is controlled by two firmware commands: ZOOM and ZOOMIN.

9.3 Flooding the Screen

The Model One can flood all visible pixels to any 24-bit value in a single frame time, synchronized with vertical blanking to eliminate tearing. The firmware command FLOOD is used to flood the screen.

In addition, the screen can be CLEARed (with the firmware command CLEAR) with any desired 24-bit pixel value. The CLEAR command, unlike the FLOOD command, is under the control of the pixel processor and vector pattern register; for example, the value can be ANDed with existing values (PIXFUN command), or patterns created with VECPAT.

10.0 OPTION CARD

The Model One series includes a serial (RS-232C) host interface, hardware vector generator, and pixel processor. In addition, the firmware needed for user support is standard. The Model One family includes the Model One Option Card, which provides several hardware options to enhance the display system. These options are: Direct Memory Access (DMA) port, pixel mover, and--for the Model One/25 Option Card only--two overlay planes.

10.1 Direct Memory Access

The Model One host Direct Memory Access (HOSTDMA) interface allows the user to perform high speed image data and graphic command transfers between the host computer and the Model One. The DMA interface minimizes the overhead for transmission of large amounts of data, providing the fastest possible screen update.

The data passed over the DMA interface may be either graphics commands to be executed by the Model One's central processor, or raw image data to be placed directly into image memory. Use of the DMA interface is transparent; no special commands are needed.

The standard DMA interface is DEC DR11-W and DRV11-B compatible; other interfaces are available.

10.2 Pixel Mover

The Pixel Mover moves a rectangular block of pixels from one location in image memory to another at very high speeds. The pixel processor mode (AND, XOR, etc.) controls the pixels which are transferred, providing additional flexibility. Two firmware commands are used for pixel moves: PIXMOV performs the actual move; PMCTL controls the pixel mover mode. The corners of the window to be moved are set with the Model One's coordinate registers. Judicious setting of the coordinate registers allows mirroring around the X or Y axis during the pixel move.

For the Model One/25, the Pixel Mover allows transfers between the red, green, and blue banks of image memory as well.

10.3 Overlay Planes

Two 512x512 overlay planes are provided on the Model One/25 option card. These planes can be used for alphanumeric display in a single screen workstation or for annotating an independent image. The overlay planes can be panned and zoomed separately from image memory; display of the overlay planes has priority over image memory (they are "in front of" image memory). Finally, each overlay plane may have one of eight possible colors.

A set of firmware commands supports the hardware options: OVRRD sets the display mode for the overlay planes, OVRZM determines the zoom factor, and OVRVAL determines the value to be written into the overlay planes.

The overlay planes also include two crosshairs (in addition to the two crosshairs in the standard firmware) and up-down scrolling for use when the overlay planes are used for alphanumeric display.

11.0 INTERACTIVE DEVICE SUPPORT

11.1 Digitizing Tablet

The Model One supports interactive input devices, such as a digitizing tablet. A set of firmware commands is available to configure the interactive device support for any application.

The Model One's digitizing tablet has an 11"x11" active surface area, and can be used to enter coordinate information and position crosshairs. Three cursor options are available: a simple stylus, a thirteen button cursor, and a sixteen button cursor. The firmware command BUTTBL allows the user to write macros to be executed whenever one of the cursor buttons is pressed; these macros can call other macros, redefine the button table (controlled by BUTTBL), execute one or a series of Model One commands--the macros provide the user with an enormous capability for local programming.

In addition, whether the user is using an interactive device or not, button table location zero may be set. When location zero is set, the macro corresponding to that location executes every 1/30th of a second; thus, the macro at location zero can easily be set for cursor tracking or other such tasks.

APPENDIX I. MODEL ONE FIRMWARE COMMAND SET SUMMARY

The following is a summary of the graphics commands supported by the standard firmware in the Model One product family. Brackets [] indicate the hexadecimal opcode of each command.

HELP	List all command mnemonics.
HELP mnemonic	Give command information.

Graphics Primitives

ARC rad,a1,a2	Draw arc of radius rad. Starting angle is a1; ending angle is a2. [11_H]
AREA1	Area fill. Boundary is any pixel different in value from the current point. The area is filled with current value. [13_H]
AREA2 vreg	Area fill. Boundary pixel value given in vreg. [14_H]
CIRCI creg	Draw circle. Location given by creg lies on the circumference. [10_H]
CIRCLE rad	Draw a circle of radius rad. [$0E_H$]
CIRCXY x,y	Draw circle. Point x,y lies on the circumference. [$0F_H$]
CLEAR	Flood current window to current pixel value. [87_H]
DRW2R dx,dy	Draw vector relative by dx,dy. [84_H]
DRW3R dx,dy	Draw vector relative by dx,dy. [83_H]
DRWABS x,y	Draw vector from current point to the point x,y. [81_H]
DRWI creg	Draw vector to location given by creg. [85_H]
DRWREL dx,dy	Draw vector relative by dx,dy. [82_H]
FILMSK rmsk,gmsk,bmsk	Image data is ANDed with masks before checking value in AREA fill commands. [$9F_H$]

INDEX

FLOOD	Flood displayed image memory to current pixel value. [07]
MOV2R dx,dy	Move relative by dx,dy. [04_H]
MOV3R dx,dy	Move relative by dx,dy. [03_H]
MOVABS x,y	Move absolute location of current point to x,y. [01_H]
MOVI creg	Move to location given by coordinate register creg. [05_H]
MOVREL dx,dy	Move relative by dx,dy. [02_H]
POINT	Set current point to current pixel value. [88_H]
POLYGN npoly, verts	Draw polygons. Npoly gives number of polygons; for each polygon, verts gives number of vertices and the vertices. [12_H]
PRMFIL flag	Primitive fill. Filled primitives are drawn if flag=1. If flag=0, the perimeter of graphics primitives is drawn. [$1F_H$]
RECREL dx,dy	Draw rectangle. Diagonal corner is dx,dy away from current point. [89_H]
RECTAN x,y	Draw rectangle Diagonal corner is point x,y. [$8E_H$]
RECTI creg	Draw rectangle. Location given by creg is diagonal corner. [$8F_H$]
TEXT1 string	Draw text string with font 1. [90_H]
TEXT2 string	Draw text string with font 2. [91_H]
TEXTC size,ang	Specify size of text and draw at angle ang. [92_H]
TEXTDN char,veclst	Define downloaded character in font 2. [26_H]
TEXTRE	Restore default character set. [$B1_H$]
VAL1K val	Set current pixel value (for 1K mode). [$B0_H$]
VAL8 val	Set current pixel value to val,val,val. [86_H]

VALUE r,g,b	Set current pixel value to r,g,b. [06_H]
VTEXT1 string	Vertical text string with font 1. [93_H]
VTEXT2 string	Vertical text string with font 2. [94_H]

Look-Up Table Commands

LUT8 index,r,g,b	Make entry r,g,b at location given by index in Red, Green, and Blue LUTS. [$1C_H$]
LUTA index,entry	Make entry in all LUTs. Place entry at location given in index. [$1B_H$]
LUTB index,entry	Make entry in Blue LUT. Place entry at location given in index. [$1A_H$]
LUTG index,entry	Make entry in Green LUT. Place entry at location given in index. [19_H]
LUTR index,entry	Make entry in Red LUT. Place entry at location given in index. [18_H]
LUTRMP code,sind,eind, sent,ent	Load LUTs with ramp function. [$1D_H$]
LUTRTE+ func	Change LUT routing function specified by func. [$1E_H$]

+Model One/20 and Model One/25 Users Only.

Image Transmissions

PIXEL8 nrows,ncols,val	Pixel by pixel image definition. Pixel values are val,val,val. [29_H]
PIXELS nrows,ncols,r,g,b	Pixel by pixel image definition. Pixel values are r,g,b. [28_H]

RUNLEN nrows, ncols, r,b,b,cnt	Run-length encoded stream. Pixel value is r,g,b. Horizontal count is cnt. [$2A_H$]
RUNLN8 nrows,ncols,val, cnt	Run-length stream. Pixel value is val,val,val. Horizontal count is cnt. [$2B_H$]

Display Control

ASCII flag	Sets host port input as free format ASCII if flag=1, if flag=0 binary. [$9B_H$]
BLANK flag	Blanks screen when flag=1, normal video is restored when flag=0. [31_H]
COLD	Coldstart. Reset the Model One. [FD_H]
CORORG x,y	Loads coordinate origin register with x,y. [37_H]
FIRSTP flag	First pixel on vectors is inhibited when flag=1, uninhibited when flag=0. [$2F_H$]
MODDIS flag	Select display mode. 512 mode is selected if flag=0, 1K mode is selected if flag=1. [$2C_H$]
MODE1K func+	Select output data routing in 1K mode. [$2D_H$]
OVRRD*+ plane,flag	Display specified overlay plane when flag=0, inhibit display when flag=1. [BA_H]
OVRVAL*+ plane,flag	Set bits to 1 in specified overlay plane when flag=1, reset bits to 0 when flag=0. [$B9_H$]
OVRZM*+ plane,flag	Display plane at scale factor 1:1 if flag=0, display at same scale fact as image memory if flag=1. [$B8_H$]
PIXCLP flag	Pixel processor clipping status. Clip on over/underflow flag=1. [2]
PIXFUN mode	Set pixel processor mode. All vectors and DMA writes are affected. [2]
PIXMOV*	Initiate pixel mover transfer. Move window specified by CREG 11 and 12 as controlled by CREG 13 and 14. [BB_H]

PMCTL* Ø Ø Ø Ø redrte,greenrte,bluerte	Set pixel mover mode; redrte, greenrte, and bluerte control writing into the red, green, and blue banks. [BF_H]
QUIT	Exit graphics mode. [FF_H]
RDMASK mask	Set Read Mask. All pixel values read from image are ANDed with mask. [$9E_H$]
SCRORG x,y	Set screen origin register to x,y. [36_H]
SPCHAR string	Redefine special characters (ENTER-GRAPHICS, etc.) [$B2_H$]
TEKEM flag	Invoke Tektronix emulator [39_H]
VECPAT mask	Vector generator pattern register is set to mask. [$2E_H$]
VGWAIT frames	Inhibit transfer of vectors from vector queue for frames frame times. [30_H]
WAIT frames	Wait for given number of frame times before continuing command execution. [$3D_H$]
WARM	Warmstart. Reinitialize Model One. [FE_H]
WINDOW x1,y1,x2,y2	Set current window. Defined by diagonal x1,y1 and x2,y2. [$3A_H$]
WRMASK bitm,bankm	Set Write-Enable Mask. Bit planes indicated by bitm and banks indicated by bankm are write-enabled. [$9D_H$]
XHAIR num,flag	Enable/Disable Crosshair number num. If flag=1 enable, if flag=0 disable. [$9C_H$]
ZOOM fact	Zoom by factor of fact=1,2,4 or 8. [34_H]
ZOOMIN	Zoom in by factor of 2. [35_H]

*Option Card Users Only.
+Model One/20 and Model One/25 Users Only.

Special Characters (Default Values)

CTL D	0	ENTERGRAPHICS
CTL P	1	Break

ESC	2	Warmstart
@	3	Line Kill
CTL H	4	Backspace
CTL F	5	ACK
CTL U	6	NACK
CTL X	7	Invoke Debug
CTL S	8	Suspend Communications
CTL Q	9	Resume Communications

FORTRAN Utility Subroutines

Call ENTGRA	Enter Graphics Mode
Call EMPTYB	Empty Buffers
Call SEND1(val)	Send one byte to output buffer.
Call SEND2(val)	Send two bytes (16 bits) to output buffer.

Readback Commands

All Read commands require a 7-bit ASCII ACK.

RDMODE flag	Set read back mode for subsequent READ commands; OFF or Ø reads back ASCII decimal; ON or 1 reads back binary format. [$D3_H$]
READBU flg,cflg	Read button number. If flag=1 wait for next button. If flag=0 send number of last button pushed. If cflg=1 send current digitizing tablet coordinate, if clfg=0 send current joystick/trackball coordinate. [$9A_H$]
READCR creg	Read coordinate register. Send x,y to port in graphics mode. [98_H]
READER	Read number of first error. [38_H]
READF func	Sets pixel readback format. Func specifies format. [27_H]
READP	Read Pixel. Send value of pixel to port in graphics mode. [95_H]
READVR vreg	Read value register. Send pixel value to port in graphics mode. [99_H]
READW nrows,ncols,bf	Read Window. Send values of pixels in window to port in graphics mode. [96_H]
READWE nrows,ncol	Read Window run-length encoded. Send values of pixels in window in run-length encloded form to port in graphics mode. [97_H]

Register Operations

CADD csum,creg	Place result of csum+creg in csum. [$A2_H$]
CLOAD creg,x,y	Load coordinate register creg with x,y. [$AØ_H$]

CMOVE cdst,csrc	Move contents of csrc into cdst. [$A1_H$]
CSUB cdif,creg	Place result of cdif-creg in cdif. [$A3_H$]
VADD vsum,vreg	Place result of vsum+vreg into vsum. [$A6_H$]
VLOAD vreg,r,g,b	Load contents of value register vreg with r,g,b. [$A4_H$]
VMOVE vdst,csrc	Move contents of vsrc with vdst. [$A5_H$]
VSUB vdif,vreg	Place result of vdif-vreg into vdif. [$A7_H$]

Software Development

*	Program comment.
ALPHAO strlen,string	Send text string to local alpha-numeric display. [$B4_H$]
DEBUG flag	Enter/Exit Command Stream Translator. Exit when flag=0, else enter. [$A8_H$]
DELAY factor	Delay transmission of characters. [$B6_H$]
DNLOAD	Download Z8002 object code. String format is Tektronix Hex. [FB_H]
HOSTO strlen,string	Send a text string to the host. [$B5_H$]
NULL	No operation. [$ØØ_H$]
PEEK addr	Display contents of CPU memory. [BD_H]
POKE addr,data	Change contents of addr in CPU memory. [BE_H]
REPLAY	Dump last 32 characters of HOSTSIO input buffer to ALPHASIO port. [BC_H]

Macro Programming

MACDEF num	Define Macro number num is terminated by MACEND command. [$8B_H$]
MACEND	End of Macro definition. [$\emptyset C_H$]
MACERA num	Erase Macro num. [$8C_H$]
MACRO num	Execute Macro num. [$\emptyset B_H$]

Interactive Device Support

BLINKC	Clear blink table. [23_H]
BLINKD lut,index	Disable Blink of specified lut,index. [21_H]
BLINKE lut,index entry1,entry2	Enable Blink of specified lut, index. Use entry 1 and entry 2 as alternate values. [$2\emptyset_H$]
BLINKR frames	Blink rate is frame times. [22_H]
BUTTBL index,nmac	Place Macro nmac in Button Table at location index. [AA_H]
BUTTON index	Execute Macro indicated by Button Table at location index. [AB_H]
FLUSH	Empty function button event queue. [15_H]
RDPIXR vreg	Places value of pixel at current point in specified value register vreg. [AF_H]

Coordinate Register Assignment

CREG 0	Current Point. Starting point of graphics primitives. Updated by a MOVE or DRAW command.
CREG 1	Joystick/Trackball Cursor Location. Current coordinate from the joystick or trackball. Updated automatically.
CREG 2	Digitizing Tablet Cursor Location. Current coordinate from the digitizing tablet. Updated automatically.
CREG 3	Coordinate Origin. Coordinate of the

	center of image memory.
CREG 4	Screen Origin. Coordinate of the pixel in the center of the screen.
CREG 5	Crosshair 0 location in Image Memory.
CREG 6	Crosshair 1 Location in Image Memory.
CREG 7*+	Crosshair 2 Location in Image Memory.
CREG 8*+	Crosshair 3 Location in Image Memory.
CREG 9	Clipping Window Origin. Lower left corner of current clipping window. All vectors are clipped to this window.
CREG 10	Clipping Window Origin. Upper right corner of current clipping window. All vectors are clipped to this window.
CREG 11,12*	Diagonal corners for PIXMOV command source window definition.
CREG 13*	Defines start corner for PIXMOV destination window.
CREG 14*	Controls direction of pixel writing for PIXMOV destination window.
CREG 15*+	Screen origin of overlay plane Ø.
CREG 16*+	Screen origin of overlay plane 1.
CREG 17-19	Reserved for future definition.
CREG 20-63	Unassigned. Available for temporary coordinate storage.

*Option Card Users Only.
+Model One/20 and Model One/25 Users Only.

Value Register Assignments

VREG 0 Current Value	The value used in all graphics primitives commands.
VREG 1	Value used for crosshair 0.
VREG 2	Value used for crosshair 1.
VREG 3	Fill Mask used for Area fills.

VREG 4*+	Color assignment for overlay plane Ø.
VREG 5*+	Color assignment for overlay plane 1.
VREG 6	For future definition.
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	For future definition.
VREG 51-63	Available for temporary value storage.

*Option Card Users Only.
+Model One/20 and Model One/25 Users Only.
**Advanced Graphics Development Firmware

System Configuration Commands

DFTCFG	Restore all ports to default configurations.
DISCFG	Display current system configurations.
SAVCFG	Save configuration set with SYSCFG.
SYSCFG HOST	[ASCII/BINARY]
SYSCFG IEEE	[address] [NORMAL] [TALK] [LISTEN]
SYSCFG SERIAL	[port-mnemonic] [RTS on/off] [CTS on/off] [STOP 1/2] [BITS 7/8] [PARITY e/o/1/h/n] [BAUD rate] [XIN on/off] [XOUT on/off] [CTRL on/off].

**Advanced Graphics Development Firmware

Default Port Configurations

Port Mnemonic	RTS	CTS	Baud	Parity	XIN	XOUT	CTRL	STOP	NBITS
MODEMSIO	off	off	1200	none	on	off	off	1	8
KEYBSIO	off	off	1200	none	on	off	on	1	8
TABLETSIO	off	off	1200	none	on	off	off	2	8
GRINSIO	off	off	1200	none	off	off	off	2	8
HOSTSIO	off	off	9600	none	off	on	off	2	8
ALPHASIO	off	off	9600	none	on	off	on	2	8

Alphanumeric Terminal Emulation**

ALPHEM** flag
Enables (flag=1 or ON) or disables (flag=0 or OFF) the alphanumeric terminal emulator. Routes text to selected window. [$C2_H$]

BOLD** flag
Enables (flag=1 or ON) or disables (flag=0 or OFF) drawing of bold text. [CC_H]

DEFWIN** window,xl,yl,x2,y2 xsize,ysize,bitm,bankm
Defines size and position of indicated window number. (xl,yl) defines first corner; (x2, y2) defines diagonal corner. xsize, ysize define text size; bitm, bankm define write mask for window (see WRMASK command). [$C0_H$]

DELWIN** window
Deletes window. [$C3_H$]

DIRCUR** x,y
Moves cursor to character position x,y within window. [$C4_H$]

GETCUR**
Returns Model One coordinates of cursor in currently-selected window. [$C9_H$]

GETPOS**
Returns character position of cursor in currently-selected window. [$C5_H$]

GETWIN**
Returns number of active window (-1 for no active window). [CE_H]

HOME**
Moves cursor to character position (0,0), the upper-left corner of the window. [CF_H]

MOVCUR** x,y
Moves cursor to Model One coordinate x,y within window limits. [$C8_H$]

OVRSTK** flag
Enables (flag=1 or ON) or disables (flag=0 or OFF) overstriking of text. [CD_H].

SCROLL** flag
Enables (flag=1 or ON) or disables (flag=0 or OFF) scrolling of text.

SELWIN** window
Select window as defined by DEFWIN. Sets routing for ALPHEM command. [$C1_H$]

SETCUR** flag
Enables (flag=1 or ON) or disables (flag=0 or OFF) cursor. [$C7_H$]

SETSIZ** xscale,yscale	Sets x,y scaling (multiples of 16 pixels). Default is (1,1).[$C6_H$]
WRAP** flag	Enables (flag=1 or ON) or disables (flag=0 or OFF) wraparound of text.[CB_H]

**Advanced Graphics Development Firmware

Display List Firmware++

DEFVW view,wcsregl,wcsreg2 dcsregl,dcsreg2, upregl,upreg2,rotate, xform,backvreg,bitm, bankm,hiseg	Defines the view into the World Coordinate System view gives the view number; wcsregl and wcsreg2 define the WCS corners; dcsregl and dcsreg2 define the display viewport; upregl and upreg2 give the ends of the UP vector; rotate defines the WCS window rotation center; xform specifies the transformation type; backvreg gives the background color; for bitm and bankm, see the WRMASK command; hiseg specifies the segment nesting. [EBH]
DELPID	Delete primitives with PIDREG and
HILITE view,flag,vreg	SEGREG. [ECH] Highlight view in the color specified by vreg. flag=1 or ON highlights primitives; flag=0 or OFF draws primitives normally. PIDREG and SEGREG (see SET) are used. [DFH]
PICKCR view,dcsreg, searchflag	Perform pick search; view is picked, using the dcsreg coordinates. searchflag=1 or ON, search from current tree location; searchflag=0 or OFF, search from the top of the segment. [E3H]
RDPID	Reads and returns the graphics primitives with PIDREG AND SEGREG (see SET). [EDH]
RDREG	Reads and returns the current SEGREG and PIDREG (see SET and PICKCR). [E0H]
RDTREE	Returns the PICKCR hierarchical history. [EEH]
REDRAW view,flag	Redisplay view; flag=1 or ON clears the window to the background color before display. view=-1 redisplays all views. [E2H]
SEGAPP segment	Open segment to append graphics primitive commands. SEGEND ends the append. [DBH]
SEGCOP segment2,segmentl	Copies segmentl into segment2. [EAH]
SEGDEF segment	Begins definition of segent. [DCH]
SEGDEL segment	Deletes segment [DEH]

SEGEND	End segment definition begun with SEGDEF or SEGAPP. [DDH]
SEGINI words	Initializes Display List Firmware; words gives the number of data words per segment block. [E1H]
SEGINQ segments	Returns the attributes of segment. (See SETATR). [E5H]
SEGPID pickid	Defines the pickid for primitives within segment until next SEGPID or SEGEND command. [D9H]
SEGREF segment	Nest specified segment within current segment. [D8H]
SEGREN segment2,segment1	Renames segment1 to segment2. [DAH]
SET global,value	Set PICKAP aperture; set PIDREG value; set SEGREG value. [46H]
SETATR segment,attribute, flag	Sets VISibility or PICKability of segment. flag=1 or ON enables attribute (VIS or PICK); flag=0 or OFF disables attribute. [F6H]
SYSTAT	Returns system memory usage and availability. [E4H]

++Optional firmware for Model One/25, Model One/40, and Model One/60