HOSTO HOSTO

SYNTAX

ASCII	HOSTO string
FORTRAN Call	CALL HOSTO (STRLEN, STRING)
Binary	[181] [strlen] ([char1][char2]...[charn])
	181 decimal = 265 octal = B5 hex

FUNCTION

The HOSTO (HOST Output) command outputs a text string to the host computer over the serial host interface. The HOSTO command is not designed to send a text string to the DMA interface.

The text is specified by string. If the command is being entered in ASCII mode from the local alphanumeric terminal or keyboard, the string to be output is the set of ASCII characters remaining on the command line. If the command is not being sent in ASCII mode, then the first byte of string contains the number of characters in the string (strlen) followed by strlen bytes containing the ASCII characters to be drawn.

Nonprintable characters can be included in HOSTO text strings. Characters with the high bit set or certain control characters (e.g., CTRL-S) can be put into string arguments in ASCII mode by using the backslash (\) to indicate that the immediately following characters are the numeric value of the desired character:

\ddd Indicates 3 decimal digits representing the value of the desired character.

\#hh Indicates the 2 hexadecimal digits representing the desired character.

\\ Indicates the backslash character (\) itself.

ASCII PARAMETER

string The text to be printed.

HOSTO HOSTO

FORTRAN PARAMETERS

INTEGER*2 STRLEN, STRING(1)

STRLEN is an integer specifying the number of characters that are to be output.

STRING is an integer array with two characters packed per 16-bit word, as in FORTRAN A2 format.

EXAMPLE

```
! HOSTO ABCDEF 1 2 3     ; Output text string "ABCDEF 1 2 3" to
                            the host.

! HOSTO WXYZ             ; Output text string "WXYZ" to the host.
```

IGNORE IGNORE

SYNTAX

ASCII	IGNORE flag
FORTRAN Call	CALL IGNORE (FLAG)
Binary	[124] [flag] (2 bytes)
	124 decimal = 174 octal = 7C hex

FUNCTION

The IGNORE command sets the display controller to ignore (flag = 1 or ON) or process (flag = 0 or OFF) subsequent incoming commands. With IGNORE ON, commands are interpreted but graphics primitives are not drawn, picked, highlighted, or unhighlighted, according to the current execution mode set with the EXMODE command.

Note: This command is intended only for use in systems without local display lists. Therefore, no example is given here.

ASCII PARAMETER

flag — Flag = 1 or ON, primitives are not drawn, picked, highlighted, or unhighlighted.

Flag = 0 or OFF, primitives are either drawn, picked, highlighted, or unhighlighted, according to the current execution mode.

FORTRAN PARAMETER

INTEGER*2 FLAG

INCPID INCPID

SYNTAX

ASCII	INCPID	
FORTRAN Call	CALL INCPID	
Binary	[215]	(1 byte)
	215 decimal = 327 octal = D7 hex	

FUNCTION

The INCPID command increments the current pick ID number by one.

EXAMPLE 1

```
! SEGDEF 1              ; Begin definition of Segment 1.
$ PICKID 50             ; Assign pick ID 50 to the following commands.
$   .
$   .
$ INCPID                ; Assign pick ID 51 to the following commands.
$   .
$   .
$ INCPID                ; Assign pick ID 52 to the following commands.
$    .
$    .
$ SEGEND                ; End definition of Segment 1.
```

EXAMPLE 2

In this example, the INCPID command is used in the definition of Segment 20. Each time Segment 20 is referenced by Segment 10, the pick ID in Segment 20 is incremented by 1.

```
! SEGDEF 10             ; Begin definition of Segment 10.
$ PICKID 1              ; Assign pick ID 1 to the following commands.
$ MOVABS -100,0         ; Move current point to position circle 1.
$ SEGREF 20             ; Nest Segment 20.
$ MOVABS 100,0          ; Move current point to position circle 2.
$ SEGREF 20             ; Nest Segment 20.
$ SEGEND                ; End pick ID 1; end definition of Segment 10.

! SEGDEF 20             ; Begin definition of Segment 20.
$ INCPID                ; Increment pick ID by 1.
$ CIRCLE 50             ; Draw circle of radius 50.
$ SEGEND                ; End definition of Segment 20.
```

INSPID INSPID

SYNTAX

ASCII	INSPID	
FORTRAN Call	CALL INSPID	
Binary	[119]	(1 byte)
	119 decimal = 167 octal = 77 hex	

FUNCTION

The INSPID command opens a segment for insertion of graphics primitives. The segment number is contained in SEGREG (segment ID register) and the primitives are inserted following the PICKID or INCPID command whose argument is equal to the quantity contained in the PIDREG (pick ID register).

Any command which is legal in segment definition mode can be inserted until a SECEND command is encountered. An error will be generated if the specified PICKID within the specified segment does not exist.

Note: The SEGREG and PIDREG can be set by picking or using the SETGL or RDICK commands.

KICK KICK

SYNTAX

ASCII	KICK reserved
FORTRAN Call	CALL KICK (RESERVED)
Binary	[75] [reserved] (2 bytes)
	75 decimal = 113 octal = 4B hex

FUNCTION

The KICK command can only be used if a DMA host interface is being used.

The KICK command generates an ATTENTION interrupt. The KICK command waits for the DONE bits to be cleared (DONE = 0), generates the interrupt, and loads the ERREG with FF (hex). The interrupt lasts 1 microsecond.

The KICK command is useful for a polling set-up; for example,including it in a button macro to indicate to the host that a button has been pushed. You should not include the KICK command in the middle of a command stream.

ASCII PARAMETER

reserved Eight-bit parameter reserved for future use; currently it must be set to zero.

FORTRAN PARAMETER

INTEGER*2 RESERVED

RESERVED must be set to 0.

LUT8 LUT8

SYNTAX

ASCII	LUT8 index, r, g, b
FORTRAN Call	CALL LUT8 (INDEX, IRENT, IGENT, IBENT)
Binary	[28] [index] [r] [g] [b] (5 bytes)
	28 decimal = 034 octal = 1C hex

FUNCTION

The LUT8 command changes the color entries in the given index of the LUT to the red, green, and blue values specified by r, g, and b.

An 8-bit system has a single 8-bit-in, 24-bit-out look-up table. A 24-bit system (configured for full color), has three distinct 8-bit-in, 8-bit-out LUTs. In both cases, the same LUT8 command is used.

The color values that are stored are the values that will be loaded into their respective digital-to-analog converters (DACs) when a pixel of value index is encountered when screen refresh is being performed.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
r, g, b	Red, green, and blue entries; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 INDEX, IRENT, IGENT, IBENT

EXAMPLE: 8-Bit System

```
! LUT8 50 255,0,170     ; Set the color out for LUT index 50 to 255,0,170
                          (pink).
! VAL8 50               ; Set current pixel value to 50 (pink).
! FLOOD                 ; Flood displayed pixels to current pixel
                          value; screen turns pink.
! LUT8 50 100,100,100   ; Change the color out for LUT index 50 to
                          100,100,100; screen turns grey.
! LUT8 50 200,100,50    ; Change the color out for LUT index 50 to
                          200,100,50; screen turns orange.
```

LUTA LUTA

SYNTAX

ASCII	LUTA index, entry	
FORTRAN Call	CALL LUTA (INDEX, ENTRY)	
Binary	[27] [index] [entry]	(3 bytes)
	27 decimal = 033 octal = 1B hex	

FUNCTION

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

The entry stored in the LUT is passed to the red, green, and blue digital to analog converters (DACs) when a pixel of value index is encountered when reading from image memory to refresh the display screen.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
entry	The entry at the LUT index; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 INDEX, ENTRY

EXAMPLE: 8-Bit System

```
! LUT8 1 255,255,255    ; Set the color out for LUT index 1 to white.
! VAL8 1                ; Set current pixel value to 1 (white).
! FLOOD                 ; Flood displayed pixels to current pixel value;
                         screen turns white.
! LUTA 1 0              ; Change the color out for LUT index 1 to 0,0,0;
                         screen turns black.
! LUTA 1 100            ; Change the color out for LUT index 1 to
                         100,100,100; screen turns grey.
```

LUTB LUTB

SYNTAX

ASCII	LUTB index, entry
FORTRAN Call	CALL LUTB (INDEX, ENTRY)
Binary	[26] [index] [entry] (3 bytes)
	26 decimal = 032 octal = 1A hex

FUNCTION

The LUTB command changes the blue component of the color value at the look-up table index to the value entry.

The entry stored in the LUT is passed to the blue digital to analog converter (DAC) when a pixel of value index is encountered when reading from image memory to refresh the display screen.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
entry	The blue component entry at the LUT index; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 INDEX, ENTRY

EXAMPLE 1: 8-Bit System

```
! LUT8 1 255,255,255    ; Set the color out for LUT index 1 to white.
! VAL8 1                ; Set current pixel value to 1 (white).
! FLOOD                 ; Flood all displayed pixels to current pixel
                           value; screen turns white.
! LUTB 1 0              ; Set blue component at LUT index 1 to 0;
                           screen turns yellow.
```

EXAMPLE 2: 24-Bit System with RGBTRU ON

```
! VALUE 0 0 100          ; Set current pixel value to r=0, g=0, b=100.
! FLOOD                  ; Flood all displayed pixels to current pixel
                           value; screen turns medium intensity blue.
! LUTB 100 0             ; Change entry at blue LUT index 100 to 0; screen
                           turns black.
! LUTB 100 255           ; Change entry at blue LUT index 100 to 255; screen
                           turns full intensity blue.
```

LUTG LUTG

SYNTAX

ASCII	LUTG index, entry
FORTRAN Call	CALL LUTG (INDEX, ENTRY)
Binary	[25] [index] [entry] (3 bytes)
	25 decimal = 031 octal = 19 hex

FUNCTION

The LUTG command changes the green component of the color value at the look-up table index to the value entry.

The green component entry stored in the LUT is passed to the green digital to analog converter (DAC) when a pixel of value index is encountered when reading from image memory to refresh the display screen.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
entry	The green component entry at the LUT index; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 INDEX, ENTRY

EXAMPLE 1: 8-Bit System

```
! LUT8 1 255,255,255    ; Set the color out for LUT index 1 to white.
! VAL8 1                ; Set current pixel value to 1 (white).
! FLOOD                 ; Flood all displayed pixels to current pixel value;
                           screen turns white.
! LUTG 1 0              ; Set green component at LUT index 1 to 0; screen
                           turns magenta.
```

EXAMPLE 2: 24-Bit System with RGBTRU ON

```
! VALUE 0 100 0          ; Set current pixel value to r=0, g=100, b=0.
! FLOOD                  ; Flood all displayed pixels to current pixel
                           value; screen turns medium intensity green.
! LUTG 100 0             ; Change entry at green LUT index 100 to 0; screen
                           turns black.
! LUTG 100 255           ; Change entry at green LUT index 100 to 255;
                           screen turns full intensity green.
```

LUTR LUTR

SYNTAX

ASCII	LUTR index, entry
FORTRAN Call	CALL LUTR (INDEX, ENTRY)
Binary	[24] [index] [entry] (3 bytes)
	24 decimal = 030 octal = 18 hex

FUNCTION

The LUTR command changes the red component of the color value at look-up table index to the value entry.

The red component entry stored in the LUT is passed to the red digital to analog converter (DAC) when a pixel of value index is encountered when reading from image memory to refresh the display screen.

ASCII PARAMETERS

index	The LUT index to be set; range is 0 to 255.
entry	The red component entry at the LUT index; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 INDEX, ENTRY

EXAMPLE 1: 8-Bit System

```
! LUT8 5 255,255,255    ; Set the color out for LUT index 5 to white.
! VAL8 5                ; Set current pixel value to 5 (white).
! FLOOD                 ; Flood all displayed pixels to current pixel value;
                           screen turns white.
! LUTR 5 0              ; Set red component at LUT index 5 to 0; screen
                           turns cyan.
```

EXAMPLE 2: 24-Bit System with RGBTRU ON

```
! VALUE 100 0 0          ; Set current pixel value to r=100, g=0, b=0.
! FLOOD                  ; Flood displayed pixels to current pixel value;
                           screen turns medium intensity red.
! LUTR 100 0             ; Change entry at red LUT index 100 to 0; screen
                           turns black.
! LUTR 100 255           ; Change entry at red LUT index 100 to 255; screen
                           turns full intensity red.
```

LUTRMP LUTRMP

SYNTAX

ASCII	LUTRMP code, sind, eind, sval, eval
FORTRAN Call	CALL LUTRMP (NUM, ISIND, IEIND, ISENT, IEENT)
Binary	[29] [code] [sind] [eind] [sval] [eval] (6 bytes)
	29 decimal = 035 octal = 1D hex

FUNCTION

The LUTRMP command loads the LUT component(s) specified by code from LUT index sind to index eind with a ramp function linearly interpolated from the start entry sval to the end entry eval. The LUTRMP command is useful whenever multiple, successive LUT entries are to be set to either a ramp function or to a constant value.

On an 8-bit system, the LUT is not sent to a linear ramping function by default, and the LUTRMP command may have somewhat unexpected results.

Note: To reset the LUT to the default, use the COLD command or the MODDIS 1 command, or use a series of individual LUT commands. The COLD command and the MODDIS 1 command both erase the screen.

ASCII PARAMETERS

code	The LUT components to load: code = 1, load blue component code = 2, load green component code = 4, load red component code = 7, load all components
sind, eind	Starting and ending indices in the LUT to load; range is 0 to 255.
sval, eval	Starting and ending entries to load into the indices; range is 0 to 255.

LUTRMP LUTRMP

FORTRAN PARAMETERS

```
INTEGER*2     NUM, ISIND, IEIND, ISENT, IEENT
```

EXAMPLE: 8-Bit System

```
! LUT8 1 255,255,255      ; Set the color out for LUT index 1 to white.
! LUT8 2 200,200,200      ; Set the color out for LUT index 2 to grey.
! LUT8 3 150,150,150      ; Set the color out for LUT index 3 to a darker
                           shade of grey.
! LUT8 4 100,100,100      ; Set the color out for LUT index 4 to a darker
                           shade of grey.
! PRMFIL ON               ; Select filled primitives.
! LUTRMP 7 0 255 0 255    ; Load indices 0 to 255 in all components of
                           LUT with values between 0 and 255 (ramp
                           function).
! VAL8 1                  ; Set current pixel value to 1 (white).
! CIRCLE 100              ; Draw a white circle.
! VAL8 2                  ; Set current pixel value to 2 (grey).
! CIRCLE 80               ; Draw a circle in a light shade of grey.
! VAL8 3                  ; Set current pixel value to 3 (darker gray).
! CIRCLE 60               ; Draw a circle in a darker shade of grey.
! VAL8 4                  ; Set current pixel value to 4 (darker grey).
! CIRCLE 40               ; Draw a circle in a darker shade of grey.
! LUTRMP 7 0 255 255 0    ; Load indices 0 to 255 in all components of
                           LUT with values between 255 and 0 (reverse
                           ramp function). The circles change colors;
                           the outer circle is now black.
! LUTRMP 1 0 255 0 255    ; Load indices 0 to 255 in the blue component
                           of the LUT with values between 0 and 255.
! LUTRMP 2 100 255 0 0    ; Load indices 100 to 255 in the green component
                           of the LUT with 0.
```

MACDEF MACDEF

SYNTAX

ASCII	MACDEF num
FORTRAN Call	CALL MACDEF (NUM)
Binary	[139] [num] (2 bytes)
	139 decimal = 213 octal = 8B hex

FUNCTION

The MACDEF command defines a new macro specified by num. After you enter the MACDEF command from the local terminal, the system displays the prompt $. You can then enter a series of commands which are included in the macro definition, and are not executed until the macro is executed. Macro definition ends when you type the command MACEND.

A macro may include any combination of valid command strings, including nested MACDEF commands and user-defined commands. Up to 16 levels of macros can be nested. Macros cannot contain QUIT or ASCII commands. The length of a macro command is limited only by the available memory space.

ASCII PARAMETER

num The macro number; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 NUM

Note: Readback commands within macro definitions are inserted into the macro definition but the host does not read any data.

EXAMPLE

```
! MACDEF 40          ; Start definition of Macro 40.
$ CIRCLE 50          ; Draw circle of radius 50.
$ CIRCLE 30          ; Draw circle of radius 30.
$ MACEND             ; End definition of Macro 40.
! MACRO 40           ; Execute Macro 40; draw two concentric circles.
```

MACEND MACEND

SYNTAX

ASCII	MACEND
FORTRAN Call	CALL MACEND
Binary	[12] (1 byte)
	12 decimal = 014 octal = 0C hex

FUNCTION

The MACEND command ends a macro definition. If no macro is being defined, an error results. A MACEND command must be used for each MACDEF command.

EXAMPLE

```
! MACDEF 17              ; Start definition of Macro 17.
$ MOVABS 50 50           ; Move current point to 50,50.
$ DRWABS 100, 150        ; Draw a line to 100,150.
$ MACEND                 ; End definition of Macro 17.
! MACRO 17               ; Execute Macro 17.
```

MACERA MACERA

SYNTAX

ASCII	MACERA num
FORTRAN Call	CALL MACERA (NUM)
Binary	[140] [num] (2 bytes)
	140 decimal = 214 octal = 8C hex

FUNCTION

The MACERA command clears macro definition number num. Macro number num cannot be executed after the MACERA command has been issued. Attempting to execute a macro that has been erased has no effect and does not result in any errors.

ASCII PARAMETER

num The macro number; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 NUM

EXAMPLE

```
! MACDEF 23          ; Begin definition of Macro 23.
$ CIRCLE 50          ; Draw a circle of radius 50.
$ MACEND             ; End definition of Macro 23.
! MOVABS 0 0         ; Move current point to 0,0.
! MACRO 23           ; Execute Macro 23; draw a circle with center
                       at 0,0.
! MACERA 23          ; Erase the definition of Macro 23.
! MOVABS 50 50       ; Move current point to 50,50.
! MACRO 23           ; Execute Macro 23; nothing happens because
                       Macro 23 has been erased.
```

MACRO MACRO

SYNTAX

ASCII	MACRO num
FORTRAN Call	CALL MACRO (NUM)
Binary	[11] [num] (2 bytes)
	11 decimal = 013 octal = 0B hex

FUNCTION

The MACRO command executes macro number num. You can also execute macros by pressing or releasing buttons (see the BUTTBL command) or by using the BUTTON command. Executing a macro that has not been defined has no effect and does not result in any errors.

ASCII PARAMETER

num The macro number; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 NUM

Note: If the macro contains readback commands, their interpretation is left up to the application program.

EXAMPLE

```
! MACDEF 23          ; Begin definition of Macro 23.
$ CIRCLE 50          ; Draw a circle of radius 50.
$ MACEND             ; End definition of Macro 23.
! MACRO 23           ; Execute Macro 23.
```

MAP MAP

SYNTAX

ASCII	MAP	
FORTRAN Call	CALL MAP	
Binary	[252]	(1 byte)
	252 decimal = 374 octal = FC hex	

FUNCTION

The MAP command displays the current allocation of Z8002 RAM space for internal storage in the Model One. This allocation is set to a default at COLDstart. You can reallocate memory with the CONFIG command. The MAP output is always sent to the local terminal or to the screen.

The default RAM space allocation is:

Macro definitions	2000 (hex)	8K bytes
Areafill/Polyfill/Zpatch scratch	0800 (hex)	2K bytes
Downloaded text (TEXTDN)	0800 (hex)	2K bytes
Alpha windows	1000 (hex)	4K bytes

EXAMPLE

```
! MAP              ; Display the current configuration (which is the
                     default in this example).

C000  DFFF    Macro definition space
E000  E7FF    Areafill/Polyfill/Zpatch scratch
E800  EFFF    Downloaded text fonts
F000  FFFF    Alpha window area
0000  0000    Downloaded Z8002 code (unused)
0000  0000    Reserved
0000  0000    Reserved
0000  0000    Reserved
```

MEMSEL MEMSEL

SYNTAX

ASCII	MEMSEL memunit
FORTRAN Call	CALL MEMSEL (IUNIT)
Binary	[72] [memunit] (2 bytes)
	72 decimal = 110 octal = 48 hex

FUNCTION

The MEMSEL command selects a memory unit (from 0 to 15) to be addressed by subsequent graphics primitives commands. For example, if memory unit 1 is selected, a subsequent CIRCLE command will draw a circle into the image memory of memory unit 1. The default is memory unit 0.

The RGBTRU command selects a group of three memory units to be addressed together as a 24-bit system. These units must begin with memory unit 0, 4, 8, or 12.

Selection of a non-existent memory unit does not cause an error. You can use the RDCNFG command to show which memory units have been configured.

ASCII PARAMETER

memunit The memory unit; range is 0 to 15. Default is 0.

FORTRAN PARAMETER

INTEGER*2 IUNIT

EXAMPLE

```
! MEMSEL 4              ; Select memory unit 4.
```

MODDIS MODDIS

SYNTAX

ASCII	MODDIS flag
FORTRAN Call	CALL MODDIS (IFLAC)
Binary	[44] [flag] (2 bytes)
	44 decimal = 054 octal = 2C hex

FUNCTION

The MODDIS command resets the Model One. For any value of flag (0, 1, or 2), the MODDIS command resets the following to their default states: look-up tables, value registers, coordinate registers, text size and rotation, blink table, READF, RDMODE, current transformation, tablet scaling factor, and z buffer. The command also resets the states of the RGBTRU and ALPHEM commands to the defaults stored in NVRAM. The MODDIS command clears the screen.

MODDIS 0 and MODDIS 1 both have the same effect, which is to reset the registers and variables listed above. These commands do not change macro definitions, segment definitions, or downloaded text font (Font 2).

MODDIS 2 does clear macro and segment definitions in addition to performing all the functions of MODDIS 0 or MODDIS 1. In effect, MODDIS 2 performs a faster version of COLDstart. Unlike COLDstart, however, MODDIS 2 does not perform diagnostics or reinitialize the Writable Control Store (WCS). MODDIS 2 also does not reset Genlock or the state of the command stream interpreter.

ASCII PARAMETER

flag	Display mode flag; values of 0 (OFF) or 1 (ON) both have the same effect. Flag = 2 (RESET) also clears segment and macro definitions.

FORTRAN PARAMETER

INTEGER*2	IFLAC

MOV2R MOV2R

SYNTAX

ASCII	MOV2R dx, dy
FORTRAN Call	CALL MOV2R (IX, IY)
Binary	[4] [dxdy] (2 bytes)

FUNCTION

The MOV2R command is a two-byte version of the MOVREL command. The MOV2R command should only be called by the host in binary mode.

MOV2R changes the WCS current point (CREG 0) to the point specified by the (dx,dy) offset from the current point. Only two bytes are required for the command. The range of dx and dy is -8 to 7, and dx,dy is specified as nibbles of a single byte.

ASCII PARAMETERS

dx, dy The relative offset for the coordinate; range is -8 to 7.

FORTRAN PARAMETERS

INTEGER*2 IX, IY

The FORTRAN library manages the packing of the two numbers into a single byte.

EXAMPLE

Because nibbles are used to specify dx,dy, some examples may be useful:

MOV2R 17 moves (+1,+1): 17 = (16*1)+1 = 00010001

MOV2R -103 moves (-7,-7): -103 = 10011001

MOV2R -1 moves (-1,-1): -1 = 16*(NOT(1)+1)+(NOT(1)+1) = 11111111 = -1

The rules to generate this number are:

1. Negative numbers are encoded in binary as two's complement notation.

2. Positive binary is converted directly to decimal or hexadecimal.

3. Negative binary is converted to negative decimal by performing a two's complement binary weighting.

4. Negative binary is converted to hexadecimal by straight conversion on binary weighting and preceding both hex numbers with #FF.

MOV3R MOV3R

SYNTAX

ASCII	MOV3R dx, dy
FORTRAN Call	CALL MOV3R (IDX, IDY)
Binary	[3] [dx] [dy] (3 bytes)

FUNCTION

The MOV3R command is a three byte form of the MOVREL command. The MOV3R command changes the current point (CREG 0) by the relative amount specified by dx,dy. This command reduces the number of bytes which must pass between the Model One and the host computer when the displacement of the current point is within the given range.

ASCII PARAMETERS

dx, dy The relative offset for coordinate; range is -128 to 127.

FORTRAN PARAMETERS

INTEGER*2 IDX, IDY

MOVABS MOVABS

SYNTAX

ASCII	MOVABS x, y	
FORTRAN Call	CALL MOVABS (IX, IY)	
Binary	[1] [highx][lowx] [highy][lowy]	(5 bytes)

FUNCTION

The MOVABS command changes the WCS current point (CREG 0) to the point specified by x,y. All subsequent 2-D graphics primitives (lines, circles, arcs, polygons, etc.) are drawn beginning at the location of the current point.

ASCII PARAMETERS

x,y	16-bit integers specifying the absolute x,y coordinate; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2 IX, IY

EXAMPLE

```
! MOVABS 50 70          ; Move the current point to 50,70.
! DRWABS 100 -10        ; Draw a line from current point to 100,-10.
! CIRCLE 15             ; Draw a circle centered at 100,-10 and with
                           radius 15.
! MOVABS 0 0            ; Move the current point to 0,0.
! CIRCLE 20             ; Draw a circle centered at 0,0 and with
                           radius 20.
```

MOVCUR MOVCUR

SYNTAX

ASCII MOVCUR x,y

FORTRAN Call CALL MOVCUR (X, Y)

Binary [200] [X] [Y] (5 bytes)

200 decimal = 310 octal = C8 hex

FUNCTION

The MOVCUR command moves the alphanumeric window cursor to the specified Model One coordinate (x,y), within the boundaries of the window.

For windows 1 through 7, the cursor position will be rounded to the nearest character position; for window 0, the precise Model One coordinate is used.

Note: The MOVCUR command has no effect within the VT100 window (window 8).

ASCII PARAMETERS

x,y Sixteen-bit parameters which specify the Model One coordinate position to which the alphanumeric window cursor is moved.

FORTRAN PARAMETERS

INTEGER*2 X, Y

EXAMPLE

```
! MOVCUR 100,100        ; Move the cursor to the Model One coordinate
                          (100,100).
```

MOVI MOVI

SYNTAX

ASCII	MOVI creg
FORTRAN Call	CALL MOVI (ICREG)
Binary	[5] [creg] (2 bytes)

FUNCTION

The MOVI command changes the WCS current point (CREG 0) to the address specified in coordinate register creg. The MOVI command is most often used to access the current coordinate from the digitizing tablet which is stored in CREG 2.

The MOVI command has the same effect at the command CMOVE 0 creg, which copies a given coordinate register into CREG 0.

ASCII PARAMETER

creg	Coordinate register; range is 0 to 63. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETER

INTEGER*2 ICREG

EXAMPLE

```
! CLOAD 15 100 150      ; Load 100,150 into CREG 15.
! MOVI 15               ; Move to location given in CREG 15.
! DRWABS 140 100        ; Draw line from current point to 140,100.
! MOVI 2                ; Move to the location given in CREG 2 (the
                         current digitizing tablet location).
! CIRCLE 25             ; Draw a circle of radius 25 centered at the
                         current point.
```

MOVREL MOVREL

SYNTAX

ASCII	MOVREL dx, dy
FORTRAN Call	CALL MOVREL (IDX, IDY)
Binary	[2] [highdx][lowdx] [highdy][lowdy] (5 bytes)

FUNCTION

The MOVREL command changes the WCS current point (CREG 0) by a relative amount specified by dx and dy. The new current point is set to the sum of the x-component of the old current point plus dx and the sum of the y-component of the old current point plus dy.

ASCII PARAMETERS

dx, dy	16-bit integer values specifying the relative offset for the coordinate; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2 IDX, IDY

EXAMPLE

```
! MOVABS 100 -130        ; Move the current point to 100,-130.
! CIRCLE 50              ; Draw circle of radius 50 centered at 100,-130.
! MOVREL 20 20           ; Move current point by 20,20 to 120,-110.
! CIRCLE 30              ; Draw circle of radius 30 centered at 120,-110.
```

NULL NULL

SYNTAX

ASCII	NULL
FORTRAN Call	CALL NULL
Binary	[0 or 10 or 13 or 128 or 138 or 141] (1 byte)
	Values in hex are: 00 or 0A or 0D or 80 or 8A or 8D.

FUNCTION

The NULL command is analogous to a NOP (No OPeration) and has no effect. The NULL command has several opcodes, any one of which executes the NULL command. The NULL command can be used to pad a command data buffer so that the Model One can be used on systems capable of transmitting only fixed length blocks.

The opcodes of the NULL command were chosen so that the Model One ignores carriage returns and linefeeds sent between commands for hosts that cannot be made to inhibit sending these characters.

OVRSTK OVRSTK

SYNTAX

ASCII	OVRSTK flag
FORTRAN Call	CALL OVRSTK (FLAG)
Binary	[205] [flag] (2 bytes)
	205 decimal = 313 octal = CD hex

FUNCTION

The OVRSTK command enables or disables overstriking of alphanumeric window text in the currently selected alphanumeric window.

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 7, only the last character in a given position will remain after the window is scrolled.

Note: The OVRSTK command has no effect within the VT100 window (window 8).

ASCII PARAMETER

flag — Eight-bit parameter that specifies whether overstriking of text is enabled:

flag = 1 or ON enables overstriking
flag = 0 or OFF (the default) disables overstriking.

FORTRAN PARAMETER

INTEGER*2 FLAG

EXAMPLE

```
! OVRSTK ON          ; Enable overstriking of alphanumeric window text
                      characters.
```

PEEK PEEK

SYNTAX

ASCII	PEEK addr
FORTRAN Call	CALL PEEK (IADDR, IWORD)
Binary	[189] [highaddr] [lowaddr] (3 bytes)
	189 decimal = 275 octal = BD hex

FUNCTION

The PEEK command sends the contents of the central processor memory location addr to the requesting device. The PEEK command is intended to be used in conjunction with the POKE command as a debugging tool for implementers, and not as a general purpose command for users.

The address must be an even number (word address); if the address is odd, bit 0 will be ignored. The data is displayed in ASCII hexadecimal.

If the PEEK command is issued by the host, the Model One will wait for an ACK (06 hex or 86 hex) character from the host before continuing command interpretation. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

ASCII PARAMETER

addr	16-bit integer specifying the address in central processor memory from which to read data.

FORTRAN PARAMETERS

Input parameter:	INTEGER*2	IADDR
Output parameter:	INTEGER*2	IWORD

IWORD is set equal to the contents of the location specified by IADDR. The FORTRAN library handles transmission of the ACK character.

EXAMPLE

```
! PEEK 0                ; Display contents of location 0.
  FFFF                   (Possible response from Model One.)
! PEEK #0FFE            ; Display contents of location 0FFE hex.
  003F                   (Possible response from Model One.)
```

PICKID PICKID

SYNTAX

ASCII	PICKID pickid
FORTRAN Call	CALL PICKID (PID)
Binary	[217] [pickid3] [pickid2] [pickid1] [pickid0] (5 bytes)
	217 decimal = 331 octal = D9 hex

FUNCTION

The PICKID command is only useful within the definition of a segment.

The PICKID command assigns a pick identification number to all the graphics primitives and other commands immediately following the PICKID command until the next instance of the PICKID command or until the end of the segment.

When a segment is executing, the current pick ID remains in effect until either the next PICKID command or the current segment completes execution. When the current segment completes execution, the pick ID is set to what it was before execution of the segment.

The primitives tagged with a PICKID label can later be deleted with the DELPID command; new primitives can be added to the end of a segment with the SEGAPP command.

ASCII PARAMETER

pickid A 32-bit pick identification number; range is 00000000 to FBFFFFFF (hex). PICKIDs in the range FC000000 to FFFFFFFF are reserved.

FORTRAN PARAMETER

INTEGER*4 PID

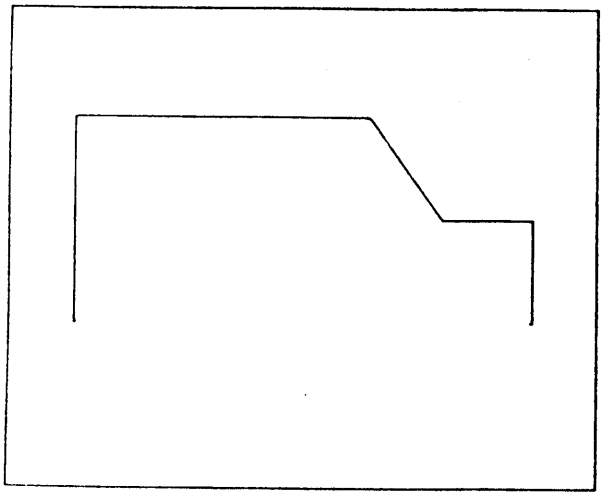
PICKID PICKID

EXAMPLE

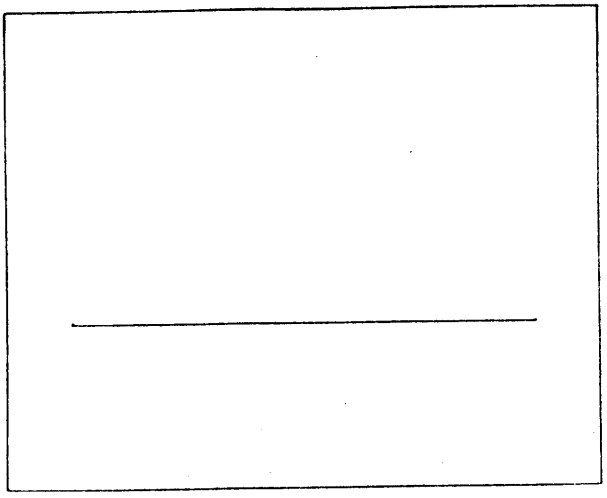
```
! SEGINI 0 256          ; Initialize display list structures with a
                          block size of 256 bytes.
! SEGDEF 1              ; Begin definition of Segment 1, which defines
                          the truck body, chassis, and the position of
                          the 2 wheels.
$ PICKID 10            ; Assign pick ID 10 label to the truck body.
$ PUSH XFORM XF2DCP    ; Push current 2-D transformation and WCS current
                          point onto the stack.
$ MOVABS -500 -160     ; Move current point to lower left corner of truck.
$ DRWREL 0 440         ; Draw 5 vectors to define truck body.
$ DRWREL 640 0
$ DRWREL 160 -220
$ DRWREL 200 0
$ DRWREL 0 -220
$ PICKID 20            ; End pick ID 10 (body); assign pick ID 20 label to
                          the chassis (bottom line) of truck.
$ DRWREL -1000 0       ; Draw chassis of truck.
$ MOVABS -320 -200     ; Move current point to position rear wheel.
$ SEGREF 2             ; Nest Segment 2, the wheel.
$ MOVABS 310 -200      ; Move current point to position front wheel.
$ SEGREF 2             ; Nest Segment 2, the wheel.
$ POP XFORM XF2DCP     ; Pop current 2-D transformation and WCS current
                          point from the stack.
$ SEGEND               ; End pick ID 20 (chassis); end definition of
                          Segment 1.

! SEGDEF 2             ; Begin definition of Segment 2, which defines the
                          shape of the wheel.
$ PICKID 30            ; Assign pick ID 30 label to the wheel.
$ PUSH CREG CURPNT     ; Save current point (center of wheel).
$ MOVREL -100 -100     ; Move current point to left corner of wheel.
$ RECREL 200 200       ; Draw wheel.
$ POP CREG CURPNT      ; Restore current point (center of wheel).
$ SEGEND               ; End pick ID 30 (wheel); end definition of
                          Segment 2, the wheel.
!                      ; Model One returns to normal command mode.
```

PICKID PICKID

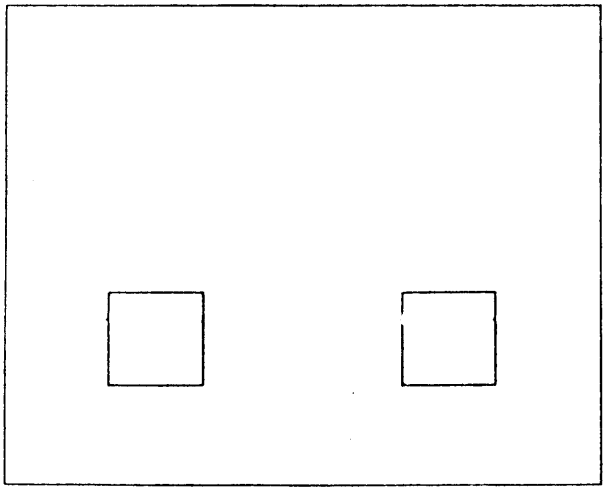
Primitives Tagged "PICKID 10"



Primitives Tagged "PICKID 20"



Primitives Tagged "PICKID 30""



PIXEL8 PIXEL8

SYNTAX

ASCII	PIXEL8 nrows, ncols, val, ...
FORTRAN Call	CALL PIXEL8 (NROWS, NCOLS, IDATA)
Binary	[41] [highnrows][lownrows] [highncols][lowncols] ([val]) ... (5 + nrows * ncols bytes)
	41 decimal = 51 octal = 29 hex

FUNCTION

The PIXEL8 command transmits an image to the Model One pixel by pixel. The array of transmitted data is nrows high and ncols wide. The upper left corner of the array is defined by the current point (CREG 0). Pixels in image memory are filled left to right, top to bottom. Each pixel value is sent as an 8-bit quantity, which is used as an index to the look-up table. The PIXEL8 command is most useful in loading one of the image memory banks with pixel data, as in an 8-bit pseudo color application.

The PIXEL8 command does not change the current point. The PIXEL8 command is not clipped. If the pixel data goes beyond the physical bounds of image memory, it will wrap around. You should avoid this situation, because the result is not always predictable.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is 0 to 32,767.
val	8-bit value specifying the index in the look-up table; range is 0 to 225.

PIXEL8 PIXEL8

FORTRAN PARAMETERS

Input Parameters:	INTEGER*2	NROWS, NCOLS
Output Parameter:	LOGICAL*1	IDATA(1)

IDATA is a byte array which contains the pixel values. The first pixel value is in the first element of IDATA, and subsequent values are stored in successive array elements. IDATA must be dimensioned to at least NROWS * NCOLS elements.

EXAMPLE

To transmit a rectangular window 10 pixels wide by 5 pixels long,

- move the current point to the upper left corner of where you want to display image

- execute the command PIXEL8 5 10, followed by 50 single-byte values.

PIXELS PIXELS

SYNTAX

ASCII	PIXELS nrows, ncols, r, g, b, ...
FORTRAN Call	CALL PIXELS (NROWS, NCOLS, IRED, IGRN, IBLU)
Binary	[40] [highnrows][lownrows] [highncols][lowncols] ([r] [g] [b]) ... (5 + 3 * nrows * ncols bytes)

40 decimal = 50 octal = 28 hex

FUNCTION

The PIXELS command transmits an image to the Model One pixel by pixel. The array of transmitted data is nrows high and ncols wide. The upper left corner of the array is given by the current point (CREG 0). Pixels in image memory are filled left to right, top to bottom. Each pixel value is sent as a full, 24-bit quantity, one byte each of red, green, and blue.

On an 8-bit system, only the red byte is used to determine the location in the look-up table. For this reason, the PIXELS command is very inefficient and you should only use it if you require compatibility with other Model Ones. Otherwise, use the PIXEL8 command.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is 0 to 32,767.
r, g, b	8-bit integers specifying the red, green, and blue color values for each pixel; range is 0 to 255.

FORTRAN PARAMETERS

Input Parameters:	INTEGER*2	NROWS, NCOLS
Output Parameters:	LOGICAL*1	IRED(1), IGRN(1), IBLU(1)

IRED, IGRN, and IBLU are byte arrays which contain the red, green, and blue pixel values. Each array must be dimensioned to at least NROWS * NCOLS elements.

EXAMPLE

To transmit a rectangular window 10 pixels wide by 5 pixels long,

- move the current point to the upper left corner of where you want to display image

- execute the command PIXELS 5 10, followed by 50 three-byte values.