CSUB CSUB

SYNTAX

ASCII	CSUB cdif, creg
FORTRAN Call	CALL CSUB (ICDIF, ICREG)
Binary	[163] [cdif] [creg] (3 bytes)
	163 decimal = 243 octal = A3 hex

FUNCTION

The CSUB command subtracts the contents of the coordinate register specified by creg from the contents of the coordinate register specified by cdif and places the result in the coordinate register specified by cdif.

This command should not be used with CREG 3 (the coordinate origin) as the cdif register.

ASCII PARAMETERS

cdif, creg Coordinate registers; range is from 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 PARAMETERS

INTEGER*2 ICDIF, ICREG

EXAMPLE

```
! CLOAD 20 100 150      ; Load CREG 20 with 100,150.
! CLOAD 21 25 30        ; Load CREG 21 with 25,30.
! CSUB 20 21            ; Subtract the contents of CREG 21 from
                           CREG 20 and place result in CREG 20.
! READCR 20             ; Read contents of CREG 20.
  00075 00120              (Response from Model One.)
```

CURSOR CURSOR

SYNTAX

ASCII	CURSOR cursor, flag
FORTRAN Call	CALL CURSOR (CURSOR, FLAG)
Binary	[74] [cursor] [flag] (3 bytes)
	74 decimal = 112 octal = 4A hex

FUNCTION

The CURSOR command turns the full-screen hardware cursor on and off. Cursor must be zero. If flag = 1 or ON, the cursor is displayed. If flag = 0 or OFF, the cursor is not displayed.

The cursor position is controlled with CREG 17. The cursor color is always white and does not change color depending on the image data.

The full-screen hardware cursor should be used in preference to the smaller firmware crosshairs controlled with the XHAIR command; the crosshairs reduce system performance, while the cursor does not.

ASCII PARAMETERS

cursor	Cursor; must be zero.
flag	Flag = 1 or ON enables the cursor; flag = 0 or OFF disables the cursor.

FORTRAN PARAMETERS

INTEGER*2 CURSOR, FLAG

EXAMPLE

```
! MACDEF 10              ; Start definition of Macro 10.
$ CMOVE 17 0             ; Copy the location of the current point
                           into CREG 17, which controls the cursor
                           position.
$ MACEND                 ; End macro definition.
! BUTTBL 0 10            ; Execute Macro 10 every 1/30th second.
! CURSOR 0 ON            ; Enable display of full screen cursor.
! MOVREL 100,100         ; Change the current point by 100,100.
                           (Note that cursor also moves.)
```

DEBUG DEBUG

SYNTAX

ASCII	DEBUG flag
FORTRAN Call	CALL DEBUG (FLAG)
Binary	[168] [flag] (2 bytes)
	168 decimal = 250 octal = A8 hex

FUNCTION

The DEBUG command is used to enter and exit the command stream translator. When flag = 1 or ON, the central processor displays, in mnemonic form, the commands that are being executed by the command interpreter. If flag = 0, DEBUG is disabled.

ASCII PARAMETER

flag	Flag = 1 or ON enables Command Stream Translator; flag = 0 or OFF disables Command Stream Translator.

FORTRAN PARAMETER

INTEGER*2	FLAG

DEFWIN DEFWIN

SYNTAX

ASCII
```
DEFWIN window, xl, yl, x2, y2, xsize, ysize,
      bitm, bankm
```

FORTRAN Call
```
CALL DEFWIN (WINDOW,X1,Y1,X2,Y2,XSIZE,YSIZE,
             BITM,BANKM)
```

Binary
```
[192] [window] [highx1] [lowx1] [highy1] [lowy1]
      [highx2] [lowx2] [highy2] [lowy2] [xsize]
      [ysize] [bitm] [bankm]      (14 bytes)
```

192 decimal = 300 octal = C0 hex

FUNCTION

These pages describe the use of the DEFWIN command for the alphanumeric windows (window 0 through 7). For a description of how the DEFWIN command is used for the VT100 window (window 8), refer to the VT100 Emulation Guide.

The DEFWIN command is used to define each of the alphanumeric windows (windows 0 through 7), specifying the

- window number
- corners of the window
- size of the text within the window, and
- the write mask.

Window 0 specifies the hardware-scrolled window, which uses the entire screen of the monitor. This window uses the screen origin register for scrolling. Thus, changes to CREG 4 will affect window 0. The seven other windows (windows 1 through 7) 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 7) will always be on top of the other windows.

Three value registers (VREGs) 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 7, which uses value registers 37, 38, and 39.

DEFWIN DEFWIN

FUNCTION, continued

For windows 1 through 7, 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.

ASCII PARAMETERS

window	The window number; the range is 0 to 7.
x1,y1	Sixteen-bit parameters which specify the coordinates of the first corner of the window (ignored for window 0).
x2,y2	Sixteen-bit parameters which specify the coordinates of the the diagonally opposite corner of the window (ignored for window 0).
xsize,ysize	Eight-bit parameters which specify the x and y scaling of text within the window. Normally set to 1,1, which is equivalent to specifying a text size of 16 with the TEXTC command. The xsize and ysize may be different, allowing tall skinny or short fat text.
bitm,bankm	Eight-bit parameters specifying the write mask, as detailed in the WRMASK command.

FORTRAN PARAMETERS

```
CALL DEFWIN (WINDOW,X1,Y1,X2,Y2,XSIZE,YSIZE,BITM,BANKM)

INTEGER*2      WINDOW, XSIZE, YSIZE, BITM, BANKM, X1, Y1, X2, Y2
```

EXAMPLE

```
! DEFWIN 1 20,20 50,50 1,1 0,0
                              ; Define 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.
```

DELAY DELAY

SYNTAX

ASCII	DELAY amount
FORTRAN Call	CALL DELAY (IAMT)
Binary	[182] [amount] (2 bytes)
	182 decimal = 266 octal = B6 hex

FUNCTION

The DELAY command inserts a delay between characters when sending readback data to the host over the HOSTIO interface. The DELAY command does not affect the DMA interface. The DELAY command is necessary because some host computers cannot accept data as fast as the Model One can send it, even at a given baud. This is particularly true at the higher baud rates.

Amount specifies the amount of time to insert between characters. The recommended amount for a VAX 11/7xx is 10 units. One unit of delay is equal to 750 microseconds.

delay = 750 * amount microseconds

ASCII PARAMETER

amount Amount of delay; range is 0 to 255. Default is 0.

FORTRAN PARAMETER

INTEGER*2 IAMT

DELPID DELPID

SYNTAX

ASCII	DELPID
FORTRAN Call	CALL DELPID
Binary	[236] (1 byte)
	236 decimal = 354 octal = EC hex

FUNCTION

The DELPID command deletes all commands with the current pick ID (stored in PIDREG) and the current segment ID (stored in SEGREG). The commands are deleted from the current pick ID to whichever comes first:

- the next pick ID, or
- the end of the segment.

The pick identification number, however, still exists within its segment. You can reopen the segment and append commands to change the segment's contents with the SEGAPP command or the INSPID command.

If you have not just picked this pick ID, thus making it current, first set both registers, PIDREG and SEGREG, with the SETGL command or with the RDPICK command.

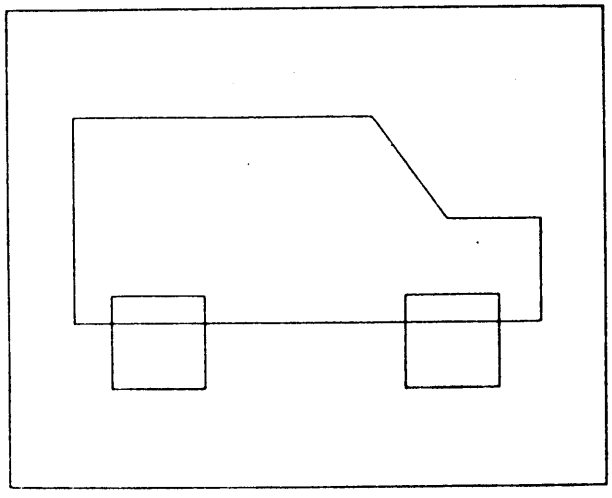
EXAMPLE

This example illustrates the use of the DELPID command. Segment 1 is already defined and contains two pick IDs. Pick ID 10 is the truck body and pick ID 20 is the chassis (bottom line) of the truck. Pick ID 20 contains two references to Segment 2, which is the definition of the wheel. Refer to the SEGDEF command for a complete definition of the truck.

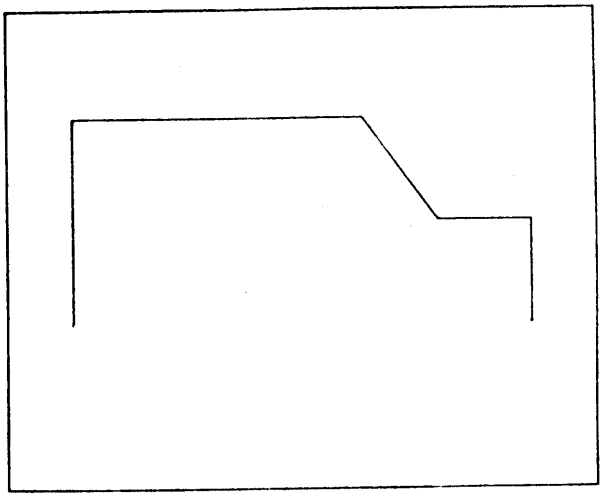
Note that Segment 2 is not deleted. However, pick ID 20 (which contains the references to Segment 2) is deleted. Therefore the wheels (Segment 2) are not drawn in this example after the DELPID command is executed.

```
! SEGREF 1                ; Execute (draw) Segment 1, truck with two wheels.
! CLOAD 19 0 -160         ; Set center of pick aperture.
! EXMODE PICK             ; Set execution mode to pick.
! SEGREF 1                ; Execute (pick) segment 1.
! RDREG 0                 ; Read back contents of segment ID and pick ID
                           registers.
 0000000001 0000000020   ; The Model One returns:  segment ID is 1, pick ID
                           is 20.
! EXMODE NORMAL           ; Restore normal draw execution mode.
! DELPID                  ; Delete all commands with pick ID 20 in Segment 1.
! VAL8 0                  ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1                ; Execute (draw) Segment 1, now truck body without
                           chassis or wheels.
```

BEFORE



AFTER



DELWIN DELWIN

SYNTAX

ASCII	DELWIN window
FORTRAN Call	CALL DELWIN (WINDOW)
Binary	[195] [window] (2 bytes)
	195 decimal = 303 octal = C3 hex

FUNCTION

The DELWIN command deletes a previously defined alphanumeric window or VT100 window, freeing the buffer space for use. The currently active alpha window cannot be deleted.

ASCII PARAMETER

window	The alphanumeric window number (0 through 7) or the VT100 window number (8).

FORTRAN PARAMETER

INTEGER*2	WINDOW

EXAMPLE

```
! SELWIN 2              ; Select window 2.
! DELWIN 3              ; Delete window 3.
! DELWIN 2              ; Delete window 2.

Error 075               ; Note: Window 2 cannot be deleted; it is selected.

DELWIN: Illegal window number
```

DFTCFG DFTCFG

SYNTAX

ASCII DFTCFG

FUNCTION

The DFTCFG command restores all ports on the Model One to the default configuration and sets the host serial communications mode to binary. In addition, it restores special characters, alpha windows, genlock, video format, and powerup status. The default configuration for the Model One is:

PORT	RTS	CTS	STOP BITS		XIN	XOUT	CTRL	PARITY	BAUD	QUEUE
KEYBSIO	OFF	OFF	2	7	ON	OFF	ON	NONE	300	0006
TABLETSIO	OFF	OFF	2	8	OFF	OFF	OFF	NONE	9600	0006
ALPHASIO	OFF	OFF	2	8	ON	OFF	ON	NONE	9600	0006
HOSTIO	OFF	OFF	1	8	OFF	ON	OFF	NONE	9600	000B

The system configuration is modified through the use of the SYSCFG and SAVCFG commands. The SYSCFG command configures the Model One's ports; the SAVCFG command stores those configurations (which are then loaded whenever a COLDstart is performed).

The default configuration is not modified by SYSCFG or SAVCFG; the DFTCFG command should be used only when it is necessary to restore all the Model One's ports to a known state.

To display the current configuration, you can use the DISCFG command.

The DFTCFG command can be executed only from the local alphanumeric terminal, and cannot be included in a macro.

The DFTCFG command executes a WARMstart as part of the command, leaving the Model One in ALPHA mode.

DFTCFG DFTCFG

EXAMPLE

```
! SYSCFG SERIAL HOSTIO XIN ON XOUT ON PARITY N

                        ; Configures port HOSTIO as indicated.
                          (See SYSCFG for details.)

! SAVCFG                 ; Save the configuration of port HOSTIO.
! DFTCFG                 ; Restore the default configuration for all
                          ports (not just port HOSTIO).
```

Note: When you execute the SYSCFG, SAVCFG, and DFTCFG commands, the Model One allows you to confirm the configuration by prompting "Are you sure??". You must respond with "y" in order for the command to be processed.

DIRCUR DIRCUR

SYNTAX

ASCII DIRCUR x,y

FORTRAN Call CALL DIRCUR (X,Y)

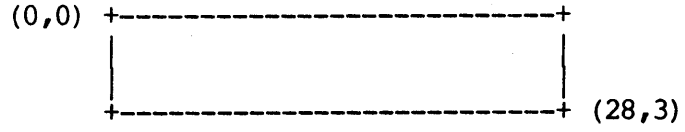
Binary [196] [x] [y] (3 bytes)

196 decimal = 304 octal = C4 hex

FUNCTION

The DIRCUR command moves the cursor to character position (x,y) within the alphanumeric window. This command has no effect for the VT100 window (window 8).

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 is 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 (assuming a full-screen window) is (211,101).

ASCII PARAMETERS

x,y Eight-bit parameters which specify the character position within the window (in terms of the x and y coordinates).

DIRCUR DIRCUR

FORTRAN PARAMETERS

```
CALL DIRCUR (X, Y)

INTEGER*2      X, Y
```

EXAMPLE

```
! DIRCUR 3,4          ; Move the cursor to character position (3,4), which
                       is the fourth row, fifth column.
```

DISCFG DISCFC

SYNTAX

ASCII DISCFG

FUNCTION

The DISCFG command displays the current configurations of the Model One's serial ports, as set with the SYSCFG command. The DISCFG command also displays the following information:

- the host mode (binary or ASCII hex)
- the ROM sequence number
- the graphics input device
- special characters, and
- the states of the GENLOCK, VIDFORM, RGBTRU, and ALPHEM commands.

Note: DISCFG command can be executed from the local terminal only, and may not be included in a macro.

DRW2R DRW2R

SYNTAX

ASCII	DRW2R dx, dy	
FORTRAN Call	CALL DRW2R (DX, DY)	
Binary	[132] [dxdy]	(2 bytes)
	132 decimal = 204 octal = 84 hex	

FUNCTION

The DRW2R command is a two-byte version of the DRWREL command. The DRW2R command should only be called by the host in binary mode.

DRW2R draws a vector from the WCS current point (CREG 0) to the point specified by the (dx,dy) offset from the current point. The current point is changed to the end point after completion of the command. Only two bytes are required for the command. The range of dx and dy is -8 to 7, and dx,dy are 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 DX, DY

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:

DRW2R 17 draws (+1,+1): 17 = (16*1)+1 = 00010001

DRW2R -103 draws (-7,-7): -103 = 10011001

DRW2R -1 draws (-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.

DRW3R DRW3R

SYNTAX

ASCII	DRW3R dx, dy
FORTRAN Call	CALL DRW3R (IDX, IDY)
Binary	[131] [dx] [dy] (3 bytes)
	131 decimal = 203 octal = 83 hex

FUNCTION

The DRW3R command is a three byte form of the DRWREL command. The DRW3R command draws a vector from the WCS current point (CREG 0) to the point relative to the current point offset by dx and dy. The current point is changed to this new point. The range of dx and dy is -128 to 127. This command reduces the number of bytes which must pass between the Model One/380 and the host computer for vectors whose maximum displacement is within the given range.

ASCII PARAMETERS

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

FORTRAN PARAMETERS

INTEGER*2	IDX, IDY

DRWABS DRWABS

SYNTAX

ASCII	DRWABS x, y
FORTRAN Call	CALL DRWABS (IX, IY)
Binary	[129] [highx] [lowx] [highy] [lowy] (5 bytes)
	129 decimal = 201 octal = 81 hex

FUNCTION

The DRWABS command draws a vector from the WCS current point (CREG 0) to the point specified by x,y and changes the current point (CREG 0) to x,y. The pixels along the line are drawn in the current pixel value (VREG 0).

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 50,50        ; Move current point to 50,50.
! DRWABS 60,50        ; Draw line to 60,50 (horizontal line 11 pixels
                       long, with both end point included).
! MOVABS 60,60        ; Move current point to 60,60.
! DRWABS 60,70        ; Draw line to 60,70 (vertical line 11 pixels
                       long).
! DRWABS 70 70        ; Draw line to 70,70 (diagonal line connected
                       to previous line).
! DRWABS 80 100       ; Draw line to 80,100.
```

DRWI DRWI

SYNTAX

ASCII	DRWI creg
FORTRAN Call	CALL DRWI (ICREG)
Binary	[133] [creg] (2 bytes)
	133 decimal = 205 octal = 85 hex

FUNCTION

The DRWI command draws a vector from the WCS current point (CREG 0) to the point given by coordinate register creg and changes the current point (CREG 0) to the new point.

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

```
! MOVABS -100 -50      ; Move current point to -100,-50.
! DRWI 4               ; Draw vector from -100,-50 to location given
                        in CREG 4.
! MOVABS -30 -60       ; Move current point to -30,-60.
! CLOAD 33 100,150     ; Load CREG 33 with 100,150.
! DRWI 33              ; Draw vector from current point (-30,-60)
                        to location given in CREG 33 (100,150).
```

DRWREL DRWREL

SYNTAX

ASCII	DRWREL dx, dy
FORTRAN Call	CALL DRWREL (IDX, IDY)
Binary	[130] [highdx] [lowdx] [highdy] [lowdy] (5 bytes)
	130 decimal = 202 octal = 82 hex

FUNCTION

The DRWREL command draws a vector from the WCS current point (CREG 0) to the point relative to the current point offset by dx and dy. The 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 PARAMETER

INTEGER*2 IDX, IDY

EXAMPLE

```
! MOVABS 50 30          ; Move to location 50,30.
! DRWREL 10 20          ; Draw line from 50,30 to 60,50.
! DRWREL 10 0           ; Draw line from 60,50 to 70,50.
! DRWREL 0 10           ; Draw line from 70,50 to 70,60.
```

EXMODE EXMODE

SYNTAX

ASCII	EXMODE mode
FORTRAN Call	CALL EXMODE (MODE)
Binary	[123] [mode] (2 bytes)
	123 decimal = 173 octal = 7B hex

FUNCTION

The EXMODE command sets the display controller to the specified execution mode. The four execution modes are: NORMAL DRAW, PICK, HILITE, and UNHILITE.

If you are not already in pick mode, and you execute EXMODE PICK, the command also clears the pick buffer and calculates the pick aperture as set with the SETGL PICKAP command.

ASCII PARAMETER

mode Execution mode (8 bits).

0 = NORMAL DRAW	normal draw mode
1 = PICK	pick mode
2 = HILITE	highlight mode
3 = UNHILITE	unhighlight mode

FORTRAN PARAMETER

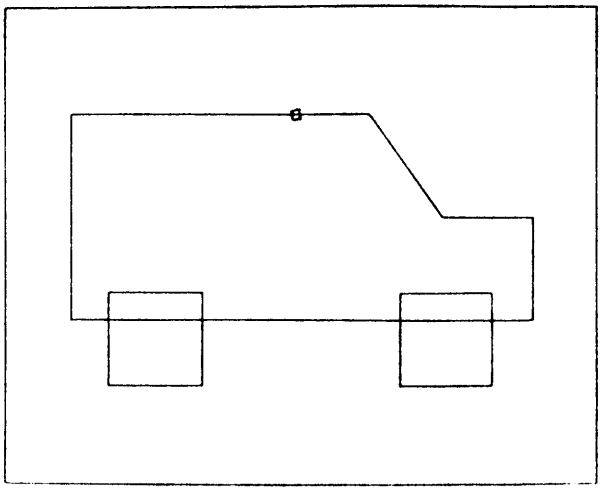
INTEGER*2 MODE

EXAMPLE 1

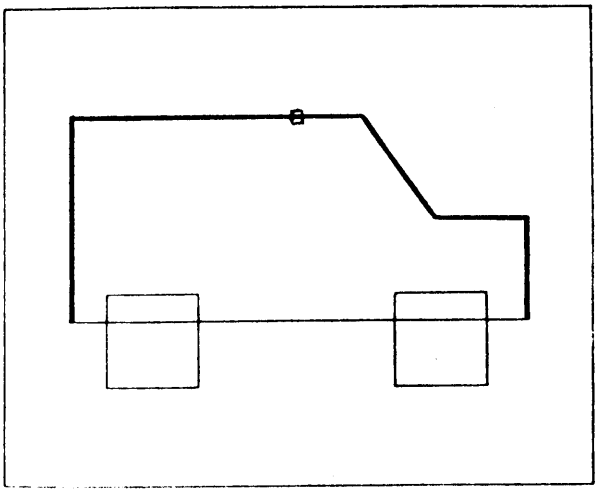
The following example illustrates the four different execution modes that you can set with the EXMODE command: normal draw, pick, highlight, and unhighlight. Segment 1 is already is already defined and contains two pick IDs. Pick ID 10 is the truck body and pick ID 20 is the chassis (bottom line) of the truck. Pick ID 20 contains two references to Segment 2, the wheel. Refer to the SEGDEF command for a complete definition of the truck.

```
! EXMODE NORMAL          ; Set execution mode to normal draw.
! SEGREF 1               ; Execute (draw) Segment 1.  (See the figure
                           labelled NORMAL DRAW.)
! SETGL PICKAP 10 10     ; Set the pick aperture size to 20 x 20.
! CLOAD 19 0,280         ; Specify the center of pick aperture. (See the
                           figures below.)
! VLOAD 45 48,0,0        ; Load VREG 45 with highlight color, red.
! EXMODE PICK            ; Set execution mode to pick.
! SEGREF 1               ; Execute (pick) Segment 1.
! RDREG 0                ; Read back contents of segment ID and pick ID
                           registers.
 0000000001 0000000010
                         ; The Model One returns: segment ID is 1, pick ID
                           is 10.
! EXMODE HILITE          ; Set execution mode to highlight.
! SEGREF 1               ; Execute Segment 1:  draw in red the geometry
                           labelled pick ID 10 in segment 1.  (See the
                           figure labelled HILITE.)
! EXMODE UNHILITE        ; Set execution mode to unhighlight.
! SEGREF 1               ; Execute Segment 1:  draw in normal color the
                           geometry that has been picked.
! EXMODE NORMAL          ; Resume normal draw mode.
```

NORMAL DRAW



HILITE



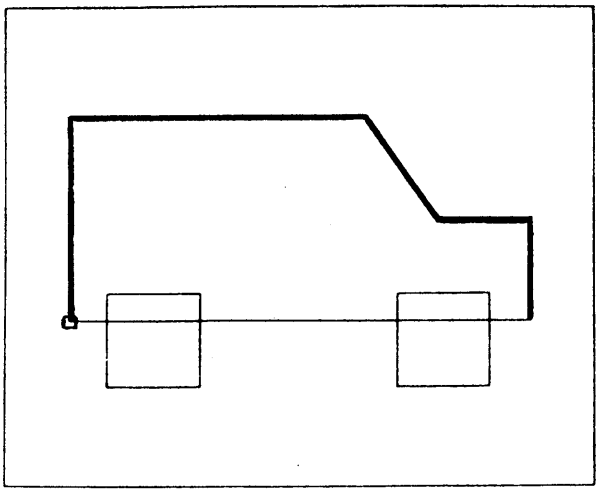
EXAMPLE 2

The following example illustrates a more complex case of picking and highlighting. In this case, the pick aperture is set at the intersection of the truck body (pick ID 10) and the chassis (pick ID 20). First we highlight the truck body (the first pick hit). Then we use the RDPICK command to update the pick ID and segment ID registers with the next pick hit (the chassis), and highlight the chassis in a different color.

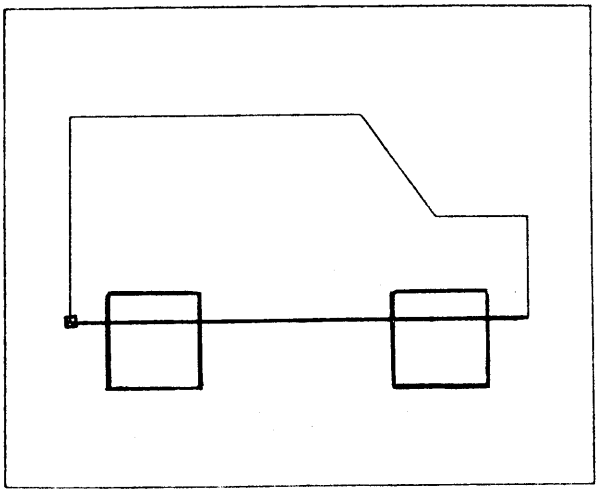
```
! CLOAD 19 -500,-160      ; Set center of pick aperture. (See the figure on
                           the following page.)
! VLOAD 45 48,0,0         ; Load VREC 45 with highlight color, red.
! EXMODE PICK             ; Enter pick mode.
! SEGREF 1                ; Execute (pick) Segment 1.
! RDPICK PICKS 0 0        ; Read back number of pick hits encountered and
                           number of pick hits recorded; registers are
                           not updated.
  00002 00002            ; The Model One returns: two pick hits encountered
                           and recorded.
! RDREG 0                 ; Read back contents of segment ID and pick ID
                           registers.
 0000000001 0000000010   ; The Model One returns: segment ID is 1, pick
                           ID is 10. The registers contain the first pick
                           hit.
! EXMODE HILITE           ; Enter highlight mode.
! SEGREF 1                ; Execute (highlight) Segment 1. The truck body
                           (pick ID 10) is drawn in red. (See the figure
                           labelled HILITE IN RED.)
! VLOAD 45 3,0,0          ; Load VREG 45 with new highlight color, blue.
! RDPICK SEGPID 0 0       ; Update both registers with the next unread
                           entry in the pick buffer.
! RDREG 0                 ; Read back the contents of the segment ID and pick
                           ID registers.
 0000000001 0000000010   ; The next unread location is the first hit, so the
                           registers are not changed.
! RDPICK SEGPID 0 0       ; Update both registers with the next unread
                           entry in the pick buffer.
! RDREG 0                 ; Read back the contents of the segment ID and
                           pick ID registers.
 0000000001 0000000020
                           ; The Model One returns: segment ID is 1, pick
                           ID is 20.
! SEGREF 1                ; Execute (highlight) Segment 1. The chassis and
                           the 2 wheels are highlighted in blue. The wheels
                           (Segment 2) are also highlighted because pick
                           ID 20 in Segment 1 is in effect when Segment 2
                           is referenced. (See the figure labelled HILITE
                           IN BLUE.)
```

EXAMPLE 2, continued

HILITE IN RED



HILITE IN BLUE



FILMSK FILMSK

SYNTAX

ASCII	FILMSK rmsk, gmsk, bmsk
FORTRAN Call	CALL FILMSK (IRMSK, IGMSK, IBMSK)
Binary	[159] [rmsk] [gmsk] [bmsk] (4 bytes)
	159 decimal = 237 octal = 9F hex

FUNCTION

The FILMSK command sets the fill mask (VREG 3) with the value specified by rmsk, gmsk, bmsk. The fill mask is ANDed with boundary values before the boundary check comparison is made in area fill commands (AREA1 and AREA2). The FILMSK command is equivalent to VLOAD 3 rmsk, gmsk, bmsk, as VREG 3 holds the fill mask.

On an 8-bit system, the second and third bytes are ignored when using the fill mask.

ASCII PARAMETERS

rmsk	The red mask; range is 0 to 255.
gmsk	The green mask; range is 0 to 255. (Ignored on 8-bit system.)
bmsk	The blue mask; range is 0 to 255. (Ignored on 8-bit system.)

FORTRAN PARAMETERS

INTEGER*2	IRMSK, IGMSK, IBMSK

EXAMPLE

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

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

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

FIRSTP FIRSTP

SYNTAX

ASCII	FIRSTP flag
FORTRAN Call	CALL FIRSTP (IFLAG)
Binary	[47] [flag] (2 bytes)
	47 decimal = 057 octal = 2F hex

FUNCTION

The FIRSTP command inhibits the writing of the first pixel of vectors after a previous DRAW command when flag = 1 or ON. This prevents shared endpoints of concatenated lines from being written twice into image memory. If flag = 0 or OFF, all pixels are written.

The FIRSTP command is especially useful if the pixel function is not set to INSert.

Note: On concatenated lines that are close to 180 degrees apart, there may be more than one shared point. The FIRSTP command cannot help in this situation.

ASCII PARAMETER

flag	Flag = 1 or ON inhibits writing of first pixel; flag = 0 or OFF allows writing of all pixels.

FORTRAN PARAMETER

INTEGER*2	IFLAG

FLOOD FLOOD

SYNTAX

ASCII	FLOOD	
FORTRAN Call	CALL FLOOD	
Binary	[7]	(1 byte)

FUNCTION

The FLOOD command changes all displayed pixels to the current pixel value (VREG 0) in a single frame time. Pixels that are not being displayed when the FLOOD command is issued are not changed.

This command is not affected by the pixel functions.

EXAMPLE

```
! LUT8 1 0,0,0          ; Set the color out for LUT index 1 to black.
! LUT8 2 0,0,255        ; Set the color out for LUT index 2 to blue.
! VAL8 1                ; Set current pixel value to 1 (black).
! FLOOD                 ; Flood displayed image; screen turns black.
! VAL8 2                ; Change current pixel value to 2 (blue).
! FLOOD                 ; Flood displayed image; screen turns blue.
```

FLUSH FLUSH

SYNTAX

ASCII	FLUSH
FORTRAN Call	CALL FLUSH
Binary	[21] (1 byte)
	21 decimal = 025 octal = 15 hex

FUNCTION

The FLUSH command empties the function button event queue. The event queue keeps a record of all function buttons which have been pressed or released at the local XY digitizer device. The queue is eight event pairs deep. An event pair consists of a button press and release. Each time the host issues a READBU command, an entry is taken out of the event queue and sent to the host.

If the host is not interested in knowing which buttons have been pressed or released, the FLUSH command should be included in macros which use function buttons. Overflow of the event queue results in subsequent function buttons being ignored.

EXAMPLE

```
! MACDEF 10          ; Start definition of Macro 10.
$ CIRCLE 50          ; Draw a circle.
$ FLUSH              ; Empty event queue.
$ MACEND             ; End macro definition.
! BUTTBL 13 10       ; Execute Macro 10 in response to Button 13.
```

FOLD FOLD

SYNTAX

ASCII	FOLD flag, quad
FORTRAN Call	CALL FOLD (FLAG, QUAD)
Binary	[88] [flag] [quad] (3 bytes)
	88 decimal = 130 octal = 58 hex

FUNCTION

The FOLD command configures depth buffer memory. The command has a different function for different hardware configurations, as described below.

Using FOLD with 24-Bit and 40-Bit Systems

For 24-bit and 40-bit systems, you should set folding off by setting the flag parameter to 0 or OFF. When FOLD is set to OFF, the quad parameter indicates the configuration type. For a 24-bit hardware configuration, set quad to 1 for 8-bit pseudo color. For a 40-bit hardware configuration, set quad to 0 for 24-bit true color.

Using FOLD with an 8-Bit System

The FOLD command allows you to reconfigure image memory to provide a 16-bit depth buffer on an 8-bit Model One. The FOLD command folds the 8-bit system's image memory to provide

- one 640 x 512 x 16 depth buffer, and
- two 640 x 512 x 8 image buffers.

The flag parameter indicates whether folding is set to off (0) or on (1, 2, or 3). For an 8-bit system, you must set folding on if you want to use the depth buffer.

If folding is on, then the quad parameter indicates which quadrant to write into (upper left or upper right).

FUNCTION, continued

With folding on, you have three choices affecting how the image is displayed.

With flag = 1 or ONEBUF, you can write into a specified quadrant (quadrant 0 or 1) and display the image in the same quadrant.

With flag = 2 or DUBBUF, you can write into a specified quadrant and display in the other quadrent. This option allows double-buffering.

With flag = 1 or 2, the clipping window is set to the boundary of the quadrant to be written into. The scale factor (zoom) is set to 2, so that the quadrant specified for display uses the whole screen. The coordinate origin needs to be set to 0,0 in the center of the display quadrant (see following page).

With flag = 3 or ALL, you can write into a quadrant of your own choosing and display to the full screen. This option allows you to see the whole image, including depth buffer values, as you develop it. You can use this option for images that cross quadrant boundaries, as long as the displayed image doesn't overlap with the depth buffer image in the the other quadrant. It is your responsibility to select coordinates that fall within the quadrant you are writing into; there is no clipping window set, and the screen origin and zoom are unaffected.

Default Condition

The default condition for the FOLD command is FOLD OFF 1.

Centering the Coordinate Origin

To ensure that you get the expected results, you must first center the coordinate origin before you set FOLD to on with options 1 or 2.

Step	Action
1	Set the coordinate origin to 0,0 by

- returning the offset from the current coordinate origin using the READCR command:

```
! READCR CORORG
```

- specifying the negative of the offset using the CORORG command. Example: If READCR CORORG returns 00110 00200, then you would use CORORG as follows:

```
! CORORG -110 -200
```

2 Set the coordinate origin to the center of the quadrent:

- for Quadrant 0 (upper left) use

```
! CORORG -320 255
```

- for Quadrant 1 (upper right) use

```
! CORORG 320 255
```

3 Execute the FOLD command.

FOLD FOLD

ASCII PARAMETERS

flag 0 = OFF Folding off; use full screen.
Note: You must have a 24-bit or 40-bit system to use this option and still have a depth buffer.

1 = ONEBUF Folding on; write into specified quadrant and display same quadrant.

2 = DUBBUF Folding on; write into specified quadrant and display other quadrant.

3 = ALL Folding on; write into specified quadrant and display full screen.

quad (If flag = 1, 2, or 3):
0 = Quadrant 0, upper left.
1 = Quadrant 1, upper right.

(If flag = 0):
0 = TRUE Color mode 0, true-color hardware configuration (40-bit).
1 = PSEUDO Color mode 1, pseudo-color hardware configuration (24-bit).

FORTRAN PARAMETERS

```
CALL FOLD (FLAG, QUAD)

INTEGER*2     FLAG, QUAD
```

EXAMPLES

```
! FOLD OFF 0          ; Configure for 40-bit 1024 x 1280 system.
                        (24-bit RGB, 16-bit z buffer)

! FOLD OFF 1          ; Configure for 24-bit 1024 x 1280 system. (Default)
                        (8-bit pseudo color, 16-bit z buffer)

! FOLD ONEBUF 0       ; Configure for 8-bit 640 x 512 folded system.
                        Write into Quadrant 0 and display same quadrant.

! FOLD ONEBUF 1       ; Configure for 8-bit 640 x 512 folded system.
                        Write into Quadrant 1 and display same quadrant.
```

EXAMPLES, continued

```
! FOLD DUBBUF 0          ; Configure for 8-bit 640 x 512 folded system.
                           Write into Quadrant 0 and display Quadrant 1.

! FOLD DUBBUF 1          ; Configure for 8-bit 640 x 512 folded system.
                           Write into Quadrant 1 and display Quadrant 0.

! FOLD ALL 0             ; Configure for 8-bit 640 x 512 folded system.
                           Write into Quadrant 0 and display full screen.

! FOLD ALL 1             ; Configure for 8-bit 640 x 512 folded system.
                           Write into Quadrant 1 and display full screen.
```

GENLOCK GENLOCK

SYNTAX

ASCII	GENLOCK flag
FORTRAN Call	CALL GENLCK (IFLAG)
Binary	[76] [flag] (2 bytes)
	76 decimal = 110 octal = 4C hex

FUNCTION

The Genlock command enables or disables the Genlock option. The Genlock option allows the user to use the Model One synchronously with other video equipment.

Important: Use the GENLOCK command only if your Model One is equipped with the Genlock option.

Note: You can use the SAVCFG command to save the GENLOCK setting. The Model One powers up with the setting most recently saved.

ASCII PARAMETER

flag Flag = 1 (or ON) enables the Genlock option (if the Model One has the Genlock option.

Flag = 0 (or OFF) turns off the Genlock option.

FORTRAN PARAMETER

INTEGER*2 IFLAG

GETCUR GETCUR

SYNTAX

ASCII	GETCUR
FORTRAN Call	CALL GETCUR (IX, IY)
Binary	[201] (1 byte)
	201 decimal = 311 octal = C9 hex

FUNCTION

The GETCUR command returns the Model One coordinate position of the alpha window cursor within the currently selected alphanumeric window or VT100 window. Note that the Model One coordinate position is rounded to the nearest character position for windows 1 through 8.

Note: The GETPOS command returns the character position of the alpha window cursor within the currently selected alphanumeric or VT100 window.

ASCII PARAMETERS

The GETCUR command has no ASCII parameters.

FORTRAN PARAMETERS

INTEGER*2 IX, IY

IX The Model One x coordinate of the alpha window cursor.
IY The Model One y coordinate of the alpha window cursor.

EXAMPLE

```
! MOVCUR 100,100        ; Move cursor to coordinate position (100,100).
! GETCUR                ; Get cursor coordinate position.

00100 00100             ; Current coordinate position.
```

GETPOS GETPOS

SYNTAX

ASCII	GETPOS
FORTRAN Call	CALL GETPOS (ICOL, IROW)
Binary	[197] (1 byte)
	197 decimal = 305 octal = C5 hex

FUNCTION

The GETPOS command returns the character position of the alpha window cursor within the currently selected alphanumeric window or VT100 window.

The column and row position of the cursor is returned.

Note: The GETCUR command returns the coordinate position of the alpha window cursor within the currently selected alphanumeric or VT100 window.

ASCII PARAMETERS

The GETPOS command takes no ASCII parameters.

FORTRAN PARAMETERS

INTEGER*2	ICOL, IROW
ICOL	Upon exit, column positon within the alpha window of the alpha window cursor.
IROW	Upon exit, row positon within the alpha window of the alpha window cursor.

EXAMPLE

```
! DIRCUR 3,4          ; Move cursor to character position (3,4), which is
                        the fourth row and the fifth column.
! GETPOS              ; Get cursor character position.

003 004              ; Current character position.
```

GETWIN GETWIN

SYNTAX

ASCII	GETWIN
FORTRAN Call	CALL GETWIN (NUM)
Binary	[206] (1 byte)
	206 decimal = 316 octal = CE hex

FUNCTION

The GETWIN command returns the number of the currently selected alphanumeric window or VT100 window. The value (-1) is returned if no window is active.

ASCII PARAMETERS

The GETWIN command takes no ASCII parameters.

FORTRAN PARAMETER

INTEGER*2 NUM

NUM	Upon exit, the number of the active window (-1 if no window is active).

EXAMPLE

```
! GETWIN              ; Return the number of the selected alpha window.

-001                  ; No alpha window is active.

! SELWIN 7            ; Select alpha window 7.
! GETWIN              ; Return the number of the selected alpha window.

007                   ; Alpha window 7 is now active.
```

HELP HELP

SYNTAX

ASCII HELP
or HELP command (mnemonic or opcode)

Binary [62] (1 byte)

62 decimal = 76 octal = 3E hex

FUNCTION

The HELP command is normally used only in local communications to the alphanumeric terminal or when using a local keyboard. The information is sent to the alpha terminal or screen, regardless of where the command was initiated.

The HELP command has three forms.

HELP lists all commands that are available, in opcode order.

HELP command lists the command mnemonic and opcode and the data transmission format of the parameters for the command.

EXAMPLE

```
! HELP MOVABS            ; Request information on the MOVABS command.

      MOVABS: OPCODE = 001
1.    <16-BIT SIGNED NUMBER>
2.    <16-BIT SIGNED NUMBER>

! HELP 3                 ; Request information on command with opcode 3.

      MOV3R: OPCODE = 003
1.    <8-BIT SIGNED NUMBER>
2.    <8-BIT SIGNED NUMBER>
```

HOME HOME

SYNTAX

ASCII	HOME
FORTRAN Call	CALL HOME
Binary	[207] (1 byte)
	207 decimal = 317 octal = CF hex

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.

The HOME command has no effect within the VT100 window (window 8).

EXAMPLE

```
! DIRCUR 1,1          ; Move cursor to character position (1,1), which is
                        the second row and the second column.
! HOME                ; Move cursor to character position (0,0).
```