XFORM2D: ASCII Syntax and Parameters XFORM2D

ASCII PARAMETERS, continued

reserved — A reserved parameter which must always be zero.

mat22 — A 2 x 2 transformation matrix. The order of the matrix elements is x basis vector and y basis vector. The elements must be specified in a 32-bit fixed-point format with a binary point between words; range is -128.0 to 127.0. The Model One ignores the high 8 bits.

mat32 — A 3 x 2 transformation matrix. The order of the matrix elements is x basis vector, y basis vector, and displacement vector. The first 4 elements must be specified in a 32-bit fixed-point format, as for mat22. The translation elements must be 16-bit signed integers.

XFORM2D: FORTRAN Subroutines and Parameters XFORM2D

FORTRAN SUBROUTINES

General Form: CALL XFM2D (TYPE, 0, MAT22, MAT32)

Relative, matrix form: CALL XFM2D0 (0, MAT22)

Absolute, matrix form: CALL XFM2D1 (0, MAT23)

Translate: CALL XFM2D2

Reset to Identity: CALL XRM2D3

Relative, vector form: CALL XF2D0X (0,XFVEC)

Absolute, vector form: CALL XF2D1X (0, XFVEC)

FORTRAN PARAMETERS

INTEGER*2 TYPE

REAL*4 MAT22(2,2), MAT32(3,2)

REAL*4 XFVEC(1)

MAT22 is defined as

```
MAT22(1,1) = x basis vector, x component
MAT22(1,2) = x basis vector, y component
MAT22(2,1) = y basis vector, x component
MAT22(2,2) = y basis vector, y component
```

MAT32 is defined as

```
MAT32(1,1) = x basis vector, x component
MAT32(1,2) = x basis vector, y component
MAT32(2,1) = y basis vector, x component
MAT32(2,2) = y basis vector, y component
MAT32(3,1) = displacement vector, x component
MAT32(3,2) = displacement vector, y component
```

XFVEC is defined as

```
XFVEC(1) = x basis vector, x component
XFVEC(2) = x basis vector, y component
XFVEC(3) = y basis vector, x component
XFVEC(4) = y basis vector, y component
XFVEC(5) = displacement vector, x component
XFVEC(6) = displacement vector, y component
```

For subroutine XF2D1X, only XFVEC(1) through XFVEC(4) is used.

XFORM2D: Host Binary Command Stream XFORM2D

Relative Transformation

[125] [type = 0] [0] [xb03][xb02][xb01][xb00] [xbl3][xbl2][xbl1][xbl0]
[yb03][yb02][yb01][yb00] [ybl3][ybl2][ybl1][ybl0]
(19 bytes)

Absolute Transformation

[125] [type = 1] [0] [xb03][xb02][xb01][xb00] [xbl3][xbl2][xbl1][xbl0]
[yb03][yb02][yb01][yb00] [ybl3][ybl2][ybl1][ybl0]
[highxt][lowxt] [highyt][lowyt] (23 bytes)

Translate

[125] [type = 2] (2 bytes)

Reset to Identity

[125] [type = 3] (2 bytes)

XFORM2D: Example XFORM2D

EXAMPLE

This example illustrates four different transformations: the identity, an absolute scaling, a relative rotation, and a translation.

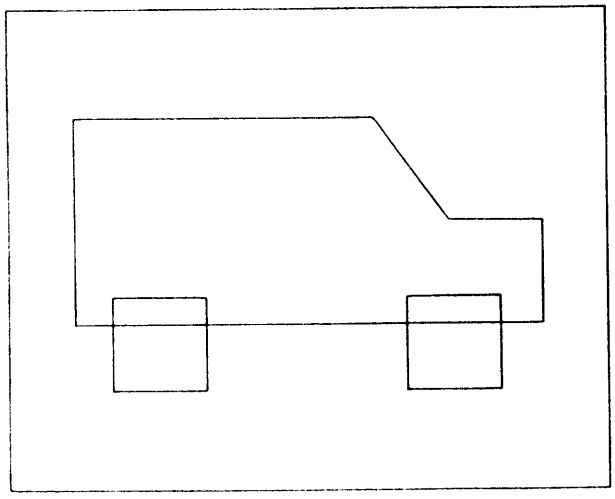
The Segment 1 which is executed in this example is defined in the example for the SEGDEF command.

```
! SEGREF 1              ; Execute (draw) Segment 1. (See figure labelled
                         IDENTITY TRANSFORMATION).

! XFORM2D ABS 0 .5 0 0 .5 0 0
                        ; Define an absolute transformation which scales
                         by half.
! VAL8 0                ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1              ; Draw Segment 1. (See figure labelled AFTER ABSOLUTE
                         SCALING.)

! XFORM2D REL 0 .866 .5 -.5 .866
                        ; Define a relative transformation of 30 degrees
                         rotation. This transformation is concatenated
                         with the absolute scaling transformation.
! VAL8 0                ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1              ; Draw Segment 1. (See figure labelled AFTER RELATIVE
                         ROTATION.)

! MOVABS 500,300        ; Move current point to 500,300.
! XFORM2D XLATE         ; Define a transformation which translates the WCS
                         origin to the old current point. This translation
                         is concatenated with the current transformation
                         of 1/2 scaling and 30 degree rotation.
! VAL8 0                ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1              ; Draw Segment 1. (See figure labelled AFTER
                         TRANSLATION.)

! XFORM2D RESET         ; Set the current transformation to the identity.
! VAL8 0                ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1              ; Draw Segment 1. (See the first figure labelled
                         IDENTITY TRANSFORMATION).
```

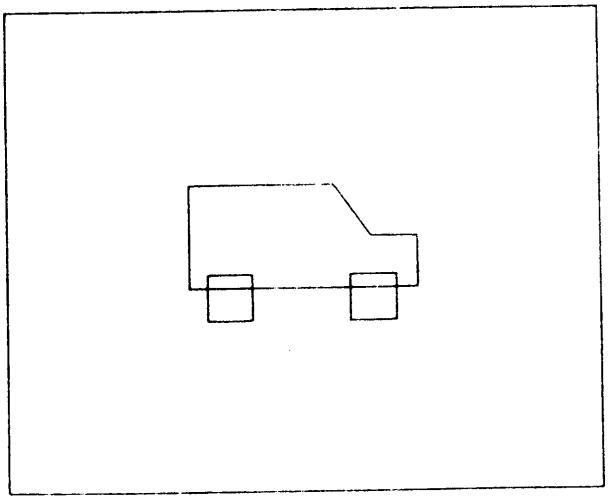
XFORM2D: Example XFORM2D

EXAMPLE, continued

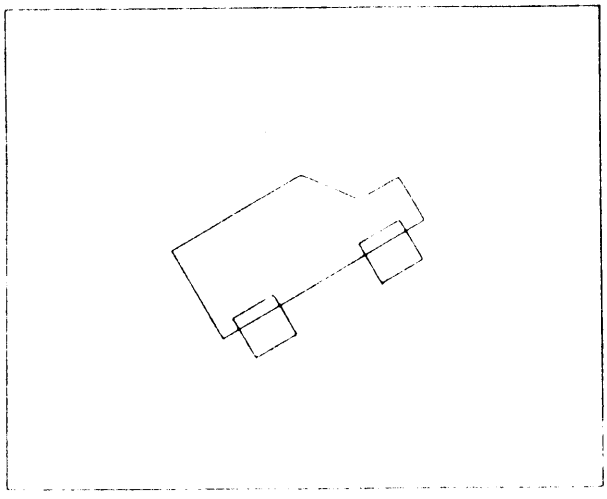
IDENTITY TRANSFORMATION



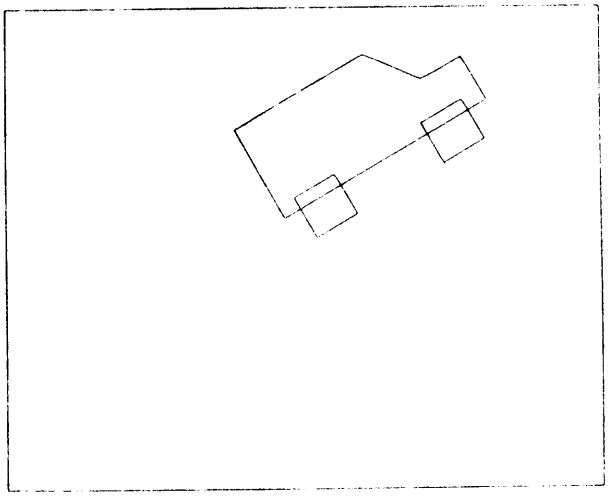
AFTER ABSOLUTE SCALING



AFTER RELATIVE ROTATION



AFTER TRANSLATION



XHAIR XHAIR

SYNTAX

ASCII	XHAIR num, flag
FORTRAN Call	CALL XHAIR (NUM, IFLAG)
Binary	[156] [num] [flag] (3 bytes)
	156 decimal = 234 octal = 9C hex

FUNCTION

The XHAIR command controls whether the crosshairs are displayed or not. Two crosshairs are available; num gives the crosshair number, 0 or 1. If flag = 0 or OFF, the crosshair num is not displayed. If flag = 1 or ON, the crosshair num is displayed.

Crosshairs, when displayed, take their location from CREG 5 (crosshair 0) and CREG 6 (crosshair 1).

The crosshair colors are determined by VREG 1 (crosshair 0) and VREG 2 (crosshair 1). The crosshair color is then XORed with the color in image memory to display the crosshair; the default crosshair value is 255.

The crosshairs look like a small plus sign (100 pixels x 100 pixels).

Using the crosshairs slightly increases the processing time for some commands. If maximum performance is important, turn off the crosshairs whenever possible. You can use the full-screen hardware cursor instead, which is controlled with the CURSOR command.

ASCII PARAMETERS

num	Crosshair number; crosshairs 0 and 1 are available.
flag	Flag = 1 or ON, enable crosshair; flag = 0 or OFF, disable crosshair.

FORTRAN PARAMETERS

INTEGER*2	NUM, IFLAG

XHAIR XHAIR

EXAMPLE

```
! XHAIR 0 ON            ; Crosshair 0 is displayed at center of screen;
                          color is default white.
! VLOAD 2 3             ; Load VREG 2 with 3 (default blue on 8-bit
                          system).
! CLOAD 6 100 100       ; Load CREG 6 with 100,100.
! XHAIR 1 ON            ; Crosshair 1 is displayed at 100,100; color is
                          blue.  Crosshair 1 takes its location from
                          CREG 6 and its color from VREG 2.
! XHAIR 1 OFF           ; Crosshair 1 is no longer displayed.
```

XMOVE XMOVE

SYNTAX

ASCII	XMOVE type, creg2, creg1	
FORTRAN Call	CALL XMOVE (TYPE, DSTREG, SRCREG)	
Binary	[127] [type] [creg2] [creg1]	(4 bytes)
	127 decimal = 177 octal = 7F hex	

FUNCTION

The XMOVE command performs a 2-D transformation on a specified x,y coordinate. The type parameter specifies which of the two XMOVE options is used:

- map from the World Coordinate Space (WCS) into the Device Coordinate Space (DCS), using the current 2-D transformation, or
- map from the DCS into the WCS, using the inverse of the current 2-D transformation.

The creg1 parameter contains the pre-transformed coordinate. Creg2 is automatically loaded with the transformed coordinate.

Note: If you set creg1 to 0 or CURPNT (the current point), you can see where geometry will be located with a new transformation before actually drawing the segments.

ASCII PARAMETERS

type The type of transformation.

0 = WCSDCS Map from WCS to DCS.
1 = DCSWCS Map from DCS to WCS.

creg1 The coordinate register containing the pre-transformed coordinate. You can use mnemonics for CREGs 0-6, 9, and 10. Refer to the table at the end of this Command Reference.

creg2 The coordinate register in which to store the transformed coordinate.

XMOVE XMOVE

FORTRAN PARAMETERS

INTEGER*2 TYPE, DSTREG, SRCREG

EXAMPLE

```
! XFORM2D ABS  0 2 0 0 2 0 0
                        ; Define an absolute transformation which scales
                          by 2.
! MOVABS 10 20          ; Move the current point to 10,20.

! XMOVE WSCDCS 20 CURPNT
                        ; Map the current point from WCS to DSC and store
                          in CREG 20.
! READCR 20             ; Read back the contents of CREG 20.
  00020 00040           ; The current point is scaled by 2.

! XMOVE DCSWCS 21 CURPNT
                        ; Map the current point from DSC to WSC and store
                          in CREG 21.
! READCR 21             ; Read back the contents of CREG 21.
  00005 00010           ; The current point is scaled by 1/2.
```

XYDIG XYDIG

SYNTAX

ASCII	XYDIG flag,fpformat,x,y	(if flag = 0 or XYSCALE)
	XYDIG flag, state	(if flag = 1 or CLIPWRAP)
FORTRAN Call	CALL XYDIG (FLAG, FPFORM, X, Y, STATE)	
Binary	[172] [flag = 0] [fpformat] [x] [y]	(11 bytes)
	[172] [flag = 1] [state]	(3 bytes)
	172 decimal = 254 octal = AC hex	

Note: [x] and [y] are each 4 bytes, arranged as High/Low/High/Low, with the first two bytes representing whole units and the second two representing fractional units.

FUNCTION

The XYDIG command has two different functions, depending on the value of flag.

When flag = 0 or XYSCALE, the XYDIG command allows you to define the scaling factor to be applied to the XY digitizer being used with the Model One. Coordinate data in CREG 2 will be determined in terms of the scaling factor you establish with the XYDIG command.

When flag = 1 or CLIPWRAP, the XYDIG command allows you to cause CREG 2 to be clipped or wrapped to screen coordinates.

The options for state are:

0 No clipping or wrapping (original case)
1 Clip, relative mouse only
2 Wrap

State = 1 (clipping) causes CREG 2 to be clipped at screen bounds (-640 to 639, -511 to 512 for a 1024 x 1280 system). Zooming or changing the clipping window (i.e. CREGs 9 and 10) does not affect this.

State = 2 (wrapping) observes the same limits as clipping, but instead of clipping, the CREG value will wrap around within the range of the screen coordinates.

FUNCTION, continued

Notes: The clipping/wrapping feature executes at the time that CREG 2 is updated from the graphics input device. Changes to CREG 2 that occur at other times (such as through CLOAD or CADD commands or via the bipolar processor) are not clipped or wrapped unless or until an incoming packet is processed from a mouse or tablet.

Wrapping can be used with any graphics input device. Clipping can only be used with a relative mouse, the M2 mouse. If you attempt to select clipping with a tablet configured, you will see the message: ILLEGAL PARAMETER.

ASCII PARAMETERS

flag — Flag indicates type of scaling to perform.

0 = XYSCALE Define scaling factor.
1 = CLIPWRAP Specify clipping or wrapping for CREG 2.

In binary format, flag must be 0 or 1.

fpformat — Specifies that the x and y parameters are in floating point format (16.16). Fpformat must be 3.

x,y — X and y scaling factors. In ASCII format, these may be a value from 0.00000 to 8.0. The binary format must be 16.16 format. The default scaling factor is 1,1.

state
0 No clipping or wrapping (default)
1 Clip, relative mouse only
2 Wrap

FORTRAN PARAMETERS

INTEGER*2 FLAG, FPFORM, STATE

REAL*4 X, Y

ZCLEAR ZCLEAR

SYNTAX

ASCII	ZCLEAR zdepth	
FORTRAN Call	CALL ZCLEAR (ZDEPTH)	
Binary	[90] [highzdepth] [lowzdepth]	(3 bytes)
	90 decimal = 120 octal = 50 hex	

FUNCTION

The ZCLEAR command clears depth buffer memory to the 16-bit zdepth specified. All the depth buffer memory is set to the specified value. The ZFUNCT command determines whether the coordinate system is z positive into or out of the screen (left- or right-handed coordinate system).

The most common values for straightforward hidden-surface removal are:

```
ZFUNCT GT        ZFUNCT of GREATER THAN
ZCLEAR -32768
```

and

```
ZFUNCT LT        ZFUNCT of LESS THAN
ZCLEAR 32767
```

For sectioning, you would use other ZCLEAR values to highlight the section cut. Example 2 illustrates sectioning.

ASCII PARAMETER

zdepth The 16-bit value to which depth buffer memory is cleared.

FORTRAN PARAMETER

INTEGER*2 ZDEPTH

Caution: Some FORTRAN compilers convert -32,768 to 0 when passed as INTEGER*2. Therefore, we recommend that you use -32,767 from FORTRAN.

EXAMPLE 1

In this example, 24-bit true color output is specified for the ZPATCH command. In order for the example to work properly on an 8-bit system, you must have set up the look-up table values for dithered true color. See the ZPATCH command for further explanation.

```
! ZFUNCT GT              ; Set the ZFUNCT to GREATER THAN.
! ZCLEAR -32768          ; Clear depth buffer memory to -32768.
! ZCLIP OFF              ; Set no clipping in z.

! ZPATCH 0 3 0,0,0 255,0,0 100,0,0 0,255,0 100,100,100 0,0,255

                         ; Send the first z patch: a triangle shading
                           from red at (0,0,0) to green at (100,0,0) to
                           blue at (100,100,100).

! ZPATCH 0 3 0,100,0 255,255,0 80,100,0 255,255,0 150,0,100 100,100,0

                         ; Send the second z patch: another triangle,
                           shading from deep yellow across the back
                           edge (0,100,0 to 80,100,0) to medium yellow
                           at the front corner (150,0,100). This triangle
                           intersects the first triangle at a z depth
                           value of 50.
```

Note: You can use this example to demonstrate the possible values for ZFUNCT, ZCLEAR, AND ZCLIP. To do this, change the parameters for any of these commands, and reissue the ZPATCH commands. You might want to put the ZPATCH commands into a macro to save retyping.

EXAMPLE 2: Sectioning

```
! ZFUNCT GT              ; Set the ZFUNCT to GREATER THAN.
! ZCLEAR 20              ; Clear depth buffer memory to 20.
! ZCLIP OFF              ; Set no clipping in z.

! ZPATCH 0 3 0,0,0 255,0,0 100,0,0 0,255,0 100,100,100 0,0,255

                         ; Send the first z patch: a triangle shading
                           from red at (0,0,0) to green at (100,0,0) to
                           blue at (100,100,100).
! ZCLIP ON 40 12         ; z clip for subsections of the image.

! ZPATCH 0 3 0,100,0 255,255,0 80,100,0 255,255,0 150,0,100 100,100,0

                         ; Send the second z patch: another triangle,
                           shading from deep yellow across the back edge
                           (0,100,0 to 80,100,0) to medium yellow at the
                           front corner (150,0,100). This triangle
                           intersects the first triangle at a z depth
                           value of 50.
! ZFUNC EQWIDE           ; Draw only the pixels whose z value equals the
                           z value of the current pixel within the
                           ZRANGE tolerance.
! ZRANGE 5 5             ; The range is 10 units wide (5 on each side).

! ZPATCH 0 3 -10,-10,40 255,255,255 110,10,40 255,255,255 110,110,40
        255,255,255

                         ; Highlight the cut at 40 on the second patch
                           and also highlight the first patch at 40.
```

ZCLIP ZCLIP

SYNTAX

ASCII	ZCLIP flag, negclip, posclip
FORTRAN Call	CALL ZCLIP (FLAG, NEAR, FAR)
Binary	[87] [flag] [highnegclip] [lownegclip] [highposclip] [lowposclip] (6 bytes)

87 decimal = 127 octal = 57 hex

FUNCTION

The ZCLIP command controls clipping of z values. Flag determines whether clipping is done at all: flag = 1 or ON enables clipping, flag = 0 or OFF disables clipping.

Negclip and posclip set the range of z values against which clipping is done. Only pixels whose z value is greater than negclip and less than posclip are drawn into image memory. Z clipping is done before ZFUNCT processing.

Note: Be sure to set posclip greater than negclip.

ASCII PARAMETERS

flag	Flag = 1 or ON enables clipping; flag = 0 or OFF disables clipping. (Default is OFF.)
negclip	The z value (from -32768 to 32767) which the new z value must be GREATER THAN.
posclip	The z value (from -32768 to 32767) which the new z value must be LESS THAN.

FORTRAN PARAMETERS

INTEGER*2 FLAG, NEAR, FAR

EXAMPLE

In this example, 24-bit true color output is specified for the ZPATCH command. In order for the example to work properly on an 8-bit system, you must have set up the look-up table values for dithered true color. See the ZPATCH command for further explanation.

```
! ZFUNCT GT          ; Set the ZFUNCT to GREATER THAN.
! ZCLEAR -32768      ; Clear depth buffer memory to -32768.
! ZCLIP ON 25 50     ; Clip to exclude those values less than 25 and
                      greater 50 than in z dimension.

! ZPATCH 0 3 0,0,0 255,0,0 100,0,0 0,255,0 100,100,100 0,0,255

                    ; Send the first z patch: a triangle shading
                      from red at (0,0,0) to green at (100,0,0) to
                      blue at (100,100,100).

! ZPATCH 0 3 0,100,0 255,255,0 80,100,0 255,255,0 150,0,100 100,100,0

                    ; Send the second z patch: another triangle, shading
                      from deep yellow across the back edge (0,100,0 to
                      80,100,0) to medium yellow at the front corner
                      (150,0,100). This triangle intersects the first
                      triangle at a z depth value of 50.
```

Note: You can use this example to demonstrate the possible values for ZFUNCT, ZCLEAR, AND ZCLIP. To do this, change the parameters for any of these commands, and reissue the ZPATCH commands. You might want to put the ZPATCH commands into a macro to save retyping.

ZFUNCT ZFUNCT

SYNTAX

ASCII	ZFUNCT function
FORTRAN Call	CALL ZFUNCT (IFUNCT)
Binary	[83] [function] (2 bytes)
	83 decimal = 123 octal = 53 hex

FUNCTION

The ZFUNCT command allows the depth buffer commands to update depth buffer memory conditionally, depending on the value of function. When function is set, the incoming z values are compared with the values already in image memory and then the pixel is drawn or not drawn based on the results.

ZFUNCT is especially useful for marking specified depths.

The table below shows the values for function and the corresponding mnemonics.

CODE	MNEMONIC	DESCRIPTION
0	OFF	Off: draw pixels without any comparison, but update z memory.
1	GT	Greater than: draw only those pixels with a depth greater than the current z depth.
2	LT	Less than: draw only those pixels with a depth less than the current z depth.
3	GE	Greater than or equal to. (Default)
4	LE	Less than or equal to.
5	EQ	Equal to.
6	NE	Not equal to.
7	EQWIDE	Equal within the tolerance given by the ZRANGE command.
8	NEWIDE	Not equal within the tolerance given by ZRANGE.
9	NONE	Draw pixels without any comparison, but do not write into z memory.

ZFUNCT ZFUNCT

ASCII PARAMETERS

function Range is 0 to 9; see previous page for values.

FORTRAN PARAMETER

```
CALL ZFUNCT (IFUNC)

INTEGER*2   IFUNC
```

EXAMPLE

In this example, 24-bit true color output is specified for the ZPATCH command. In order for the example to work properly on an 8-bit system, you must have set up the look-up table values for dithered true color. See the ZPATCH command for further explanation.

```
! ZFUNCT GT              ; Set the ZFUNCT to GREATER THAN.
! ZCLEAR -32768          ; Clear depth buffer memory to -32768.
! ZCLIP OFF              ; Set no clipping in z.

! ZPATCH 0 3 0,0,0 255,0,0 100,0,0 0,255,0 100,100,100 0,0,255

                         ; Send the first z patch: a triangle shading
                           from red at (0,0,0) to green at (100,0,0) to
                           blue at (100,100,100).

! ZPATCH 0 3 0,100,0 255,255,0 80,100,0 255,255,0 150,0,100 100,100,0

                         ; Send the second z patch: another triangle,
                           shading from deep yellow across the back
                           edge (0,100,0 to 80,100,0) to medium yellow
                           at the front corner (150,0,100). This triangle
                           intersects the first triangle at a z depth
                           value of 50.
```

Note: You can use this example to demonstrate the possible values for ZFUNCT, ZCLEAR, AND ZCLIP. To do this, change the parameters for any of these commands, and reissue the ZPATCH commands. You might want to put the ZPATCH commands into a macro to save retyping.

ZGRID ZGRID

ASCII SYNTAX

```
ZGRID  type, numu, numv, (gridlx, gridly, gridlz, gridlval)
```

FORTRAN CALL

```
CALL ZGRID (TYPE, NUMU, NUMV, IDATA)
```

HOST BINARY COMMAND STREAM

```
[85] [type=0] [numu] [numv] ( [highgridlx][lowgridlx]
     [highgridly][lowgridly]  [highgridlz][lowgridlz]
     [gridlred] [gridlgrn] [gridlblu])...
     (4 + numu * numv * 9  bytes)

[85] [type=1] [numu] [numv] ( [highgridlx][lowgridlx]
      [highgridly][lowgridly]  [highgridlz][lowgridlz]
      [gridlval])...
      (4 + numu * numv * 7  bytes)

[85] [type=2] [numu] [numv] ( [highgridlx][lowgridlx]
      [highgridly][lowgridly]  [highgridlz][lowgridlz]
      [highgridlval] [lowgridvall])...
      (4 + numu * numv * 8  bytes)

85 decimal = 125 octal = 55 hex
```

FUNCTION

The ZGRID command allows you to specify a grid of z patches. This is especially useful for parametric surfaces, such as superquadrics. It provides for considerable data compaction.

The type parameter indicates the type of color value that is given for each vertex: 8-bit intensity, 16-bit pseudo color, or 24-bit full color.

Numu and numv specify the dimensions of the grid.

For each vertex, you specify the x,y,z coordinates and the color value.

FUNCTION, continued

On an 8-bit system (or on a 24-bit system which is configured with a depth buffer), there are only 8 bit planes of image memory. If you specify 24-bit true color or 16-bit pseudo color output for the ZGRID command, the Model One performs dithering.

If you want to specify 8-bit or 16-bit pseudo color, you must ramp the look-up table values appropriately for your application.

If you want to specify 24-bit true color, you must set up the look-up table values in a specific way. The following VAX FORTRAN 77 program dither-ramps the look-up table for dithered true color.

```
INTEGER*2     IR, IG, IB, I
INTEGER*2     RG(0:7), B(0:3)
DATA   RG/0,36,72,108,144,180,216,255/      ! Define red and green levels
DATA   B/0,85,170,255/                      ! Define blue levels
    .
    .
DO I = 0,255
        IB = IAND(I,3)                      ! Load 3 bits blue
        IG = ISHFT(IAND(I,28),-2)           ! Load 3 bits green
        IR = ISHFT(IAND(I,224),-5)          ! Load 3 bits red
        CALL LUT8(I,RG(IR),RG(IG),B(IB)     ! Define color
```

Comments: IAND is a logical AND for integers.

ISHFT is a bitwise left shift, and using a negative value gives a right shift.

ZGRID ZGRID

ASCII PARAMETERS

type — Indicates the type of color value which is specified for each vertex.

- 0 — 24-bit color (8 bits each of red, green, and blue.
- 1 — 8-bit intensity value (0 to 255)
- 2 — 16-bit pseudo color which is interpreted as an 8.8 fixed point number (0.0 to 255.996).

numu — Number of vertices in the u direction.

numv — Number of vertices in the v direction.

gridlx, gridly, gridlz — 16-bit numbers for the x, y, z coordinates of the first vertex. Range is -32,768 to + 32,767.

gridlval — Color value associated with first vertex. The number of bits is determined by the type parameter.

FORTRAN PARAMETERS

```
CALL ZGRID (TYPE, NUMU, NUMV, IDATA)

INTEGER*2     NUMU, NUMV, TYPE, NPARMS, IDATA(1)
```

IDATA is an array of data for each vertex of the grid: x,y,z and color value (8, 16, or 24 bits) for each vertex. The data is IDATA should be organized so that the u-direction data increases more rapidly than the v-direction data.

Note: There is a way to convert REAL*4 color value data to the 8.8 fixed point format required by type 2. The following example illustrates how to do this.

```
INTEGER*2     I16COL
LOGICAL*1     I4COL(4)
EQUIVALENCE (I16COL, I4COL(2))
    .
    .
IDATA(1) = X
IDATA(2) = Y
IDATA(3) = Z
CALL FIXPT (1.1, I4COL)
IDATA(4) = I16COL
```

EXAMPLE

To enter a 5 x 6 grid, with 8-bit pseudo color values, you would enter the command

ZGRID 1 5 6 (30 vertices, each with x, y, z and 8-bit color value)

You must enter the vertices sequentially along each row or column.

ZOOM ZOOM

SYNTAX

ASCII	ZOOM fact	
FORTRAN Call	CALL ZOOM (IFACT)	
Binary	[52] [fact]	(2 bytes)
	52 decimal = 068 octal = 34 hex	

FUNCTION

The ZOOM command sets the display scale factor to an integer scale factor from 1 to 16, specified by fact.

When the scale factor is changed, the display is zoomed about the center of the screen.

ZOOM does not change the current screen origin, clipping window, or coordinate origin.

Any scale factor that is not a power of two (any other than 1, 2, 4, 8, or 16) will result in some displacement of the center screen pixels. This is because the screen origin can only be changed on 20 pixel boundaries horizontally.

ASCII PARAMETER

fact Display scale factor; range is 1 to 16.

FORTRAN PARAMETER

INTEGER*2 IFACT

EXAMPLE

```
! MOVABS 0 0          ; Move current point to 0,0.
! CIRCLE 30           ; Draw circle of radius 30.
! ZOOM 2              ; Set scale factor to 2:1.
! ZOOM 4              ; Set scale factor to 4:1.
! ZOOM 1              ; Restore scale factor to 1:1.
```

ZOOMIN ZOOMIN

SYNTAX

ASCII	ZOOMIN
FORTRAN Call	CALL ZOOMIN
Binary	[53] (1 byte)
	53 decimal = 065 octal = 35 hex

FUNCTION

The ZOOMIN command sets the display scale factor to two times the current display scale factor. If the current scale factor is 8 to 16, the ZOOMIN command restores the display scale factor to 1.

When the scale factor is changed, the display is zoomed about the center of the screen.

ZOOMIN does not change the current screen origin, clipping window, or coordinate origin.

Any scale factor that is not a power of two (any other than 1, 2, 4, 8, or 16) will result in some displacement of the center screen pixels. This is because the screen origin can only be changed on 20 pixel boundaries horizontally.

EXAMPLE

```
! MOVABS 0 0          ; Set current point to 0,0.
! CIRCLE 30           ; Draw circle of radius 30.
! ZOOMIN              ; Double current scale factor (set to 2:1).
! ZOOMIN              ; Double current scale factor (set to 4:1).
! ZOOMIN              ; Double current scale factor (set to 8:1).
! ZOOMIN              ; Double current scale factor (set to 16:1).
! ZOOMIN              ; Double current scale factor (restore to 1:1).
```

ZPATCH ZPATCH

ASCII SYNTAX

ZPATCH type, nvert, (xvertl, yvertl, zvertl, vertlval) ...

FORTRAN CALL

CALL ZPATCH (TYPE, NVERT, IDATA)

HOST BINARY COMMAND STREAM

[82] [type=0] [nvert] [highxvertl][lowxvertl] [highyvertl][lowyvertl] [highzvertl][lowzvertl] [vertlred] [vertlgrn] [vertlblu] (3 + nvert * 9 bytes)

[82] [type=1] [nvert] [highxvertl][lowxvertl] [highyvertl][lowyvertl] [highzvertl][lowzvertl] [vertlval] (3 + nvert * 7 bytes)

[82] [type=2] [nvert] [highxvertl][lowxvertl] [highyvertl][lowyvertl] [highzvertl][lowzvertl] [highvertlval] [lowvertlval] 3 + nverts * 8 bytes)

82 decimal = 122 octal = 52 hex

FUNCTION

The ZPATCH command interpolates, shades, and depth-buffers a polygonal patch; it does not change the current value or the current point. The shading is performed according to the flag set by SHMODE. The patch may have concavities in y but it must be completely convex in x. (No more than two edges of the same patch can cross a single scan line.)

The type parameter indicates the type of color value that is given for each vertex: 8-bit intensity, 16-bit pseudo color, or 24-bit full color.

Nvert specifies the number of vertices in the polygon.

For each vertex, you specify the x,y,z coordinates and the color value.

FUNCTION, continued

Dithering on an 8-Bit System

On an 8-bit system (or on a 24-bit system which is configured with a depth buffer), there are only 8 bit planes of image memory. If you specify 24-bit true color or 16-bit pseudo color output for the ZPATCH command, the Model One performs dithering.

If you want to specify 8-bit or 16-bit pseudo color, you must ramp the look-up table values appropriately for your application.

If you want to specify 24-bit true color, you must set up the look-up table values in a specific way. The following VAX FORTRAN 77 program dither-ramps the look-up table for dithered true color.

```
INTEGER*2     IR, IG, IB, I
INTEGER*2     RG(0:7), B(0:3)
DATA   RG/0,36,72,108,144,180,216,255/      ! Define red and green levels
DATA   B/0,85,170,255/                      ! Define blue levels
    .
    .
DO I = 0,255
         IB = IAND(I,3)                     ! Load 3 bits blue
         IG = ISHFT(IAND(I,28),-2)          ! Load 3 bits green
         IR = ISHFT(IAND(I,224),-5)         ! Load 3 bits red
         CALL LUT8(I,RG(IR),RG(IG),B(IB)    ! Define color
```

Comments: IAND is a logical AND for integers.

ISHFT is a bitwise left shift, and using a negative value gives a right shift.

ZPATCH ZPATCH

ASCII PARAMETERS

type — Indicates the type of color value which is specified for each vertex.

- 0 — 24-bit color (8 bits each of red, green, and blue).
- 1 — 8-bit intensity value (0 to 255).
- 2 — 16-bit pseudo color which is interpreted as an 8.8 fixed-point number (0.0 to 255.996).

nvert — Number of vertices in the polygon (0 to 255).

xvert1, yvert1, zvert1 — 16-bit numbers for x,y, and z coordinates of first vertex. Range is -32,768 to +32,767.

vert1val — Color value associated with first vertex. The number of bits is determined by the type parameter.

FORTRAN SUBROUTINES AND PARAMETERS

```
CALL ZPATCH (TYPE,NVERT,IDATA)

INTEGER*2     NVERT, IDATA(1), TYPE
```

IDATA is an array of data for each vertex in the polygon. Each vertex contains x,y,z data and color data (8, 16, or 24 bit).

Note: There is a way to convert REAL*4 color value data to the 8.8 fixed point format required by type 2 (16-bit pseudo color). The following example illustrates how to do this.

```
INTEGER*2     I16COL
LOGICAL*1     I4COL(4)
EQUIVALENCE (I16COL, I4COL(2))
    .
    .
    .
IDATA(1) = X
IDATA(2) = Y
IDATA(3) = Z
CALL FIXPT (1.1, I4COL)
IDATA(4) = I16COL
```

FORTRAN SUBROUTINES AND PARAMETERS, continued

Additional Subroutines

The following subroutines are provided for defining triangles and quadrilaterals, with 24-bit color values or 8-bit intensity values.

CALL ZP3INT (IDATA) For triangle with 8-bit intensity values.

CALL ZP4INT (IDATA) For quadrilateral with 8-bit intensity values.

CALL ZP3RGB (IDATA) For triangle with 24-bit RGB values.

CALL ZP4RBG (IDATA) For quadrilateral with 24-bit RGB values.

EXAMPLE

In this example, 24-bit true color output is specified for the ZPATCH command. In order for the example to work properly on an 8-bit system , you must have set up the look-up table values for dithered true color.

```
! ZFUNCT GT              ; Set the ZFUNCT to GREATER THAN.
! ZCLEAR -32768          ; Clear depth-buffer memory to -32768.
! ZCLIP OFF              ; Set no clipping in z.

! ZPATCH 0 3 0,0,0 255,0,0 100,0,0 0,255,0 100,100,100 0,0,255

                         ; Send the first z-patch: a triangle shading
                           from red at (0,0,0) to green at (100,0,0) to
                           blue at (100,100,100).

! ZPATCH 0 3 0,100,0 255,255,0 80,100,0 255,255,0 150,0,100 100,100,0

                         ; Send the second z-patch: another triangle,
                           shading from deep yellow across the back
                           edge (0,100,0 to 80,100,0) to medium yellow
                           at the front corner (150,0,100). This triangle
                           intersects the first triangle at a z-depth
                           value of 50.
```

Note: You can use this example to demonstrate the possible values for ZFUNCT, ZCLEAR, AND ZCLIP. To do this, change the parameters for any of these commands, and reissue the ZPATCH commands. You might want to put the ZPATCH commands into a macro to save retyping.

ZRANGE ZRANGE

SYNTAX

ASCII	ZRANGE dposz, dnegz
FORTRAN Call	CALL ZRANGE (DPOSZ, DNEGZ)
Binary	[54] [highdposz][lowdposz] [highdnegz][lowdnegz] (5 bytes) 54 decimal = 124 octal = 54 hex

FUNCTION

The ZRANGE command sets the tolerance for the ZFUNCT modes of EQWIDE (equal within tolerance) and NEWIDE (not equal within tolerance). Dposz gives the positive z change that is permitted from the z value at the pixel already there; dnegz gives the negative z change. This command is used for sectioning and contouring.

ASCII PARAMETERS

dposz 16-bit positive change (-32768 to 32767)
dnegz 16-bit negative change (-32768 to 32767)

FORTRAN PARAMETERS

INTEGER*2 DPOSZ, DNEGZ

Caution: Some FORTRAN compilers convert -32,768 to 0 when passed as INTEGER*2. Therefore, we recommend that you use -32,767 from FORTRAN.

ZRANGE ZRANGE

EXAMPLE

In this example, 24-bit true color output is specified for the ZPATCH command. In order for the example to work properly on an 8-bit system, you must have set up the look-up table values for dithered true color. See the ZPATCH command for further explanation.

```
! ZFUNCT GT              ; Set the ZFUNCT to GREATER THAN.
! ZCLEAR -32768          ; Clear depth-buffer memory to -32768.
! ZCLIP OFF              ; Set no clipping in z.

! ZPATCH 0 3 0,0,0 255,0,0 100,0,0 0,255,0 100,100,100 0,0,255

                         ; Send the first z patch: a triangle shading
                           from red at (0,0,0) to green at (100,0,0) to
                           blue at (100,100,100).

! ZRANGE 10 0            ; Draw only those new pixels within this ZRANGE.
! ZFUNCT EQWIDE          ; Set ZFUNCT to EQWIDE (equal within togerance).

! ZPATCH 0 3 0,100,0 255,255,0 80,100,0 255,255,0 150,0,100 100,100,0

                         ; Send the second z patch: another triangle,
                           shading from deep yellow across the back
                           edge (0,100,0 to 80,100,0) to medium yellow
                           at the front corner (150,0,100). This triangle
                           intersects the first triangle at a z depth
                           value of 50.
```

MNEMONICS FOR COORDINATE REGISTER AND VALUE REGISTER PARAMETERS

The table below lists the mnemonics which you can use for coordinate register parameters. For coordinate registers 11 through 63, there are no mnemonics.

CREG #	Mnemonic	Description
0	CURPNT	Current point. Starting point of graphics primitives commands. Updated by a MOVE or a DRAW.
1	JOYSTK	Unscaled XY digitizer location; current coordinates from digitizer. Updated automatically.
2	DIGTZR	Scaled XY digitizer location; current coordinates from digitizer. Updated automatically.
3	CORORG	Coordinate origin. Coordinates of the center of image memoroy.
4	SCRORG	Screen origin. Coordinates of the pixel in the center of the screen.
5	XHAIR0	Crosshair 0 location in image memory.
6	XHAIR1	Crosshair 1 location in image memory.
7-8		Reserved.
9	LWNORG	Clipping window origin. Lower-left corner of current clipping window. All vectors are clipped to this window.
10	UWNORG	Clipping window origin. Upper-right corner of current clipping window. All vectors are clipped to this window.

The table below lists the mnemonics which you can use for value register parameters. For value registers 4 through 63, there are no mnemonics.

VREG #	Mnemonic	Description
0	CURVAL	Current value; value used in all graphics primitives commands.
1	XR0VAL	Value used for Crosshair 0.
2	XR1VAL	Value used for Crosshair 1.
3	FILMSK	Fill mask used for area fills.