TEXTC TEXTC

SYNTAX

ASCII	TEXTC size, ang
FORTRAN Call	CALL TEXTC (ISIZE, IANG)
Binary	[146] [size] [highang][lowang] (4 bytes)
	146 decimal = 222 octal = 92 hex

FUNCTION

The TEXTC command sets the size of text and the baseline angle for subsequent text drawing commands: TEXT1, TEXT2, VTEXT1, and VTEXT2.

The size parameter specifies the scale factor, which may range from 0 to 255. Size = 16 is the default. A size of 32 doubles the size of the text. The parameter ang specifies the angle in which the text will be drawn. The angle is measured in one-degree increments counter-clockwise. For the TEXT1 and TEXT2 commands (horizontal text), zero degrees is the positive x axis. An angle of zero degrees causes the TEXT1 and TEXT2 commands to draw normally oriented text from left to right. For the VTEXT1 and VTEXT2 commands (vertical text), an angle of zero degrees causes the commands to draw normally oriented text from top to bottom.

Note: The TEXTC and TEXTN commands work independently of one another. Whichever command was executed last takes precedence.

ASCII PARAMETERS

size	Text size; range is 0 to 255. (Default 16)
ang	Text angle; range is -32,768 to 32,767. (Default 0)

FORTRAN PARAMETERS

INTEGER*2	ISIZE, IANG

EXAMPLE

```
! TEXTl Horizontal text      ; Draw text string in default size 16 and
                               default angle 0.
! TEXTC 16 30                ; Set text size to 16, angle to 30 degrees.
! TEXTl Angled text          ; Draw text at a 30 degree angle.
! MOVABS 0 50                ; Move current point to 0,50.
! TEXTC 32 0                 ; Set text size to 32, angle to 0 degrees.
! TEXTl Larger text          ; Draw double sized text.
```

TEXTDN TEXTDN

SYNTAX

ASCII	TEXTDN char, numvec, veclst
FORTRAN Call	CALL TEXTDN (ICHAR, NUMPTS, IPOINT)
Binary	[38] [char] [highnumvec][lownumvec] [highvl][lowvl] ... [highvn][lowvn] (4 + 2 * numvec bytes)

38 decimal = 046 octal = 26 hex

FUNCTION

The TEXTDN command defines a second font, Font 2, for text drawing. Font 2 is used by the TEXT2 and VTEXT2 commands. The parameter char specifies the character to be defined. Veclst defines the series of relative moves and draws that specify the character. Numvec specifies the number of vectors that will be drawn.

The TEXTDN command uses a 64 x 64 matrix for the definition of each character. Each move and draw of one unit in the matrix is equivalent to a pixel. With normal text size and angle (size = 16, ang = 0), the vectors will be drawn exactly as specified. To achieve varying spacing between user-defined characters, you should make the last part of the character definition a move to the beginning of the next character.

When transmitting data from the host, each relative move or draw requires two bytes of information. Each vector is specified as follows:

Bits 0-6:	7-bit two's complement relative y change; range -64 to 63.
Bit 7:	May be 0 or 1.
Bits 8-14:	7-bit two's complement relative x change; range -64 to 63.
Bit 15:	1 for DRAW, 0 for MOVE.

You may define 128 different characters. Font 2 is initialized to the same character set as Font 1. The font for a particular character only changes after the new character is defined with the TEXTDN command. This allows you to change just a portion of the character set.

The Font 1 characters are listed at the end of this command description for reference.

FUNCTION, continued

Note: The VTEXT2 command uses Font 2 (if you do not define a Font 2, it uses Font 1). The VTEXT2 command starts each character (after the first character) at a position 6 pixels to the left and 8 pixels down from the position at which the previous character stopped. This implies that to ensure proper alignment, VTEXT2 characters should be the same size as VTEXT1 characters, which use a 5 x 7 dot matrix.

ASCII PARAMETERS

char	8-bit integer specifying the number of the character to be defined.
numvec	16-bit integer specifying the number of vectors to be drawn.
veclst	16-bit integers specifying the vectors that define the character.

FORTRAN PARAMETERS

INTEGER*2 ICHAR, NUMPTS, IPOINT(1)

IPOINT is an integer array containing the list of moves and draws required to draw the character; the format described above is used.

FORTRAN EXAMPLE

The following FORTRAN program defines a special character, which is a box with an "x" in it. The call to the TEXTDN command replaces ASCII A with this new character. The subsequent call to the TEXT2 command draws the new character on the screen. If the host computer does not accept numbers in hex, you should convert them to octal.

```
program textdn_example
integer*2   text(8)          /* Dimension array for character definition.
call RTINIT ('-serial',0)    /* Initialize library for serial output.
call ENTGRA                  /* Enter graphics mode.

text(1) = #8A00          /* Draw in positive x 10 pixels.
text(2) = #800A          /* Draw in positive y 10 pixels.
text(3) = #F600          /* Draw in negative x 10 pixels.
text(4) = #8076          /* Draw in negative y 10 pixels.
text(5) = #8A0A          /* Draw in positive x 10 pixels and y 10 pixels.
text(6) = #7600          /* Move in negative x 10 pixels.
text(7) = #8A76          /* Draw in positive x 10 pixels and negative y 10
                         /* pixels.
text(8) = #1400          /* Move in positive x 20 pixels, putting the starting
                         /* point of the next character 20 pixels to the
                         /* right of the character being defined.

call TEXTDN (65,8,text) /* Replace ASCII A with new definition.
call TEXT2 (1,A)        /* Draw A on the screen.
call QUIT               /* Exit Graphics mode.
call exit               /* Exit from FORTRAN.
end                     /* End from FORTRAN.
```

TEXTDN TEXTDN

Decimal	Hex	Character	Decimal	Hex	Character	Decimal	Hex	Character
0	00	NUL	46	2E	. (period)	92	5C	\
1	01	SOH	47	2F	/	93	5D	]
2	02	STX	48	30	0	94	5E	^
3	03	ETX	49	31	1	95	5F	_ (line)
4	04	EOT	50	32	2	96	60	` (quote)
5	05	ENQ	51	33	3	97	61	a
6	06	ACK	52	34	4	98	62	b
7	07	BEL	53	35	5	99	63	c
8	08	BS	54	36	6	100	64	d
9	09	HT	55	37	7	101	65	e
10	0A	LF	56	38	8	102	66	f
11	0B	VT	57	39	9	103	67	g
12	0C	FF	58	3A	:	104	68	h
13	0D	CR	59	3B	;	105	69	i
14	0E	SO	60	3C	<	106	6A	j
15	0F	SI	61	3D	=	107	6B	k
16	10	DLE	62	3E	>	108	6C	l
17	11	DC1	63	3F	?	109	6D	m
18	12	DC2	64	40	@	110	6E	n
19	13	DC3	65	41	A	111	6F	o
20	14	DC4	66	42	B	112	70	p
21	15	NAK	67	43	C	113	71	q
22	16	SYN	68	44	D	114	72	r
23	17	ETB	69	45	E	115	73	s
24	18	CAN	70	46	F	116	74	t
25	19	EM	71	47	G	117	75	u
26	1A	SUB	72	48	H	118	76	v
27	1B	ESC	73	49	I	119	77	w
28	1C	FS	74	4A	J	120	78	x
29	1D	GS	75	4B	K	121	79	y
30	1E	RS	76	4C	L	122	7A	z
31	1F	US	77	4D	M	123	7B	{
32	20	SP	78	4E	N	124	7C	|
33	21	!	79	4F	O	125	7D	}
34	22	"	80	50	P	126	7E	~
35	23	#	81	51	Q	127	7F	DEL
36	24	$	82	52	R			
37	25	%	83	53	S			
38	26		84	54	T			
39	27	'	85	55	U			
40	28	(	86	56	V			
41	29	)	87	57	W			
42	2A	*	88	58	X			
43	2B	+	89	59	Y			
44	2C	, (comma)	90	5A	Z			
45	2D	-	91	5B	[			

TEXTN TEXTN

SYNTAX

ASCII	TEXTN xsize, ysize, xangle, yangle
FORTRAN Call	CALL TEXTN (XSIZE, YSIZE, XANGLE, YANGLE)
Binary	[169] [xsize] [ysize] [hixangle] [loxangle] [hiyangle] [loyangle] (7 bytes)
	169 decimal = 251 octal = A9 hex

FUNCTION

The TEXTN command, like the TEXTC command, sets the size of text and the baseline angle for subsequent text drawing commands. The TEXTN command allows independent control of the x and y components of the text size and angle. Note that if xsize = ysize and xangle = yangle, the TEXTN command is identical to the TEXTC command.

The following are some examples of the kinds of text you can create using the TEXTN command.

Tall skinny letters	xsize < ysize
Short fat letters	xsize > ysize
Forward italics	xangle = 0, yangle < 0
Backward italics	xangle = 0, yangle > 0
Forward italics at an angle	xangle > 0, yangle = 0
Backward italics at an angle	xangle < 0, yangle = 0

Note: The TEXTC and TEXTN commands work independently of one another. Whichever command was executed last takes precedence.

TEXTN TEXTN

ASCII PARAMETERS

xsize	The x component size; range is 0 to 255. (Default 16)
ysize	The y component size; range is 0 to 255. (Default 16)
xangle	The x angle; range is -32,768 to 32,767. (Default 0)
yangle	The y angle; range is -32,768 to 32,767. (Default 0)

FORTRAN PARAMETERS

INTEGER*2 XSIZE, YSIZE, XANGLE, YANGLE

EXAMPLE

```
! MOVABS 0 0              ; Move current point to 0,0.
! TEXTN   32 16 0 0       ; Set up for short, fat text.
! TEXTl Short fat text
! MOVABS 0 -25            ; Move current point to 0,-25.
! TEXTN 32 32 0 -45       ; Set up for double sized text, forward italics.
! TEXTl Forward italics
! MOVABS 0 -50            ; Move current point to 0,-50.
! TEXTN 16 16 0 45        ; Set up for normal sized text, backward italics.
! TEXTl Backward italics
```

TEXTRE TEXTRE

SYNTAX

ASCII	TEXTRE
FORTRAN Call	CALL TEXTRE
Binary	[177] (1 byte)
	177 decimal = 261 octal = B1 hex

FUNCTION

The TEXTRE command erases the definition of any user defined characters sent via the TEXTDN command and resets Font 2 equal to Font 1. The TEXTRE command frees all of the space used by previously defined characters.

VADD VADD

SYNTAX

ASCII	VADD vsum, vreg
FORTRAN Call	CALL VADD (IVSUM, IVREG)
Binary	[166] [vsum] [vreg] (3 bytes)
	166 decimal = 246 octal = A6 hex

FUNCTION

The VADD command adds the contents of value register vreg to the contents of value register vsum and places the result into value register vsum.

vsum = vsum + vreg

On an 8-bit system, only the first byte of the value register is used as an index into the look-up table. As a result, any information in the second and third bytes is ignored. On a 24-bit system in true color mode, all 3 bytes of information are maintained.

ASCII PARAMETERS

vsum, vreg Value registers; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

INTEGER*2 IVSUM, IVREG

EXAMPLE

```
! VLOAD 10 37 0 0        ; Load VREG 10 with 37,0,0.
! VLOAD 11 100 0 0       ; Load VREG 11 with 100,0,0.
! VADD 10 11             ; Add contents of VREG 10 and VREG 11 and place
                            result in VREG 10.
! READVR 10              ; Read contents of VREG 10.
  137 000 000               (Response from Model One.)
```

VAL1K VAL1K

SYNTAX

ASCII	VAL1K val
FORTRAN Call	CALL VAL1K (IRGBV)
Binary	[176] [val] (2 bytes)
	176 decimal = 260 octal = B0 hex

FUNCTION

The VAL1K command is provided for compatibility with the Model One/40. Unless you require this compatibility, use the VAL8 or VALUE commands instead.

The VAL1K command sets the current pixel value (VREG 0) to the value val. All operations which write into image memory use this value.

On an 8-bit system, val is used as an index into the look-up table. The value val is treated modulo 64. Therefore, VAL1K only allows addressing of look-up table indices 0 to 63.

On a 24-bit system with RGBTRU ON, the VAL1K command parses the value val as follows: the top 2 bits are discarded, the next 2 bits indicate the top bits for red, the next two bits indicate the top bits for green, and the lowest two bits indicate the top bits for blue.

ASCII PARAMETER

val Current pixel value; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 IRGBV

VAL8 VAL8

SYNTAX

ASCII	VAL8 val	
FORTRAN Call	CALL VAL8 (IVAL)	
Binary	[134] [val]	(2 bytes)
	134 decimal = 206 octal = 86 hex	

FUNCTION

The VAL8 command sets the current pixel value (VREG 0) to the value val. All operations which write into image memory use this value.

On an 8-bit system, the VAL8 command sets only the first byte of VREG 0 to the value val. The 8-bit value val is used as an index into the look-up table.

On a 24-bit system with RGBTRU ON, the VAL8 command sets each of the three bytes of VREG 0 (red, green, blue) to the same value val.

ASCII PARAMETER

val The current pixel value; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 IVAL

EXAMPLE 1: 8-Bit System

```
! VAL8 48              ; Set current pixel value to 48 (default LUT value
                        255,0,0: red).
! CIRCLE 50            ; Draw a red circle.
! LUT8 100 255 0 255   ; Change the color out for LUT index 100 to
                        255,0,255 (magenta).
! VAL8 100             ; Set current pixel value to 100 (magenta).
! CIRCLE 100           ; Draw a magenta circle.
```

EXAMPLE 2: 24-Bit System with RGBTRU ON

```
! PRMFIL ON          ; Select filled primitives.
! VAL8 48            ; Set current pixel value to r=48, g=48, b=48.
! CIRCLE 100         ; Draw a filled grey circle.
! VAL8 200           ; Set current pixel value to r=200, g=200, b=200.
! CIRCLE 50          ; Draw a circle in a lighter shade of grey.
```

VALUE VALUE

SYNTAX

ASCII	VALUE r, g, b
FORTRAN Call	CALL VALUE (IRED, IGRN, IBLU)
Binary	[6] [r] [g] [b] (4 bytes)

FUNCTION

The VALUE command changes the current pixel value (VREG 0) to the value specified by r, g, b. All operations which write into image memory use VREG 0, the current pixel value.

On a 24-bit system with RGBTRU ON, the VALUE command specifies a full 24 bits of pixel value data (8 bits each for red, green, and blue).

On an 8-bit system, only the first byte (red) is used as an index into the look-up table. The other two bytes are ignored. Therefore, the VALUE command is inefficient for an 8-bit system, and should be used only if compatibility with a Model One/40 is a requirement. Use VAL8 instead.

ASCII PARAMETERS

r, g, b	8-bit parameters specifying the red, green, and blue components of the current pixel value; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 IRED, IGRN, IBLU

EXAMPLE: 24-Bit System with RGBTRU ON

```
! VALUE 0 255 0          ; Set current pixel value to r=0, g=255, b=0.
! CIRCLE 50              ; Draw a green circle.
```

VECPAT VECPAT

SYNTAX

ASCII	VECPAT mask
FORTRAN Call	CALL VECPAT (MASK)
Binary	[46] [mask] (2 bytes)
	46 decimal = 056 octal = 2E hex

FUNCTION

The VECPAT command specifies the pattern of pixels to be repeated along every subsequent line that is drawn. The lines can be solid, dotted, dashed, or any other repetitive pattern that you specify.

The VECPAT command sets the vector generator pattern register to a 16-bit value specified by mask. Every time a pixel is generated by the vector generator, the mask is rotated by one bit. If the least significant bit of the mask is set to 1, the pixel will be written into image memory. If the bit is reset to 0, the pixel will be skipped.

The VECPAT command affects all line drawing commands and all graphics primitive commands, such as RECTAN, CIRCLE, and so on.

The VECPAT command also affects the filling of graphics primitives when PRMFIL is on, as well as the fill pattern used when the CLEAR command is executed.

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

ASCII PARAMETER

mask 16-bit pattern register mask; range is 0 to 65,535.

FORTRAN PARAMETER

INTEGER*2 MASK

EXAMPLE

```
! MOVABS 0 0              ; Move current point to 0,0.
! VECPAT #F0F0            ; Set vector pattern mask to draw lines with dashes.
! DRWABS 100 100          ; Draw dashed line.
! RECTAN 0                ; Draw a rectangle with dashed lines.
! VECPAT #AAAA            ; Set vector pattern mask to draw dotted lines.
! DRWABS 200 200          ; Draw dotted line.
! VECPAT #00FF            ; Set vector pattern mask to draw lines with long
                           dashes.
! DRWABS 0,200            ; Draw dashed line.
! MOVABS -200 -200        ; Move current point to -200,-200.
! PRMFIL ON               ; Select filled primitives.
! CIRCLE 50               ; Draw a circle filled with the pattern set with
                           the vector pattern mask.
```

VIDFORM VIDFORM

SYNTAX

ASCII	VIDFORM attrib, flag
FORTRAN Call	CALL VIDFRM (ATTR, FLAG)
Binary	[8] [attrib] [flag] (3 bytes)
	8 decimal = 010 octal = 08 hex

FUNCTION

The VIDFORM command is used to specify the video format. There are two basic forms of the VIDFORM command: VIDFORM INTERLACE and VIDFORM SPECIAL.

The VIDFORM INTERLACE (attrib = 0) command switches video control between interlacing (flag = 1 or ON) and noninterlacing (flag = 0 or OFF). The interlaced format is useful for interfacing to hardcopy devices that cannot accept the high video rate of the noninterlaced format. Also most RS-343 cameras require an interlaced format.

The VIDFORM SPECIAL (attrib = 1) command applies only to systems with special (non-standard) pixel PROMs. The VIDFORM SPECIAL ON command allows the selection of the default video format for the pixel PROMs in your Model One (even if this format is a custom video format). For standard systems, this is equivalent to a noninterlaced 1280 x 1024 format (e.g., VIDFORM SPECIAL ON has the same effect as VIDFORM INTERLACE OFF for systems with standard pixel PROMs).

The VIDFORM setting is saved with the SAVCFG command. The Model One powers up with the setting most recently saved.

Notes:

- If you have a 60 Hz monitor, but the configuration is saved as 30 Hz (VIDFORM INTERLACE ON), you need to do a VIDFORM INTERLACE OFF. Otherwise, objects drawn on the screen will be double in size and the center of objects are at the bottom of the screen.
- If you use VIDFORM SPECIAL ON, the DISCFG video format information may not reflect the actual video format for your system.
- If you use VIDFORM SPECIAL OFF, be sure to then set the video format that you want (e.g., VIDFORM INTERLACE ON).

VIDFORM VIDFORM

ASCII PARAMETERS

attrib — For host communications, attrib must equal 0 or 1. From the local terminal you can enter INTERLACE or SPECIAL.

flag — Flag = 1 or ON, enable interlacing or special format; flag = 0 or OFF, disable interlacing or special format. (Default is OFF.)

FORTRAN PARAMETERS

INTEGER*2 ATTR, FLAG

VLOAD VLOAD

SYNTAX

ASCII	VLOAD vreg, r, g, b
FORTRAN Call	CALL VLOAD (IVREG, IRED, IGRN, IBLU)
Binary	[164] [vreg] [r] [g] [b] (5 bytes)
	164 decimal = 244 octal = A4 hex

FUNCTION

The VLOAD command loads the value register specified by vreg with the pixel value specified by r,g,b.

On an 8-bit system, only the first byte (red) of the value register is used as an index into the look-up table. As a result, any information in the second and third bytes is ignored. On a 24-bit system in true color mode, all three bytes are maintained.

ASCII PARAMETERS

vreg	Value register; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.
r, g, b	Red, green, and blue components of the pixel value; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 IVREG, IRED, IGRN, IBLU

EXAMPLE

```
! VLOAD 13 50 0 0        ; Load VREG 13 with 50,0,0.
! READVR 13              ; Read contents of VREG 13.
  050 000 000               (Response from Model One.)
```

VMOVE VMOVE

SYNTAX

ASCII	VMOVE vdst, vsrc
FORTRAN Call	CALL VMOVE (IVDST, IVSRC)
Binary	[165] [vdst] [vsrc] (3 bytes)
	165 decimal = 245 octal = A5 hex

FUNCTION

The VMOVE command copies the value in the value register specified by vsrc to the value register specified by vdst.

On an 8-bit system, only the first byte of the value register is used as an index into the look-up table. As a result, any information in the second and third bytes is ignored. On a 24-bit system in true color mode, all 3 bytes of information are maintained.

ASCII PARAMETERS

vdst	Destination value register; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.
vsrc	Source value register; range is 0 to 63.

FORTRAN PARAMETERS

INTEGER*2 IVDST, IVSRC

EXAMPLE

```
! VLOAD 10 25 0 0        ; Load VREG 10 with 25,0,0.
! VMOVE 11 10            ; Move contents of VREG 10 into VREG 11.
! READVR 11              ; Read contents of VREG 11.
  025 000 000              (Response from Model One.)
```

VSUB VSUB

SYNTAX

ASCII	VSUB vdif, vreg
FORTRAN Call	CALL VSUB (IVDIF, IVREG)
Binary	[167] [vdif] [vreg] (3 bytes)
	167 decimal = 247 octal = A7 hex

FUNCTION

The VSUB command subtracts the contents of value register vreg from the contents of value register vdif and places the result into value register vdif.

vdif = vdif - vreg

On an 8-bit system, only the first byte of the value register is used as an index into the look-up table. As a result, any information in the second and third bytes is ignored. On a 24-bit system in true color mode, all 3 bytes of information are maintained.

ASCII PARAMETERS

vdif, vreg Value registers; range is 0 to 63. You can use mnemonics for VREGs 0 through 3. Refer to the table at the end of this Command Reference.

FORTRAN PARAMETERS

INTEGER*2 IVDIF, IVREG

EXAMPLE

```
! VLOAD 10 25 0 0        ; Load VREG 10 with 25,0,0.
! VLOAD 11 100 0 0       ; Load VREG 11 with 100,0,0.
! VSUB 11 10             ; Subtract contents of VREG 10 from contents of
                            VREG 11 and place result in VREG 11.

! READVR 11              ; Read contents of VREG 11.
  075 000 000               (Response from Model One.)
```

VTEXT1 VTEXT1

SYNTAX

ASCII	VTEXT1 string
FORTRAN Call	CALL VTEXT1 (STRLEN, STRING)
Binary	[147] [strlen] ([char1][char2]...[charn])
	147 decimal = 223 octal = 93 hex

FUNCTION

The VTEXT1 command draws a vertical text string into image memory with Font 1. If the text size is the default 16, each character uses 5 x 7 pixels. Descenders take an additional two pixels.

The parameter string specifies the text to be drawn. If the command is entered in ASCII mode from the local alphanumeric terminal or keyboard, then the string to be drawn is the set of ASCII characters which follow the VTEXT1 command on the command line. If the VTEXT1 command is sent from the host, then the length of the string must be specified. Strlen specifies the number of characters in the string, and is followed by strlen bytes containing the ASCII characters to be drawn.

The current point (CREG 0) specifies the starting point for the text string and remains unchanged by the command.

You can specify the size of the text and the baseline angle of the text using the TEXTC or TEXTN commands.

2-D transformations affect the rotation, scaling, and translation of text that is drawn with any of the text drawing commands.

ASCII PARAMETER

string The text to be drawn.

VTEXT1 VTEXT1

FORTRAN PARAMETERS

```
INTEGER*2     STRLEN, STRING(1)
```

STRLEN is an integer specifying the number of characters that are to be drawn. STRING is an integer array with two characters packed per 16-bit word, as in FORTRAN A2 format.

EXAMPLE

```
! MOVABS 0 0              ; Move current point to 0,0.
! VTEXT1 Vertical text    ; Draw text string in default size 16 and angle 0.
! MOVABS 20 0             ; Move current point to 20,0.
! TEXTC 32 0              ; Change text size to 32.
! VTEXT1 Double-sized text.
! TEXTC 16 90             ; Change text size to 16, angle to 90.
! VTEXT1 Text at a right angle.
```

VTEXT2 VTEXT2

SYNTAX

ASCII	VTEXT2 string
FORTRAN Call	CALL VTEXT2 (STRLEN, STRING)
Binary	[148] [strlen] ([charl][char2]...[charn])
	148 decimal = 224 octal = 94 hex

FUNCTION

The VTEXT2 command draws a vertical text string into image memory with Font 2. Font 2 is a user defined character set which is downloaded using the TEXTDN command.

At power-on or COLDstart, each character in Font 2 defaults to the same character in Font 1. When Font 2 is downloaded, each character replaces the power-on default definition.

The VTEXT2 command is issued in the same manner as the VTEXTl command. The current point (CREG 0) specifies the starting point for the text string and remains unchanged by the command. Strlen specifies the length of the string when text is not sent in ASCII mode.

Note: The VTEXT2 command starts each character (after the first character) at a position 6 pixels to the left and 8 pixels down from the position at which the previous character stopped. This implies that to ensure proper alignment, VTEXT2 characters should be the same size as VTEXTl characters, which use a 5 x 7 dot matrix.

ASCII PARAMETER

string The text to be drawn.

FORTRAN PARAMETERS

INTEGER*2 STRLEN, STRING(1)

STRLEN is an integer specifying the number of characters that are to be drawn. STRING is an integer array with two characters packed per 16-bit word, as in FORTRAN A2 format.

WAIT WAIT

SYNTAX

ASCII	WAIT frames	
FORTRAN Call	CALL WAIT (FRAMES)	
Binary	[61] [highframes] [lowframes]	(3 bytes)
	61 decimal = 075 octal = 3D hex	

FUNCTION

The WAIT command waits for the number of frame times specified by frames before continuing command execution. The WAIT command is often used for choreographing graphic displays and synchronizing updates with vertical blanking.

The delay in command execution in seconds is frames divided by 60. If the frames parameter is zero, command execution will stop until the next vertical blanking interval. The VIDFORM command does not affect the WAIT cycle.

ASCII PARAMETER

frames	16-bit parameter specifying the number of frame times to wait; range is 0 to 65,535.

FORTRAN PARAMETER

INTEGER*2	FRAMES

WARM WARM

SYNTAX

ASCII	WARM
FORTRAN Call	CALL WARM
Binary	[254] (1 byte)
	254 decimal = 376 octal = FE hex

FUNCTION

The WARM command reintializes the Model One firmware. This clears the serial input and output queues, reinitializes the functions set with DIP switches or the SYSCFG command, and returns the system to Alpha mode.

You can also execute the WARM command by pressing the WARMstart special character (default CTRL V). Note that disabling the WARMstart special character does not disable the WARM command.

WINDOW WINDOW

SYNTAX

ASCII	WINDOW x1, y1, x2, y2	
FORTRAN Call	CALL WINDOW (IX1, IY1, IX2, IY2)	
Binary	[58] [highx1] [lowx1] [highy1] [lowy1] [highx2] [lowx2] [highy2] [lowy2]	(9 bytes)

58 decimal = 072 octal = 3A hex

FUNCTION

The WINDOW command sets the current clipping window to the rectangle specified by x1,y1,x2,y2. The lower left corner of the window (CREG 9) is specified by x1 and y1. The upper right corner of the window (CREG 10) is specified by x2 and y2. X1 must be less than or equal to x2, and y1 must be less than or equal to y2.

All vectors and graphics primitives are clipped to the current clipping window.

The default clipping window is equal to the physical memory size.

ASCII PARAMETERS

x1, y1	16-bit parameters specifying the lower left corner of clipping window; range is -32,768 to 32,767.
x2, y2	16-bit parameters specifying the upper right corner of clipping window; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2	IX1, IY1, IX2, IY2

EXAMPLE

```
! WINDOW -20 -20 30 30      ; Set current clipping window to rectangle
                              with opposite corners at -20,-20 and 30,30.
! MOVABS 0 0                ; Move current point to 0,0.
! CIRCLE 25                 ; Draw circle of radius 25; only part of the
                              circle is displayed.
! TEXT1 Clipped text        ; Draw text string; only part of the text is
                              displayed.
! CLEAR                     ; Clear current window to current pixel value.
```

WMSK16 WMSK16

SYNTAX

ASCII	WMSK16 mask
FORTRAN Call	CALL WMSK16 (MASK) CALL WMSK16S (MASK)
Binary	[68] [highmask] [lowmask] (3 bytes)
	68 decimal = 104 octal = 44 hex

FUNCTION

The WMSK16 command specifies a 16-bit write mask for the Model One's image memory planes. Only the high byte of the write mask is used. The high eight bits correspond to the 8 bit planes of image memory. The low byte of the mask is ignored. If a bit is set (1), the corresponding bit plane is write-enabled; if the bit is cleared (0), the corresponding bit plane may not be written into.

When used in conjunction with the RMSK16 command (read mask), the WMSK16 command can be used for double buffering and animation by writing into those bit planes not displayed and displaying those bit planes not being written into.

The write and read masks can also be used to store multiple images in image memory and select them for display.

Note: The WMSK16 command is intended to be used with an 8-bit Model One system. Use the WRMASK command with a 24-bit system in true color mode.

ASCII PARAMETER

mask	The 16-bit write mask; range is 32,768 to 65,280 (#0100 to #FF00).

WMSK16 WMSK16

FORTRAN PARAMETER

CALL WMSK16 (MASK)
CALL WMSK16S (MASK)

INTEGER*2 MASK

The WMSK16 subroutine uses only the high 8 bits of MASK.

The WMSK16S subroutine swaps the high and low bytes of MASK. Thus you can specify the mask in the low byte of MASK, and do not have to swap the bytes yourself.

EXAMPLE

```
! MOVABS 0 0            ; Move current point to 0,0.
! PRMFIL ON             ; Select filled primitives.
! WMSK16 #0100          ; Write enable only bit plane 0.
! LUT8 1 255 0 0        ; Change the color out for LUT index 1 to red.
! VAL8 1                ; Set current pixel value to red.
! RECTAN 100 100        ; Draw a filled red square.

! MOVABS 200 200        ; Move current point to 200,200.
! WMSK16 #1000          ; Write enable only bit plane 4.
! LUT8 16 0 255 0       ; Change the color out for LUT index 16 to green.
! VAL8 16               ; Set current pixel value to green.
! CIRCLE 50             ; Draw green circle.

! RMSK16 #0100          ; Read enable only bit plane 0; only the red
                          square is displayed.
! RMSK16 #1000          ; Read enable only bit plane 4; only the green
                          circle is displayed.
! RMSK16 #1100          ; Read enable bit planes 0 and 4; both the circle
                          and the square are displayed.
```

WRAP WRAP

SYNTAX

ASCII	WRAP flag
FORTRAN Call	CALL WRAP (FLAG)
Binary	[203] [flag] (2 bytes)
	203 decimal = 313 octal = CB hex

FUNCTION

The WRAP command controls whether the alphanumeric window cursor will wrap around to the next line when the end of the line is reached.

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

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

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

ASCII PARAMETER

flag — Eight-bit parameter which specifies whether wraparound is enabled:

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

FORTRAN PARAMETER

INTEGER*2 FLAG

EXAMPLE

```
! WRAP OFF          ; Disable wrapping of the cursor when the end of
                      the text line is reached.
```

WRMASK: 8-Bit System WRMASK

SYNTAX

ASCII	WRMASK bitm, bankm
FORTRAN Call	CALL WRMASK (IBITM, IBANKM)
Binary	[157] [bitm] [bankm] (3 bytes)
	157 decimal = 235 octal = 9D hex

FUNCTION

These pages describe the use of the WRMASK command for a Model One 8-bit system. The use of the WRMASK command for a 24-bit system is described on the following pages.

For an 8-bit system, we recommend the use of the WMSK16 command instead of the WRMASK command.

The WRMASK command enables or disables write operations into the 8 image memory planes of the Model One. The two parameters, bitm and bankm, are used together to specify the write mask. Bankm specifies which pairs of bit planes to write-enable. Bitm specifies whether the lower planes, upper planes, or both planes of each pair are to be write-enabled.

bitm

bitm = 0	Do not write-enable either plane of the pairs specified by bankm.
bitm = 64	Write-enable plane 0, 2, 4, or 6 (lower planes of the pairs only).
bitm = 128	Write-enable planes 1, 3, 5, or 7 (upper planes of the pairs only).
bitm = 192	Write-enable both planes of the pairs specified by bankm.

bankm

Bits 7-4	Unused: must be zeros.
Bit 3	1: Write-enable bit planes 6 and 7.
	0: Do not write-enable bit planes 6 and 7.
Bit 2	1: Write-enable bit planes 4 and 5.
	0: Do not write-enable bit planes 4 and 5.
Bit 1	1: Write-enable bit planes 2 and 3.
	0: Do not write-enable bit planes 2 and 3.
Bit 0	1: Write-enable bit planes 0 and 1.
	0: Do not write-enable bit planes 0 and 1.

WRMASK: 8-Bit System WRMASK

ASCII PARAMETERS

bitm Values may be 0, 64, 128, 192, as described above.

bankm Range is 0 to 15, as described above.

FORTRAN PARAMETERS

INTEGER*2 IBITM, IBANKM

EXAMPLE

```
! WRMASK 64,7          ; Write enable bit planes 0, 2, and 4.
```

WRMASK: 24-Bit System WRMASK

SYNTAX

ASCII	WRMASK bitm, bankm
FORTRAN Call	CALL WRMASK (IBITM, IBANKM)
Binary	[157] [bitm] [bankm] (3 bytes)
	157 decimal = 235 octal = 9E hex

FUNCTION

This page describes the use of the WRMASK command for the Model One 24-bit system in true color mode. The use of the WRMASK command for an 8-bit system is described on the previous page.

The WRMASK command sets the write mask for the Model One's image memory planes. Bitm specifies the 8-bit write mask. If a bit in bitm is set (1), the corresponding bit plane in image memory is write-enabled; if a bit is cleared (0), the corresponding bit plane may not be written into.

The same 8-bit mask is used for the red, green, and blue image memory banks. You use the bankm parameter to specify which of the image memory banks to apply the mask to.

Bankm is set as follows:

Bits 7-3	Currently unused: must be zeros.
Bit 2	1: Write-enable red bank.
	0: Do not write-enable red bank.
Bit 1	1: Write-enable green bank.
	0: Do not write-enable green bank.
Bit 0	1: Write-enable blue bank.
	0: Do not write-enable blue bank.

ASCII PARAMETERS

bitm	8-bit write mask; range is 0 to 255.
bankm	8-bit parameter specifying which bank to write-enable; range is 0 to 7.

WRMASK: 24-Bit System WRMASK

FORTRAN PARAMETERS

```
INTEGER*2      IBITM, IBANKM
```

EXAMPLE

```
! WRMASK #01 7          ; Write-enable bit plane 0 in all three memory
                         banks.
```

XFORM2D Command: Overview XFORM2D

GENERAL SYNTAX

XFORM2D type, (reserved, arg)

FUNCTION

The XFORM2D command defines a linear transformation which maps 2-D coordinates in the 16-bit World Coordinate Space (WCS) into coordinates in the Display Coordinate Space (DCS).

Types of Transformations

You can define the following four types of transformations with the XFORM2D command.

XFORM2D REL	Specifies a 2 x 2 matrix which performs relative scaling and/or rotation centered about the current point; matrix elements are concatenated with the previous transformation. The new origin is at the old current point, and the new WCS currrent point is set to 0,0.
XFORM2D ABS	Specifies a 3 x 2 matrix which performs absolute scaling with translation, rotation centered about the current point with translation, or both with translation. The WCS current point is set to 0,0.
XFORM2D XLATE	The new WCS origin is set to the old current point, and the new WCS current point is set to 0,0. The DCS current point is updated.
XFORM2D RESET	Specifies a return to the default transformation (the identity). The WCS curent point is set to 0,0 and the DCS curent point is updated.

What You Have to Specify

The parameters which you specify for the XFORM2D command vary according to the type of transformation.

The RESET and XLATE types require only one parameter, type.

XFORM2D Command: Overview XFORM2D

What You Have to Specify, continued

The REL and ABS types require a reserved parameter, which must equal zero.

For the relative transformation type, REL, you must then specify a 2 x 2 matrix, which is made up of the transformed x basis and y basis vectors.

For the absolute transformation type, ABS, you must specify a 3 x 2 matrix, which is made up of the transformed x basis and y basis vectors, as well as the displacement vector.

The transformation matrices are described in more detail below.

Default Transformation: The Identity

The default transformation is the identity, which specifies no scaling, no rotation, and no translation.

```
     x basis vector =  (1, 0)
     y basis vector =  (0, 1)
displacement vector =  (0, 0)
```

To return to the default identity, use the RESET type of the XFORM2D command.

Translation With No Scaling or Rotation

Use the XLATE type of the XFORM2D command to specify a translation with no scaling or rotation. Before executing the command, move the current point with the MOVABS or MOVREL command to reflect the translation you wish to effect (i.e. the displacement vector).

Absolute Transformation: XFORM2D ABS

For the absolute transformation type, ABS, you must specify a 3 x 2 matrix, consisting of the transformed x basis and y basis vectors, as well as the displacement vector.

Listed below are the sets of vectors (or 3 x 2 matrices) which describe scaling with translation, rotation with translation, and scaling followed by rotation with translation.

XFORM2D Command: Overview XFORM2D

FUNCTION, continued

Absolute Transformation: XFORM2D ABS, continued

To specify no translation, use zeros in the last row.

To concatenate different transformations, apply the rules of matrix multiplication.

Scale by scale factors Sx and Sy
Translate by displacement vector D

```
     x basis vector =  (Sx, 0)
     y basis vector =  (0,  Sy)
displacement vector =  (Dx, Dy)
```

Rotate by A degrees
Translate by displacement vector D

```
     x basis vector =  (cos(A), sin(A))
     y basis vector =  (-sin(A), cos(A))
displacement vector =  (Dx, Dy)
```

Scale by scale factors Sx and Sy
Rotate by A degrees
Translate by displacement vector D

```
     x basis vector =  (Sx  cos(A),  Sy  sin(A))
     y basis vector =  (-Sx  sin(A), Sy  cos(A))
displacement vector =  (Dx, Dy)
```

Relative Transformation: XFORM2D REL

For the relative transformation type, REL, you must specify a 2 x 2 matrix. This matrix includes the x and y basis vectors as listed above, but does not include the displacement vector.

XFORM2D Command: Overview XFORM2D

CONTENTS

The following pages describe these aspects of the XFORM2D command:

- ASCII Syntax and Parameters
- FORTRAN Subroutines and Parameters
- Host Binary Command Stream, and
- Example.

XFORM2D: ASCII Syntax and Parameters XFORM2D

ASCII SYNTAX

Relative Transformation
(type = 0 (REL))

```
XFORM2D  type, reserved, mat22
```

Absolute Transformation
(type = 1 (ABS))

```
XFORM2D  type, reserved, mat32
```

Reset to Identity or Translate
(type = 2 (XLATE) or 3 (RESET))

```
XFORM2D  type
```

ASCII PARAMETERS

type — The type of transformation (1 byte).

- 0 = REL — Specifies a 2 x 2 matrix which performs relative scaling and/or rotation centered about the current point; matrix elements are concatenated with the previous transformation. The new origin is at the old current point, and the new WCS current point is set to 0,0.
- 1 = ABS — Specifies a 3 x 2 matrix which performs absolute scaling with translation, rotation centered about the current point with translation, or both with translation. The WCS current point is set to 0,0.
- 2 = XLATE — The new WCS origin is set to the old current point, and the new WCS current point is set to 0,0. The DCS current point is updated.
- 3 = RESET — Specifies a return to the default transformation (the identity). The WCS current point is set to 0,0 and the DCS current point is updated.