Raster Technologies

MODEL ONE/80

COMMAND REFERENCE

Revision 1.0

October 11, 1985

Part # 552-00031-001

Raster Technologies, Inc.
9 Executive Park Drive
North Billerica, MA 01862
(617) 667-8900

RASTER TECHNOLOGIES
MODEL ONE

NOTICE:

The information contained in this document is subject to change without notice.

RASTER TECHNOLOGIES DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS MATERIAL (INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE), EITHER EXPRESS OR IMPLIED. RASTER TECHNOLOGIES SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM ANY ERROR CONTAINED HEREIN, INCLUDING, BUT NOT LIMITED TO, FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS MATERIAL.

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual may cause interference with to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

INTRODUCTION TO THE MODEL ONE/80 COMMAND REFERENCE

Purpose

The Model One/80 Command Reference describes how to use each of the Model One/80's commands.

The Command Reference is intended to be a reference guide for users who have a basic understanding of the Model One/80's capabilities.

Other Documentation

Refer to the following documents for a more comprehensive discussion of how the commands interrelate and how to use the commands in graphics applications:

- Model One/80 Programming Guide
- Model One/80 Display List Firmware Reference
- Alphanumeric Terminal Emulation Guide
- Tektronix 4014 Emulation Guide, and
- Model One/80-S Shading and Depth-Buffering Reference.

The Model One/80 and Model One/10 Command Reference Card 2.0 provides a quick reference for the command set.

Information Provided for Each Command

The same types of information are provided for each command, in the same basic format. The following sections are included in each command description.

SYNTAX

The syntax section gives the ASCII syntax for the command, the FORTRAN subroutine call, and the host binary command stream.

FUNCTION

The function section explains the function of the command, including what it does, the reasons for using it, and its variations.

Continued on next page

INTRODUCTION TO THE MODEL ONE/80 COMMAND REFERENCE, continued

ASCII PARAMETERS

The ASCII parameters section lists and describes each parameter. The description specifies the range for the parameter, the default value (if any), the input formats, any required parameter combinations, and any other needed information.

FORTRAN PARAMETERS

The FORTRAN parameters section lists the type declaration for parameters in the FORTRAN subroutine. For subroutines which can accept parameters of different types, this section also notes how the parameters should be declared in the application program.

This section does not describe each parameter in detail. You can refer to the ASCII parameters section for more detailed information.

EXAMPLE

For most of the commands, an example is provided to illustrate how the command is used. Unless noted otherwise, the examples are intended to run on 8-bit systems. You can modify the examples to run on 24-bit systems by changing the value commands.

Note: These examples assume that the default conditions are in effect.

Unsupported Commands for the Model One/80

The following commands appear on the HELP list, but are not supported for the Model One/80: LUTRTE, MODE1K, and PIXEL4.

Internal Diagnostics Commands

The following commands are supported but are designed for internal diagnostics purposes only, and should not be used in general applications programs: DIAGS, HDBRPT, MPEEK, MPOKE, and TWIDDLE.

MODEL ONE/80 COMMANDS BY FUNCTIONAL GROUPS

Alphanumeric Terminal Emulation

ALPHEM	DIRCUR	HOME	SELWIN
BOLD	GETCUR	MOVCUR	SETCUR
DEFWIN	GETPOS	OVRSTK	SETSIZ
DELWIN	GETWIN	SCROLL	WRAP

Display Control

BLANK	RDMASK	VIDFORM	XHAIR
CURSOR	RGBTRU	WINDOW	ZOOM
FIRSTP	RMSK16	WMSK16	ZOOMIN
MODDIS	VECPAT	WRMASK	

Display List (Optional)

DELPID	PUSH	SEGDEF	SETATR
EXMODE	RDPICK	SEGDEL	SETGL
INCPID	RDPID	SEGEND	SYSTAT
INSPID	RDREG	SEGINI	XFORM2D
IGNORE	RDXFORM	SEGINQ	XMOVE
PICKID	SEGAPP	SEGREF	
POP	SEGCOP	SEGREN	

Graphics Primitives

ARC	DRWI	POLYGN	TEXTRE
AREA1	DRWREL	PRMFIL	VAL1K
AREA2	FILMSK	RECREL	VAL8
CIRCI	FLOOD	RECTAN	VALUE
CIRCLE	MOV2R	RECTI	VTEXT1
CIRCXY	MOV3R	TEXT1	VTEXT2
CLEAR	MOVABS	TEXT2	
DRW2R	MOVI	TEXTC	
DRW3R	MOVREL	TEXTDN	
DRWABS	POINT	TEXTN	

Image Transmission

PIXEL8		PIXMOV
PIXELS		PMCTL
PIXFUN		RUNLEN
		RUNLN8

MODEL ONE/80 COMMANDS BY FUNCTIONAL GROUPS, continued

Interactive Device Support

BUTTBL
BUTTON
FLUSH
RDPIXR
XYDIG

Look-Up Table

BLINKC
BLINKD
BLINKE
BLINKR
LUT8
LUTA
LUTB
LUTG
LUTR
LUTRMP

Macro Programming

MACDEF
MACEND
MACERA
MACRO

Readback

RDCNFG
RDMODE
READBU
READCR
READER
READF
READP
READVR
READW
READWE

Register Operations

CADD
CLOAD
CMOVE
CORORG
CSUB
SCRORG
VADD
VLOAD
VMOVE
VSUB

Shading and Depth-Buffering (Optional)

FOLD
SHMODE
ZCLEAR
ZCLIP
ZFUNCT
ZGRID
ZPATCH
ZRANGE

MODEL ONE/80 COMMANDS BY FUNCTIONAL GROUPS, continued

Software Development

*	MAP
ALPHAO	NULL
DEBUG	PEEK
DELAY	POKE
HELP	PORTO
HOSTO	REPLAY
KICK	WAIT

System Configuration

ASCII	QUIT
COLD	RDCNFG
DFTCFG	SAVCFG
DISCFG	SPCHAR
GENLOCK	SYSCFG
MEMSEL	TEKEM

---

* (ASTERISK) * (ASTERISK)

SYNTAX *

FUNCTION

The * (asterisk) command allows use of program comments when in pure ASCII format mode, as set with the ASCII command. Any characters following the asterisk (and before the carriage return) are ignored.

EXAMPLE

```
!*              ; This is a comment which is ignored.
```

ALPHAO ALPHAO

SYNTAX

ASCII	ALPHAO string
FORTRAN Call	CALL ALPHAO (STRLEN, STRING)
Binary	[180] [strlen] ([char1] [char2]...[charn])
	180 decimal = 264 octal = B4 hex

FUNCTION

The ALPHAO (ALPHA Output) command outputs a text string on the local alphanumeric display screen. The text to be output is specified by string. If the command is being entered in ASCII mode from the local alphanumeric terminal or keyboard, string is the set of ASCII characters remaining on the command line. If the command is not being sent in ASCII mode, then strlen must be given. Strlen contains the number of characters in the string followed by a string with strlen bytes.

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

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

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

\\ Indicates the backslash character (\) itself.

ASCII PARAMETER

string The text to be printed.

ALPHAO ALPHAO

FORTRAN PARAMETERS

INTEGER*2 STRLEN, STRING(1)
STRLEN is an integer specifying the number of characters that are to be output.

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

EXAMPLE

```
! ALPHAO ABCDEF 1 2 3          ; Output text string "ABCDEF 1 2 3".
! ALPHAO WXYZ                  ; Output text string "WXYZ".
```

FORTRAN EXAMPLE

```
CRLF = CHAR(10)//CHAR(12)
CALL ALPHAO(10,'12345678'//CRLF)
```

ALPHEM ALPHEM

SYNTAX

ASCII	ALPHEM flag
FORTRAN Call	CALL ALPHEM (FLAG)
Binary	[194] [flag] (2 bytes)
	194 decimal = 302 octal = C2 hex

FUNCTION

The ALPHEM command turns the alphanumeric terminal emulator or the VT100 emulator on or off. When either emulator is on, the selected window of the display monitor is used as a scrolling text window. Any characters typed at the Raster Technologies keyboard are sent to the window. The window must be defined (see DEFWIN) and selected (SELWIN) before it can be used.

Note: You can configure the Model One to power up into either the alphanumeric terminal window 0 or the VT100 window by using the SYSCFG command. Refer to the pages called SYSCFG EMULATE.

ASCII PARAMETER

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

FORTRAN PARAMETER

INTEGER * 2 FLAG

EXAMPLE

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

ARC ARC

SYNTAX

ASCII ARC rad, a1, a2

FORTRAN Call CALL ARC (IRAD, IA1, IA2)

Binary

[17] [highrad][lowrad] [highal][lowal] [higha2][lowa2]

17 decimal = 021 octal = 11 hex

FUNCTION

The ARC command draws a circular arc with its center at the WCS current point (CREG 0), a radius of rad, the starting angle a1, and ending angle a2. The angles are specified in integer degrees measured counter-clockwise. An angle of 0 degrees is the positive x axis. An angle of 90 degrees is the positive y axis. The arc is drawn counter-clockwise from the starting angle to the ending angle.

Notes

All arcs where the absolute value of the difference between a1 and a2 is greater than 360 are clipped to 360 degrees. Thus, in PIXFUN XOR mode, the following two commands would produce equivalent results:

```
! ARC 100 0 380
! ARC 100 0 360
```

There is a slight (but visible) instance of round-off error in angles between 315 and 360 degrees. This round-off has a tendency to flatten out lines from end-points to the center of the arc (for filled arcs only). This is really only discernable in PIXFUN XOR mode.

Drawing arcs while a skewed transformation is in place may lead to unpredictable rotations. A skewed transformation is a transformation in which the x scale and y scale are not equal.

ARC ARC

ASCII PARAMETERS

rad 16-bit integer value specifying the radius of the arc; range is -32,768 to +32,767.

a1 16-bit integer value specifying the starting angle; range is -32,768 to +32,767.

a2 16-bit integer value specifying the ending angle; range is -32,768 to +32,767.

FORTRAN PARAMETERS

INTEGER*2 IRAD, IA1, IA2

EXAMPLE

```
! MOVABS 0 0          ; Move current point to location  0,0.
! ARC 75 45 135       ; Draw circular arc of radius 75, starting
                         at 45 degrees and ending at 135 degrees.
! ARC 100 -30 60      ; Draw circular arc of radius 100, starting
                         at 30 degrees and ending at 60 degrees.
! PRMFIL ON           ; Select filled primitives.
! ARC 40 -10 40       ; Draw filled (pie shape) arc of radius 40,
                         starting at -10 degrees, ending at 40 degrees.
```

AREA1 AREA1

SYNTAX

ASCII	AREA1
FORTRAN Call	CALL AREA1
Binary	[19] (1 byte)
	19 decimal = 023 octal = 13 hex

FUNCTION

The AREA1 command performs area filling. AREA1 sets all pixels in a given closed region to the current value (VREG 0). The region extends from the WCS current point (CREG 0) outward in all directions until a boundary whose pixel values differ from the pixel value at the current point is reached. The boundary pixel values and the original pixel values are ANDed with the fill mask (VREG 3) before the comparison is made. The fill mask is set by the FILMSK command.

EXAMPLE

```
! LUT8 1 255,0,0          ; Set the color out for LUT index 1 to red.
! LUT8 2 255,255,255      ; Set color out for LUT index 2 to white.
! LUT8 3 0,255,0          ; Set color out for LUT index 3 to green.
! VAL8 1                  ; Set current pixel value to 1 (red).
! CIRCLE 30               ; Draw red circle of radius 30.
! MOVABS 25 20            ; Move current point to 25,20.
! CIRCLE 35               ; Draw red circle of radius 35.
! VAL8 2                  ; Set current pixel value to 2 (white).
! AREA1                   ; Begin area fill in white from 25,20 outward to
                            boundary.
! MOVABS 10 -10           ; Move current point to 10,-10.
! VAL8 3                  ; Set current pixel value to 3 (green).
! AREA1                   ; Begin area fill in green from 10,-10 outward to
                            boundary (the intersection of the two circles).
```

AREA2 AREA2

SYNTAX

ASCII	AREA2 vreg	
FORTRAN Call	CALL AREA2 (IVREG)	
Binary	[20] [vreg]	(2 bytes)
	20 decimal = 024 octal = 14 hex	

FUNCTION

The AREA2 command performs area filling. AREA2 sets all pixels in a given closed region to the current value (VREG 0). The region extends from the WCS current point (CREG 0) outward until a boundary of pixels whose value is specified by value register vreg or VREG 0 is reached. The boundary pixel values and the value specified by value register vreg are ANDed with the fill mask (VREG 3) before the comparison is made. The fill mask is set by the FILMSK command.

ASCII PARAMETER

vreg	The value register containing the boundary pixel value; 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 PARAMETER

INTEGER*2 IVREG

EXAMPLE

```
! LUT8 1 255,0,0        ; Set the color out for LUT index 1 to red.
! LUT8 2 0,0,255        ; Set the color out for LUT index 2 to blue.
! LUT8 3 0,255,255      ; Set the color out for LUT index 3 to cyan.
! VAL8 1                ; Set the current pixel value to 1 (red).
! CIRCLE 100            ; Draw red circle of radius 100.
! VAL8 2                ; Set the current pixel value to 2 (blue).
! CIRCLE 50             ; Draw blue circle of radius 50.
! VAL8 3                ; Set current pixel value to 3 (cyan).
! VLOAD 9 3,0,0         ; Load VREG 9 with 3,0,0.
! AREA2 9               ; Begin area fill in cyan. Boundary pixel value
                          is in VREG 9. (The inner blue circle is
                          overwritten because it is not drawn in boundary
                          pixel value.)
```

ASCII ASCII

SYNTAX

ASCII	ASCII flag
FORTRAN Call	CALL ASCII (IFLAC)
Binary	[155] [flag] (2 bytes)
	155 decimal = 233 octal = 9B hex

FUNCTION

The ASCII command sets the host input port. If flag = 0 or OFF, the host port input command stream is interpreted as pure ASCII format; all subsequent commands must be sent from the host to the Model One exactly as they would be typed in locally. If flag = 0 or OFF, the host port input command stream is interpreted as the default 8-bit binary or ASCII hex, as set with the SYSCFG command.

You never use the ASCII command in local mode. The ASCII command is usually sent from the host as a binary character string sequence.

Pure ASCII format requires many more characters to be sent from the host to execute a series of commands and should be used only when the command stream must be directly interpreted by the programmer or user rather than the Model One.

PARAMETER

flag	Flag = 1 or ON sets host port to pure ASCII format; flag = 0 or OFF (default) sets host port to binary or ASCII hex.

FORTRAN PARAMETER

INTEGER*2	IFLAG

BLANK BLANK

SYNTAX

ASCII	BLANK flag
FORTRAN Call	CALL BLANK (IFLAG)
Binary	[49] [flag] (2 bytes)
	49 decimal = 061 octal = 31 hex

FUNCTION

The BLANK command allows you to blank or unblank the screen. If flag = 1 or ON, the screen is blanked, and the contents of image memory are no longer displayed. This frees all cycles of image memory for writing and reading operations by the Model One's CPU and bipolar processor. This allows up to 3.5 times the pixel writing rate of an unblanked screen.

ASCII PARAMETER

flag	Flag = 1 or ON blanks the screen; flag = 0 or OFF restores normal video.

FORTRAN PARAMETER

INTEGER*2 IFLAG

EXAMPLE

```
! PRMFIL ON          ; Select filled primitives.
! CIRCLE 50          ; Draw circle of radius 50.
! BLANK 1            ; Blank screen.
! CIRCLE 100         ; Draw circle of radius 100.
! BLANK 0            ; Restore normal video.
```

BLINKC BLINKC

SYNTAX

ASCII	BLINKC
FORTRAN Call	CALL BLINKC
Binary	[35] (1 byte)
	35 decimal = 043 octal = 23 hex

FUNCTION

The BLINKC command clears the blink table, and disables blink of all LUT locations. The LUT entries are set to entryl of their blink values, as set with the BLINKE command.

BLINKD BLINKD

SYNTAX

ASCII	BLINKD lut, index
FORTRAN Call	CALL BLINKD (LUT, INDEX)
Binary	[33] [lut] [index] (3 bytes)
	33 decimal = 041 octal = 21 hex

FUNCTION

The BLINKD command disables blinking of the LUT location specified by lut. The index gives the pixel value in image memory that will address this location in the LUT.

Caution: The color that is left in the LUT location as a result of the BLINKD command is undefined. Therefore, you should use the appropriate LUT command to set the desired color.

ASCII PARAMETERS

lut	lut=7, disable all look-up table components. lut=1, disable the blue component of LUT. lut=2, disable the green component of LUT. lut=4, disable the red component of LUT.
index	LUT index location; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2	LUT, INDEX

BLINKE BLINKE

SYNTAX

ASCII	BLINKE lut, index, entry1, entry2
FORTRAN Call	CALL BLINKE (LUT, INDEX, IENT1, IENT2)
Binary	[32] [lut] [index] [entry1] [entry2]
	32 decimal = 040 octal = 20 hex

FUNCTION

The BLINKE command enables blinking of the LUT location specified by lut. The index specifies the address of the location in the LUT to blink. Entry1 and entry2 are swapped back and forth at the rate specified by the BLINKR command.

PARAMETERS

lut	lut=7, enable all look-up table components. lut=1, enable blue component of LUT. lut=2, enable green component of LUT. lut=4, enable red component of LUT.
index	LUT index location; range is 0 to 255.
entry1	First LUT entry to blink between; range is 0 to 255.
entry2	Second LUT entry to blink between; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2	LUT, INDEX, IENT1, IENT2

EXAMPLE

```
! BLINKE 7 63 255 125        ; Blink location 63 in all LUT components,
                              blinking between entries 255 and 125.
! BLINKC                     ; Location 63 has value 255 in all LUT
                              components.
! BLINKE 7 63 100 255        ; Blink between entries 100 and 255.
! BLINKC                     ; Location 63 has value 100 in all LUT
                              components.
```

BLINKR BLINKR

SYNTAX

ASCII	BLINKR frames
FORTRAN Call	CALL BLINKR (FRAMES)
Binary	[34] [frames] (2 bytes)
	34 = 042 octal = 22 hex

FUNCTION

The BLINKR command sets the blink rate to frames frame times. This rate is specified as the number of frame times between swapping between the two LUT entries. One frame time is 1/60 second.

The VIDFORM command has no effect on the blink rate.

ASCII PARAMETER

frames Each frame time is 1/60 second; range is 0 to 255.

FORTRAN PARAMETER

INTEGER*2 FRAMES

BOLD BOLD

SYNTAX

ASCII	BOLD flag
FORTRAN Call	CALL BOLD (FLAG)
Binary	[204] [flag] (2 bytes)
	204 decimal = 314 octal = CC hex

FUNCTION

The BOLD command controls the drawing of text for the alphanumeric terminal emulator and the VT100 emulator. BOLD ON or BOLD 1 (the default) draws bold text, in proportion to the scaling factor. BOLD OFF or BOLD 0 draws normal (thin) text.

ASCII PARAMETER

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

FORTRAN PARAMETER

INTEGER*2 FLAG

EXAMPLE

```
! BOLD ON                ; Enables drawing of bold text.
```

BUTTBL BUTTBL

SYNTAX

ASCII	BUTTBL number, macnum
FORTRAN Call	CALL BUTTBL (INDEX, MACNUM)
Binary	[170] [number] [macnum] (3 bytes)
	170 decimal = 252 octal = AA hex

FUNCTION

The BUTTBL command is used to load the Button Table. Number gives the button number. Macnum gives the macro number to execute when the button is pressed (for positive number) or released (for negative number).

Buttons can be found on locator devices such as a tablet or mouse. The number of buttons available is a function of the type of device used. A button can always be generated by using the BUTTON command.

ASCII PARAMETERS

number	Button number; range is -63 to 63.
macnum	Macro number; range is 0 to 255.

FORTRAN PARAMETERS

INTEGER*2 INDEX, MACNUM

EXAMPLE

```
! MACDEF 37          ; Start definition of Macro 37.
$ CIRCLE 50          ; Draw a circle.
$ FLUSH              ; Flush the button queue.
$ MACEND             ; End macro definition.
! BUTTBL 13 37       ; Execute Macro 37 when Button 13 is pressed.
! BUTTBL -13 37      ; Execute Macro 37 when Button 13 is released.
```

BUTTON BUTTON

SYNTAX

ASCII	BUTTON index
FORTRAN Call	CALL BUTTON (INDEX)
Binary	[171] [index] (2 bytes)
	171 decimal = 253 octal = AB hex

FUNCTION

The BUTTON command executes the macro specified by the Button Table at location index. This command performs the same function as pressing function button index on the Raster Technologies keyboard or on locator devices such as a tablet or a mouse.

Negative button index values indicate that the button macro will be executed when the button is released rather than when the button is pressed.

ASCII PARAMETER

index Button number; range is -63 to 63.

FORTRAN PARAMETER

INTEGER*2 INDEX

EXAMPLE

```
! MACDEF 15        ; Begin definition of Macro 15.
$ CIRCLE 50        ; Draw a circle.
$ MACEND           ; End Macro definition.
! BUTTBL 21 15     ; Execute Macro 15 when Button 21 is pressed.
! BUTTON 21        ; Simulate having pushed Button 21.
```

CADD CADD

SYNTAX

ASCII	CADD csum, creg
FORTRAN Call	CALL CADD (ICSUM, ICREG)
Binary	[162] [csum] [creg]
	162 decimal = 242 octal = A2 hex

FUNCTION

The CADD command adds the contents of the coordinate register specified by creg to the contents of the coordinate register specified by csum and places the result in the coordinate register specified by csum.

ASCII PARAMETERS

csum, creg — Coordinate registers for addition; 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 PARAMETERS

INTEGER*2 ICSUM, ICREG

EXAMPLE

```
! CLOAD 25 100 150      ; Load CREG 25 with 100,150.
! CLOAD 26 10 20        ; Load CREG 26 with 10,20.
! CADD 25 26            ; Add the contents of CREG 26 to CREG 25
                           and place result in CREG 25.
! READCR 25             ; Read the contents of CREG 25
  00110 00170              (Response from Model One.)
! CADD 25 26            ; Add the contents of CREG 26 to CREG 25
                           and place result in CREG 25.
! READCR 25             ; Read contents of CREG 25
  00120 00190              (Response from Model One.)
```

CIRCI CIRCI

SYNTAX

ASCII	CIRCI creg
FORTRAN Call	CALL CIRCI (ICREG)
Binary	[16] [creg] (2 bytes)
	16 decimal = 020 octal = 10 hex

FUNCTION

The CIRCI command draws a circle with the center point of the circle at the WCS current point (CREG 0) and the point specified by coordinate register creg on the circumference of the circle. The CIRCI command is useful for drawing circles with the radius controlled by an interactive device such as the digitizing tablet.

ASCII PARAMETER

creg	Coordinate register for point on circumference; 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 100 100        ; Move to location 100,100.
! CIRCI 4               ; Draw circle whose center is 100,100 and
                           whose circumference includes the location
                           given in CREG 4.
! MOVABS 120 150        ; Move to location 120,150.
! CLOAD 27 200 220      ; Draw circle whose center is at 120,150 and
                           whose circumference includes the location
                           given in CREG 27 (200,220).
```

CIRCLE CIRCLE

SYNTAX

ASCII	CIRCLE rad
FORTRAN Call	CALL CIRCLE (IRAD)
Binary	[14] [highrad] [lowrad] (3 bytes)
	14 decimal = 16 octal = 0E hex

FUNCTION

The CIRCLE command draws a circle of radius rad, with the center at the WCS current point (CREG 0). The circle is drawn in the current pixel value. A circle of radius zero sets the current point to the current pixel value.

If the circle is transformed, it is drawn as a circle, with the radius defined so that the point represented by the coordinates (current-point-X plus rad, Y) is on the circumference.

ASCII PARAMETER

rad	16-bit integer value specifying the circle radius; range is -32,768 to 32,767.

FORTRAN PARAMETER

INTEGER*2 IRAD

EXAMPLE

```
! MOVABS 100 150        ; Move current point to 100,150.
! CIRCLE 30             ; Draw circle of radius 30 centered at 100,150.
! MOVREL 10 0           ; Move current point by 10,0 to 110,150.
! CIRCLE 20             ; Draw circle of radius 20 centered at 110,150.
! CIRCLE 10             ; Draw circle of radius 10 centered at 110,150.
```

CIRCXY CIRCXY

SYNTAX

ASCII	CIRCXY x, y
FORTRAN Call	CALL CIRCXY (IX, IY)
Binary	[15] [highx][lowx] [highy][lowy] (5 bytes)
	15 decimal = 017 octal = 0F hex

FUNCTION

The CIRCXY command draws a circle with the center of the circle at the WCS current point (CREG 0) and the point x,y on its circumference.

If the circle is transformed, it is drawn as a circle, with the circumference point interpreted according to the current transformation.

ASCII PARAMETERS

x, y	16-bit integer values specifying the coordinates for the point on the circumference; range is - 32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2 IX, IY

EXAMPLE

```
! MOVABS 0 0          ; Move current point to 0,0.
! CIRCXY 30 30        ; Draw circle centered at 0,0 with 30,30 on
                         its circumference.
! CIRCXY 30 40        ; Draw circle centered at 0,0 with 30,40 on
                         its circumference.
! CIRCXY 50 50        ; Draw circle centered at 0,0 with 50,50 on
                         its circumference.
```

CLEAR CLEAR

SYNTAX

ASCII	CLEAR
FORTRAN Call	CALL CLEAR
Binary	[135] (1 byte)
	135 decimal = 207 octal = 87 hex

FUNCTION

The CLEAR command changes all pixels in the currently defined window to the current pixel value (VREG 0). Pixels outside the current clipping window are unchanged. The corners of the current window are held in CREGs 9 and 10.

CLEAR is affected by the pixel function. The CLEAR command uses the vector pattern register.

EXAMPLE

```
! WINDOW -50 -50 50 50  ; Set current clipping window to rectangle at
                          center of screen.
! CLEAR                 ; Clear current window to current pixel value
                          (default white).
! PIXFUN XOR            ; Set pixel processor mode to XOR.
! LUT8 3 0,0,255        ; Set the color out for LUT index 3 to blue.
! VAL8 3                ; Set current pixel value to 3 (blue).
! CLEAR                 ; Clear current window to current pixel value;
                          window turns yellow because PIXFUN is set to XOR.
```

CLOAD CLOAD

SYNTAX

ASCII	CLOAD creg, x, y
FORTRAN Call	CALL CLOAD (ICREG, IX, IY)
Binary	[160] [creg] [highx][lowx] [highy][lowy] (6 bytes)
	160 decimal = 240 octal = A0 hex

FUNCTION

The CLOAD command loads the coordinate register specified by creg with the values specified by x, y.

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

ASCII PARAMETERS

creg	Coordinate register to be loaded; 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.
x, y	16-bit integer values specifying the (x, y) coordinate pair to be loaded into the specified coordinate register; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2	ICREG, IX, IY

EXAMPLE

```
! CLOAD 17 100 150       ; Load CREG 17 with 100,150.
! READCR 17              ; Read contents of CREG 17.
  00100 00150             (Response from Model One.)
! CLOAD 17 50 -50        ; Load CREG 17 with 50,-50.
! READCR 17              ; Read contents of CREG 17.
  00050 -00050            (Response from Model One.)
```

CMOVE CMOVE

SYNTAX

ASCII	CMOVE cdst, csrc
FORTRAN Call	CALL CMOVE (ICDST, ICSRC)
Binary	[161] [cdst] [csrc] (3 bytes)
	161 decimal = 241 octal = A1 hex

FUNCTION

The CMOVE command copies the contents of the coordinate register specified by csrc to the coordinate register specified by cdst. Any data in cdst is replaced by the new data.

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

ASCII PARAMETERS

cdst, csrc	Coordinate registers; 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 PARAMETERS

INTEGER*2 ICDST, ICSRC

EXAMPLE

```
! CLOAD 25 100,150      ; Load CREG 25 with 100,150.
! CLOAD 26 20,-50       ; Load CREG 26 with 20,-50.
! READCR 26             ; Read contents of CREG 26.
  00020 -00050           (Response from Model One.)
! CMOVE 26 25           ; Move contents of CREG 25 into CREG 26.
! READCR 26             ; Read contents of CREG 26.
  00100 00150            (Response from Model One.)
```

COLD COLD

SYNTAX

ASCII	COLD
FORTRAN Call	CALL COLD
Binary	[253]
	253 decimal = 375 octal = FD hex

FUNCTION

The COLD command performs a Model One/80 COLDstart, equivalent to pushing the START button on the back panel. COLD executes the Model One/80's restart sequence and diagnostics. The COLDstart state of the Model One/80 is defined as follows.

```
MODDIS 1                ; Load default LUT.
MEMSEL 0                ; Select first memory unit in 24-bit
                          configuration as image memory.
ZOOM 1                  ; Set scale factor of 1:1.
CORORG 0,0              ; Set Coordinate Origin to 0,0.
SCRORG 0,0              ; Set Screen Origin to 0,0.
WINDOW x1,y1,x2,y2      ; Set clipping window to image memory bounds.
PRMFIL 0                ; Select unfilled primitives.
MACERA 0 through
  MACERA 255            ; Erase all macros.
TEXTRE                  ; Erase all user defined fonts.
VAL8 0                  ; Set current pixel value to 0.
VECPAT #FFFF            ; Draw solid lines.
PIXFUN 0                ; Set INSert mode.
PIXCLP 0                ; Use wraparound on pixel clipping.
CLEAR                   ; Clear image memory to 0.
VAL8 255                ; Set current pixel value to 255.
BUTTBL -63,0 to
  BUTTBL 63,0           ; Set all button table entries to zero.
DELAY 0                 ; No delay.
WMSK16 #FF00            ; Enable all bit planes for writing.
RMSK16 #FF00            ; Enable all bit planes for reading.
```

The Model One/80 is in ALPHA mode after the COLDstart.

COLD COLD

Notes: If you want to reset the Model One/80 to a state other than the above, you will want to set up a standard command to execute after the COLDstart command.

In addition, if the COLD command is sent from the host, you must wait roughly three seconds before sending any more data.

EXAMPLE

In the following example, the response from the Model One/80 will vary depending on the configuration of your system.

```
! COLD                    ; COLDstart the Model One/80.

Image memory size:   01280 x 01024
Display size:        01280 x 01024
Memory units present:
 000 FF00
 001 FF00
 002 FF00

Raster Technologies Model One/80, Revision 1.7.
```

CONFIG CONFIG

SYNTAX

ASCII CONFIG macro, fill, text, alpha, unused, reserved

FORTRAN Call CALL CONF80 (MACROS,FILL,TEXT,ALPHA,UNUSED)

Binary [36] [himacro][lomacro] [hifill][lofill]
[hitext][lotext] [hialpha][loalpha]
[hiunused][lounused] [00][00][00][00][00][00]
(17 bytes)

36 decimal = 44 octal = 24 hex

FUNCTION

The CONFIG command allocates RAM space for internal storage of macro definitions, area fills, downloaded text, and alpha windows.

The default RAM space allocation is:

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

The MAP command shows the current space allocation.

If the allocation given with the CONFIG command exceeds 16K bytes, the message "not enough space for definition" is given, and the allocation remains unchanged.

Caution: The CONFIG command destroys all currently defined macros, text, and alpha windows. You should never use it inside macros or segments.

ASCII PARAMETERS

macro	Space for macro definitions.
fill	Scratch space for area fill, polygon fill, and zpatch.
text	Space for downloaded text fonts (TEXTDN).
alpha	Space for alphanumeric window definitions.
unused	Free space.
reserved	These six bytes are reserved and must be set to zero.

CONFIG CONFIG

FORTRAN PARAMETERS

INTEGER*2 MACROS, FILL, TEXT, ALPHA, UNUSED

EXAMPLE

```
! CONFIG #2000 #1600 #0000 #1000      ; Reallocate all downloaded text
                                       font space to area/polygon fill
                                       scratch space.
! MAP                                 ; Display new configuration.

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

CORORG CORORG

SYNTAX

ASCII	CORORG x, y
FORTRAN Call	CALL CORORG (X, Y)
Binary	[55] [highx][lowx] [highy][lowy] (5 bytes)
	55 decimal = 067 octal = 37 hex

FUNCTION

The CORORG command sets the coordinate origin register (CREG 3) to the point specified by x,y. The contents of this register are added to all incoming coordinates; all coordinates are relative displacements from the coordinate origin. The CORORG command resets all other coordinate registers and should be used only immediately after a COLDstart.

The CORORG command should be used in conjunction with the SCRORG command to maintain the coordinate origin at the center of the screen.

After a CORORC command, CREG 3 will contain the cumulative offset from the original origin after COLDstart (0, 0). To return to the original CORORG, execute a CORORG command with values set opposite:

```
READCR 3
  00100 -00100
CORORG -00100 00100
```

ASCII PARAMETERS

x, y	16-bit integer values specifying the new coordinate origin; range is -32,768 to +32,767.

FORTRAN PARAMETERS

INTEGER*2	X, Y

CORORG CORORG

EXAMPLE

```
! COLD                    ; Perform a COLDstart.
! CORORG 100 -100         ; Set coordinate origin register (CREG 3) to
                           100,-100.
! CIRCLE 50               ; Draw a circle (it appears in the middle of the
                           screen).
! SCRORG -100 100         ; Set screen origin in conjunction with the change
                           you made in the coordinate origin; circle shifts
                           towards the lower right corner.
! READCR 3                ; Read back CREG 3 (coordinate origin).
  00100 -00100            ; Model One response.
! CORORG -100 100         ; Reset coordinate origin to original position;
                           circle goes further to the bottom left corner.
! READCR 3                ; Read back CREG 3 (coordinate origin).
  00000 00000             ; Model One response (coordinate origin returns to
                           0,0, the default).
! SCRORG 100 -100         ; Reset screen origin to original position; circle
                           returns to the middle of the screen.
```