RDPID RDPID

EXAMPLE

```
! SEGDEF 2
$ PICKID 30
$ PUSH CREG CURPNT
$ MOVREL -100 -100
$ RECREL 200 200
$ POP CREG CURPNT
$ SEGEND


! SETGL SEGREG 2          ; Load the segment ID register (SEGREG) with 2.
! SETGL PIDREG 30         ; Load the pick ID register (PIDREG) with 30.
! RDPID 10                ; Read back commands with segment ID 2 and pick
                           ID 30.  Insert a carriage return after each 10
                           elements.

  00016
 117 000 000 002 255 156 255 156 137 000
 200 000 200 118 000 000 000 000 000 000

                           ; The 16 is the number of bytes returned.  The
                           following numbers are the opcodes and parameters
                           of the commands labelled pick ID 30 in Segment 2.
```

RDPIXR RDPIXR

SYNTAX

ASCII	RDPIXR vreg
FORTRAN Call	CALL RDPIXR (IVREG)
Binary	[175] [vreg] (2 bytes)
	175 decimal = 257 octal = AF hex

FUNCTION

The RDPIXR command reads the pixel value from image memory at the current point (CREG 0) and places the value into the value register specified by vreg.

On an 8-bit system, only the first byte of the current pixel value is kept. The second and third bytes are ignored. On a 24-bit system with RGBTRU ON, all three bytes of information are maintained.

ASCII PARAMETER

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.

FORTRAN PARAMETER

INTEGER*2 IVREG

EXAMPLE 1: 8-Bit System

```
! VAL8 45              ; Change current pixel value to 45.
! POINT                ; Set current point to current pixel value.
! RDPIXR 13            ; Read pixel value at current point and place
                          value in VREG 13.
! READVR 13            ; Display contents of VREG 13.
  045 000 000             (Response of 8-bit Model One.)
```

EXAMPLE 2: 24-Bit System with RGBTRU ON

```
! VALUE 255 100 100    ; Change current pixel value to r=255, g=100, b=100.
! POINT                ; Set current point to current pixel value.
! RDPIXR 14            ; Read pixel value at current point and place
                          value in VREG 14.
! READVR 14            ; Display contents of VREG 14.
  255 100 100             (Response of 24-bit Model One.)
```

RDREG RDREG

SYNTAX

ASCII	RDREG reserved
FORTRAN Call	CALL RDREG (ISEG, IPID)
Binary	[224] [reserved = 0] (2 bytes)
	224 decimal = 340 octal = E0 hex

FUNCTION

The RDREG command reads back the current segment ID number (stored in SEGREG) and current pick ID number (stored in PIDREG), as set by one of these commands:

- EXMODE PICK followed by a SEGREF (picking)
- SETGL SEGREG and SETGL PIDREG, or
- RDPICK.

The RDREG command returns two 32-bit quantities.

This command can be used to determine if there was a pick hit after picking. If there was no pick hit, RDREG returns two negative ones.

ASCII PARAMETER

reserved An 8-bit reserved parameter (must = 0).

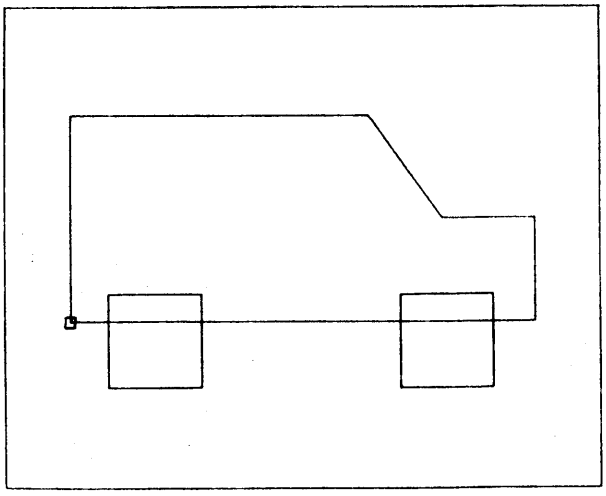
FORTRAN PARAMETER

Output Parameters: INTEGER ISEG, IPID

ISEG Current segment ID number.
IPID Current pick ID number

EXAMPLE

This example illustrates how to use the RDREG command. It also shows how the EXMODE PICK and RDPICK commands update the segment ID and pick ID registers. The Segment 1 executed in this example is defined in the SEGDEF command example, and is illustrated below.



```
! CLOAD 19 -500,-160     ; Set center of pick aperture (see figure).
! EXMODE PICK            ; Enter pick mode.
! SEGREF 1               ; Execute (pick) Segment 1.
! RDPICK PICKS 0 0       ; Read back number of pick hits encountered and
                            number of pick hits recorded.
  00002 00002            ; The Model One returns:  two pick hits encountered
                            and recorded.
! RDREG 0                ; Read back contents of segment ID and pick ID
                            registers.
 0000000001 0000000010  ; The Model One returns: segment ID is 1, pick ID
                            is 10.  The registers are loaded with the first
                            pick hit.
! RDPICK SEGPID 1 2      ; Read back segment ID/pick ID pairs, starting
                            with the first pick buffer entry and returning
                            information on 2 entries.
 0000000001 0000000010  ; The Model One reads back information on 2 entries.
 0000000001 0000000020
! RDREG 0                ; Read back contents of segment ID and pick ID
                            registers.
 0000000001 0000000020  ; The Model One returns: segment ID is 1, pick ID
                            is 20.  The registers have been updated by the
                            RDPICK command with the last entry read back.
```

RDXFORM RDXFORM

SYNTAX

ASCII	RDXFORM type
FORTRAN Call	CALL RDXFRM (FUNC,MATRIX,XBASIS,YBASIS,ZBASIS,OFFSET)
Binary	[214] [type] (2 bytes)
	214 decimal =326 octal = D6 hex

FUNCTION

The RDXFORM command returns the matrix elements of the current 2-D transformation. The two translation elements are returned as 16-bit numbers. The four rotation/scaling elements are displayed (and returned in ASCII mode) as:

SIIIII.FFFFFF

where S is the sign of element: space if positive, "-" if negative
I is the integer part of the element
. is a decimal point
F is the fractional part of the element.

If you previously set RDMODE = 1 (see the RDMODE command), then only four bytes are returned (with no decimal point).

ASCII PARAMETERS

type The type of transformation to be returned (8 bits).

0 = XF2D The current 2-D transformation.
1 Reserved.

RDXFORM RDXFORM

FORTRAN SUBROUTINE CALLS AND PARAMETERS

CALL RDXFRM (FUNC, MATRIX, XBASIS, YBASIS, ZBASIS, OFFSET)

Input Parameter:	INTEGER*2	FUNC
Output Parameters:	REAL*4	MATRIX(1), XBASIS(3), YBASIS(3), ZBASIS(3), OFFSET(3)

FUNC The type of transformation to be returned. On the Model One/80, the only transformation type is 2-D (FUNC = 0).

MATRIX A singly subscripted array containing a 2-D absolute transformation matrix converted from 16.16 fixed point to REAL*4.

MATRIX is returned in 3 x 2 form so that
MATRIX(1,1) = x basis vector, x component
MATRIX(1,2) = x basis vector, y component
MATRIX(2,1) = y basis vector, x component
MATRIX(2,2) = y basis vector, y component
MATRIX(3,1) = displacement vector, x component
MATRIX(3,2) = displacement vector, y component

The parameters XBASIS, YBASIS, ZBASIS, and OFFSET do not apply to the Model One/80.

The following subroutine calls are provided for convenience.

CALL RDXFM0 (MATRIX) (Read back 2-D transformation matrix)

CALL RDXF0X (VECTOR) (Read back 2-D transformation matrix into a 6 element vector)

VECTOR is returned so that
VECTOR (1) = x basis vector, x component
VECTOR (2) = x basis vector, y component
VECTOR (3) = y basis vector, x component
VECTOR (4) = y basis vector, y component
VECTOR (5) = displacement vector, x component
VECTOR (6) = displacement vector, y component

EXAMPLE

```
! XFORM2D ABS  0 2. 0 0 2. 0 0
                        ; Define an absolute 2-D transformation.
! RDXFORM 0             ; Read back the current 2-D transformation.
 00002.000000
 00000.000000
 00000.000000
 00002.000000
 00000
 00000
```

READBU READBU

SYNTAX

ASCII	READBU flag, cflag
FORTRAN Call	CALL READBU (IFLAG, ICFLAG, IBUTT, X, Y)
Binary	[154] [flag] [cflag] (3 bytes)
	154 decimal = 232 octal = 9A hex

FUNCTION

The READBU command returns to the port in Graphics mode the function button number of a button that is pressed or released, as well as the location of the XY digitizer at the time the button was pressed or released.

The READBU command removes one entry from the function button queue, which is eight event pairs deep. An event pair consists of a button press and release. If a button is released, the negative of the button number is stored in the button event queue.

The flag parameter indicates whether the Model One should respond immediately, and whether button releases should be recognized as well as button hits. If flag = 0, the Model One responds immediately. If there is a button hit in the queue, the button number is sent along with the coordinates of the XY digitizer. Otherwise, a button number of zero is sent along with the coordinates 0,0. If flag = 1, the Model One waits until there is a button hit entry in the button queue, and then sends the data. If flag = 2 or 3, the READBU command works the same way as with flag = 0 or 1, except that both button hits and releases are recognized.

Note: Button releases are stored in the queue even if you don't want to use them. If a button release is the first item in the queue, this release is removed when you issue the READBU command with flag = 2 or 3. If a button release is the first item in the queue, and you issue the READBU command with flag = 0 or 1, then both the button release and the next item (button press) are removed from the queue.

The cflag parameter indicates the type of coordinate data to be returned. Cflag = 0 indicates to return raw XY digitizer coordinate data in CREG 1. Cflag = 1 indicates to return scaled coordinate data in CREG 2. Cflag = 2 indicates to return coordinates from a large tablet, such as a Summagraphics microgrid.

READBU READBU

FUNCTION, continued

If RDMODE is 0, the function button number and coordinate data are sent in ASCII decimal format. When flag = 0 or 1, the format is FORTRAN I3, 2I6 followed by a carriage return. When flag = 2 or 3, the format is FORTRAN I4, 2I6. If RDMODE is 1, the data is returned as 5 binary bytes, the first for the button number, the next 4 representing the x,y coordinate requested.

For serial communications only, if the READBU command was sent from the host, the host must send an ACK (06 hex or 86 hex) to the Model One to resume normal command interpretation. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

ASCII PARAMETERS

flag Event queue flag.

0 = NOWAIT	Return next button hit immediately, or return button 0 if the event queue is empty.
1 = WAIT	Return the next button hit as soon as there is one, and wait if necessary.
2 = UPNOWAIT	Return next button hit or release immediately, or return button 0 if the event queue is empty.
3 = UPWAIT	Return the next button hit or release as soon as there is one, and wait if necessary.

cflag Coordinate flag.

0 = UNSCALED	Return raw coordinate data in CREG 1.
1 = SCALED	Return scaled coordinate data in CREG 2.
2 = RAW	Return coordinates from large tablet, such as a Summagraphics microgrid.

READBU READBU

FORTRAN PARAMETERS

Input Parameters:	INTEGER*2	IFLAG, ICFLAG
Output Parameters:	INTEGER*2	IBUTT
	INTEGER*4	X, Y

IBUTT returns the number of the button pressed or released. X and Y return the coordinates of the XY digitizer at the time the button was pressed or released.

EXAMPLE

The command sequence in this example is executed after the user has pressed and released button 4 once. The location of the XY digitizer at the time of the press and release was at 10,-15.

```
! READBU 3 0              ; Wait until there is a button hit or release.
                            Return raw coordinate data in CREG 1.
  004 00010 -00015          (Response from Model One.)
! READBU 3 0              ; Reissue previous command.
  -004 00010 -00015         (Response from Model One; the button
                            number is now negative which indicates a
                            button release.)
! READBU 3 0              ; Reissue previous command again. There is no
                            response until another button is pressed.
```

READCR READCR

SYNTAX

ASCII	READCR creg
FORTRAN Call	CALL READCR (ICREG, IX, IY)
Binary	[152] [creg] (2 bytes)
	152 decimal = 230 octal = 98 hex

FUNCTION

The READCR command sends the data in the coordinate register specified by creg to the port in Graphics mode. If the RDMODE is 0, the address is sent as two ASCII decimal numbers representing the x and y coordinates of the address. The numbers are sent in FORTRAN 2I6 format, followed by a carriage return. If the RDMODE is 1, the numbers are sent as 4 binary bytes (2 for x, 2 for y).

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

ASCII PARAMETERS

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

FORTRAN PARAMETERS

Input Parameter:	INTEGER*2	ICREG
Output Parameters:	INTEGER*2	IX, IY

IX and IY are returned from the subroutine with the x and y coordinates contained in the coordinate register specified by ICREG.

EXAMPLE

```
! CLOAD 23 110 200      ; Load CREG 23 with 110,200.
! READCR 23             ; Read contents of CREG 23.
  00110 00200             (Response from Model One.)
```

READER READER

SYNTAX

ASCII	READER
FORTRAN Call	CALL READER (IERROR)
Binary	[56] (1 byte)
	56 decimal = 070 octal = 38 hex

FUNCTION

The READER command can be used to determine whether an error has occurred. The READER command returns a one-byte value giving the code of the first error which occurred since the last READER or COLDstart command. The READER command clears the buffer which stores the error code.

Refer to the Model One Error Messages Reference Guide for a list of error codes.

If no error has occurred, the READER command returns zero.

Note: If the READER command returns "000", then you should assume that no error exists. However, there is an error message #000, "Illegal Call to routine ERROR" (which results from a firmware/hardware error), which is very unlikely to be generated. If an application fails and no other error messages besides "000" are generated, then the "000" may indicate an illegal call to ERROR.

FORTRAN PARAMETER

LOGICAL*1 IERROR

IERROR is an output parameter which contains a one-byte error code.

READF READF

SYNTAX

ASCII	READF func
FORTRAN Call	CALL READF (IFUNC)
Binary	[39] [func] (2 bytes)
	39 decimal = 047 octal = 27 hex

FUNCTION

The READF command controls the format and meaning of the data sent by the Model One when a READW and READWE command is executed. The parameter func specifies the format. The default setting for func is 0. The possible values for func are listed in the table below.

Function	Data	ASCII decimal Format (RDMODE 0)	Binary Format (RDMODE 1)
0	Full 24-bit data	FORTRAN 3I3	3 binary bytes
1	Red component only	FORTRAN I3	1 binary byte
2	Green component only	FORTRAN I3	1 binary byte
3	Blue component only	FORTRAN I3	1 binary byte
4*	Packed r,g,b	FORTRAN I3	1 binary byte

Note: When func = 4, the data packing depends on whether RGBTRU is ON or OFF. With RGBTRU OFF, READF 4 returns only the first byte of the pixel value; the format is the same as READF 1. With RGBTRU ON, READF 4 packs the high 2 bits of each byte (r,g,b) into a single byte. This is the same RGB format used to send data to the Model One in the VAL1K command.

ASCII PARAMETER

func Specifies the format; range is 0 to 4.

FORTRAN PARAMETER

INTEGER*2 IFUNC

READF READF

EXAMPLE: 24-Bit System with RGBTRU ON

```
! RUNLEN  10 20 255,0,255 99 50,50,50 99

                          ; Draw a rectangle 10 rows high and 20 rows wide.
                            The top 5 rows are magenta (255,0,255) and the
                            bottom 5 rows are grey (50,50,50).

! READF 0                 ; Set READWE format for full 24-bit data.
                            (READF 0 is the default.)
! READWE 10 20 1          ; Read a 10 by 20 window in run-length encoded form.
  255 000 255 099
  050 050 050 099           (Response of Model One.)

! READF 1                 ; Set READWE format for 8-bit data (red component
                            only).
! READWE 10 20 1          ; Read the same window.
  255 099
  050 099                   (Response of Model One.)
```

READP READP

SYNTAX

ASCII	READP
FORTRAN Call	CALL READP (IRED, IGRN, IBLU)
Binary	[149] (1 byte)
	149 decimal = 225 octal = 95 hex

FUNCTION

The READP command returns the pixel value of the current point (CREG 0) to the port in Graphics mode. When RDMODE is set to 0, the pixel value is returned as three ASCII decimal numbers representing the red, green, and blue components of the pixel value. The numbers are sent in FORTRAN 3I3 format, followed by a carriage return. If RDMODE is set to 1, the pixel value is returned as three binary bytes.

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

FORTRAN PARAMETERS

INTEGER*2 IRED, IGRN, IBLU

IRED, IGRN, AND IBLU are output parameters which contain the red, blue and green values of the pixel at the current point. The FORTRAN library handles transmission of the ACK character.

EXAMPLE 1: 8-Bit System

```
! LUT8 51 255,0,255      ; Set the color out for LUT index 51 to magenta.
! VAL8 51                ; Set current pixel value to 51 (magenta).
! FLOOD                  ; Flood displayed image memory to magenta.
! READP                  ; Read pixel value at current point.
  051 000 000             (Response from Model One.)
```

EXAMPLE 2: 24-Bit System with RGBTRU ON

```
! VALUE 255 0 255        ; Set current pixel value to r=255, g=0, b=255.
! FLOOD                  ; Flood displayed image memory to magenta.
! READP                  ; Read pixel value at current point.
  255 000 255             (Response from Model One.)
```

READVR READVR

SYNTAX

ASCII	READVR vreg
FORTRAN Call	CALL READVR (IVREG, IRED, IGRN, IBLU)
Binary	[153] (1 byte)
	153 decimal = 231 octal = 99 hex

FUNCTION

The READVR command returns the pixel value in value register vreg to the port in Graphics mode. When RDMODE is set to 0, the pixel value is returned as three ASCII decimal numbers representing the red, green, and blue components of the pixel value. The numbers are sent in FORTRAN 3I3 format and are followed by a carriage return. When RDMODE is set to 1, the pixel value is returned as three binary bytes.

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

ASCII PARAMETER

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.

FORTRAN PARAMETERS

Input parameter:	INTEGER*2	IVREG
Output parameters:	INTEGER*2	IRED, IGRN, IBLU

IRED, IGRN, AND IBLU contain the red, blue and green components of the pixel value contained in the value register specified by IVREG.

The FORTRAN library handles transmission of the ACK character.

EXAMPLE

```
! VLOAD 3 35 0 0          ; Load VREG 3 with 35,0,0.
! READVR 3                ; Read contents of VREG 3.
  035 000 000                (Response from Model One.)
```

READW READW

SYNTAX

ASCII	READW nrows, ncols, bf
FORTRAN Call	CALL READW (NROWS, NCOLS, IRED, IGRN, IBLU)
Binary	[150] [highnrows][lownrows] [highncols][lowncols] [bf] (6 bytes)
	150 decimal = 226 octal = 96 hex

FUNCTION

The READW command instructs the Model One to read back a rectangular array of pixels to the port in Graphics mode. Nrows and ncols specify the number of rows and columns to read. The current point is used as the upper left-hand corner of the window. The window is scanned left to right and top to bottom.

The pixel values are sent to the host in the format set by the READF command. The RDMODE command determines whether the data is returned in binary or ASCII decimal format.

For serial communications, the bf parameter (blocking factor) tells the Model One how many pixel values to send before inserting a carriage return into the output stream. If the end of the window is reached before the block is filled, the block is padded with zeros and sent. After sending each block, the Model One then waits for an ACK (06 hex or 86 hex) character from the host before sending out another block of data. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

Note: The READW command sends data to the host in the same format that the PIXELS and PIXEL8 commands use to send data to the Model One.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns.
bf	8-bit integer specifying the blocking factor.

READW READW

FORTRAN SUBROUTINES AND PARAMETERS

CALL READW (NROWS, NCOLS, IRED, IGRN, IBLU)

Input parameters: INTEGER*2 NROWS, NCOLS

Output parameters: LOGICAL*1 IRED(1), IGRN(1), IBLU(1)

IRED, IGRN, and IBLU are byte (8-bit) arrays which contain the pixel values returned by the subroutine call. The arrays should be declared in the main program to a dimension at least as large as NROWS * NCOLS.

The FORTRAN library handles transmission of the ACK character.

Additional Subroutines

The READW subroutine is a shell which calls a series of lower level routines to do its work. The lower level routines are functionally independent and may be called by themselves. READW calls one of the following routines, depending on the value of the readback format (IFMT) in the RASTEK common block. The readback format is changed by calling the READF subroutine.

CALL READW0 (NROWS, NCOLS, IRED, IGRN, IBLU)	Full 24-bit data (IFMT = 0)
CALL READWR (NROWS, NCOLS, IRED)	Red component only (IFMT = 1)
CALL READWG (NROWS, NCOLS, IGRN)	Green component only (IFMT = 2)
CALL READWB (NROWS, NCOLS, IBLU)	Blue component only (IFMT = 3)
CALL READW4 (NROWS, NCOLS, IVAL)	Packed r,g,b data (IFMT = 4)

Note: For READW4, IVAL is declared LOGICAL*1. The format of the data in IVAL depends on whether RGBTRU is ON or OFF. Refer to the READF command for information on the format.

READWE READWE

SYNTAX

ASCII	READWE nrows, ncols, bf
FORTRAN Call	CALL READWE (NROWS, NCOLS, IDATA, NBYTES)
Binary	[151] [highnrows][lownrows] [highncols][lowncols] [bf] (6 bytes)
	151 decimal = 227 octal = 97 hex

FUNCTION

The READWE command instructs the Model One to read back a rectangular array of pixels to the port in Graphics mode. The Model One sends the data in a run-length encoded format. This format consists of a pixel value followed by a count of the horizontal pixels in a row which are set to that value. The count is set to one less than the number of pixels set to the value.

Nrows and ncols specify the number of rows and columns to read. The current point is used as the upper left corner of the window. The window is scanned left to right and top to bottom.

The pixel values are sent to the host in the format set by the READF command. The RDMODE command determines whether the data is returned in binary or ASCII decimal format.

For serial communications, the bf parameter (blocking factor) tells the Model One how many pixel and count pairs to send before inserting a carriage return into the output stream. If the end of the window is reached before the block is filled, the block is padded with zeros and sent. After sending each block, the Model One then waits for an ACK (06 hex or 86 hex) character from the host before sending out another block of data. The acknowledge character must be sent from the host as a single 7-bit control character, regardless of whether the host normally sends data to the Model One in 8-bit binary or ASCII hex.

Note: The READWE command sends data to the host in the same format that the RUNLEN and RUNLN8 commands use to send data to the Model One.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns.
bf	8-bit integer specifying the blocking factor.

FORTRAN SUBROUTINES AND PARAMETERS

CALL READWE (NROWS, NCOLS, IDATA, NBYTES)

Input parameters:	INTEGER*2	NROWS, NCOLS
Output parameters:	LOGICAL*1	IDATA(1)
	INTEGER*4	NBYTES

IDATA is a byte (8-bit) array which contains the run-length encoded data. The format of the data in IDATA varies depending on the value of the readback format (IFMT) in the RASTEK common block. The readback format is changed by calling the READF subroutine. If IFMT = 0 (indicating full color 24-bit data), then IDATA should be dimensioned in the main program to be at least NROWS * NCOLS * 4. For all other values of IFMT, IDATA should be dimensioned to be at least NROWS * NCOLS * 2.

NBYTES returns the number of elements in IDATA.

Additional Subroutines

READWE is a shell which calls a series of lower level routines to do its work. The lower level routines are functionally independent and may be called by themselves. READWE calls one of the following routines, depending on the value of the readback format (IFMT) in the RASTEK common block. The readback format is changed by calling the READF subroutine.

CALL RDWE0 (NROWS, NCOLS, IDATA, NBYTES)	Full 24-bit data (IFMT = 0)
CALL RDWER (NROWS, NCOLS, IRED, NBYTES)	Red component only (IFMT = 1)
CALL RDWEG (NROWS, NCOLS, IGRN, NBYTES)	Green component only (IFMT = 2)
CALL RDWEB (NROWS, NCOLS, IBLU, NBYTES)	Blue component only (IFMT = 3)
CALL RDWE4 (NROWS, NCOLS, IVAL, NBYTES)	Packed r,g,b data (IFMT = 4)

Note: For RDWE4, IVAL is declared LOGICAL*1. The format of the data in IVAL depends on whether RGBTRU is ON or OFF. Refer to the READF command for information on the format.

EXAMPLE: 8-Bit System

```
! RUNLN8  10 20 3 99 48 99     ; Draw a rectangle.  The top 5 rows are blue
                                  and the bottom 5 rows are red.
! READWE 10 20 1               ; Read a 10 by 20 window in run-length
                                  encoded form.
  003 000 000 099                 (Response from Model One.)
  048 000 000 099

! READF 1                      ; Set READWE format for 8-bit data (red
                                  component only).
! READWE 10 20 1               ; Read same window.
  003 099                         (Response from Model One.)
  048 099
```

RECREL RECREL

SYNTAX

ASCII	RECREL dx, dy	
FORTRAN Call	CALL RECREL (IDX, IDY)	
Binary	[137] [highdx] [lowdx] [highdy] [lowdy]	(5 bytes)
	137 decimal = 211 octal = 89 hex	

FUNCTION

The RECREL command draws a rectangle in image memory with one corner at the WCS current point (CREG 0) and the diagonally opposite corner displaced from the current point by dx,dy. The rectangle is drawn in the current pixel value (VREG 0). The current point is unchanged.

ASCII PARAMETERS

dx, dy	16-bit integers specifying the displacement from the current point of the opposite corner of the rectangle; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2 IDX, IDY

EXAMPLE

```
! MOVABS 100 150      ; Move the current point to 100,150.
! RECREL 10 10        ; Draw rectangle with diagonally opposite corner
                       displaced from current point by 10,10 (at 110,160).
! RECREL -20 -30      ; Draw rectangle with diagonally opposite corner
                       displaced from current point by -20,-30
                       (at 80,120). The two rectangles intersect at
                       the current point, which has not changed.
```

RECTAN RECTAN

SYNTAX

ASCII	RECTAN x,y
FORTRAN Call	CALL RECTAN (IX, IY)
Binary	[142] [highx] [lowx] [highy] [lowy] (5 bytes)
	142 decimal = 216 octal = 8E hex

FUNCTION

The RECTAN command draws a rectangle in image memory with one corner at the WCS current point (CREG 0) and the diagonally opposite corner at the point specified by x,y. The current point is unchanged.

ASCII PARAMETERS

x, y	16-bit integers specifying the coordinates for the diagonally opposite corner of the rectangle; range is from -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2 IX, IY

EXAMPLE

```
! MOVABS 30 50          ; Move current point to 30,50.
! RECTAN 70 100         ; Draw rectangle with opposite corners located at
                          30,50 and 70,100.
! RECTAN -70 -100       ; Draw rectangle with opposite corners located at
                          30,50 and -70 -100. The two rectangles intersect
                          at the current point, which has not changed.
```

RECTI RECTI

SYNTAX

ASCII	RECTI creg
FORTRAN Call	CALL RECTI (ICREG)
Binary	[143] [creg] (2 bytes)
	143 decimal = 217 octal = 8F hex

FUNCTION

The RECTI command draws a rectangle with one corner at the current point (CREG 0) and the diagonally opposite corner at the point specified by coordinate register creg. The current point is unchanged.

ASCII PARAMETERS

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

FORTRAN PARAMETERS

INTEGER*2 ICREG

EXAMPLE

```
! MOVABS 0 0          ; Move current point to 0,0.
! CLOAD 17 50 50      ; Load CREG 17 with 50,50.
! RECTI 17            ; Draw rectangle with opposite corners at
                         0,0 (current point) and 50,50 (CREG 17).

! MOVABS 200,100      ; Move current point to 200,100.
! RECTI 17            ; Draw rectangle with opposite corners at
                         200,100 (current point) and 50,50 (CREG 17).
```

REPLAY REPLAY

SYNTAX

ASCII	REPLAY
FORTRAN Call	CALL REPLAY
Binary	[188] (1 byte)
	188 decimal = 274 octal = BC hex

FUNCTION

The REPLAY command sends a dump of the last 32 characters sent by the host over the HOSTIO interface to the local alphanumeric display screen. The characters are displayed in ASCII hexadecimal format. The last character output is the last character that was sent by the host.

EXAMPLE

```
! REPLAY                    ; Dump last 32 characters of HOSTIO input buffer
                              to ALPHASIO port.

00 FF F0 FD E0 E2 20 40     (Possible response from Model One.)
30 3F E3 20 21 31 00 00
33 53 E5 25 20 32 37 70
7F FF FF FF 30 3F 55 F5
```

RGBTRU RGBTRU

SYNTAX

ASCII	RGBTRU flag
FORTRAN Call	CALL RGBTRU (FLAG)
Binary	[78] [flag] (2 bytes)
	78 decimal = 116 octal = 4E hex

FUNCTION

The RGBTRU command enables and disables RGBTRU mode. In RGBTRU mode, you can use a 24-bit Model One as a true color system, with 8 bits each of red, green, and blue. Flag = 1 or ON enables RGBTRU mode; flag = 0 or OFF disables RGBTRU mode. With RGBTRU ON, the depth buffer is not available.

For RGBTRU mode to work correctly, three consecutive memory units must be available, the first of which is 0, 4, 8, or 12. You use the MEMSEL command to select the first memory unit of the triplet.

In RGBTRU mode, there are three 8-bit-in, 8-bit-out look-up tables. Each look-up table corresponds to a primary color: red, green, and blue. By default, the indices of the look-up tables correspond directly to the color intensity values. For example, the default entry at index 255 of the red LUT is 255.

Once RGBTRU mode is enabled, the value commands work as follows:

VALUE r,g,b Writes r, g, and b values to the first, second, and third memory units of the triplet.

VAL8 val Writes the value val to all three memory units.

VAL1K val Parses the value val as follows: the top two bits are discarded, the next two bits indicate the top two bits for red, the next two bits indicate the top two bits of green, and the low two bits indicate the top two bits of blue. The low 6 bits of each of red, green, and blue are set to zeros. This results in a default value the same as those for VAL1K 0 to 63 for VAL1K with RGBTRU OFF on an 8-bit system with default LUT values.

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

RGBTRU RGBTRU

ASCII PARAMETER

flag Flag = 1 or ON, enable RGBTRU mode; flag = 0 or OFF, disable RGBTRU mode.

FORTRAN PARAMETER

INTEGER*2 FLAG

RMSK16 RMSK16

SYNTAX

ASCII	RMSK16 mask	
FORTRAN Calls	CALL RMSK16 (MASK) CALL RMSK16S (MASK)	Swaps high and low bytes
Binary	[67] [highmask] [lowmask]	(3 bytes)
	67 decimal = 103 octal = 43 hex	

FUNCTION

The RMSK16 command specifies a 16-bit read mask for the Model One's image memory planes. Only the high byte of the read mask is used. The high eight bits correspond to the eight bit planes of image memory. The low byte of the mask is ignored. The read mask is ANDed with the data from image memory immediately before the data enters the look-up table. If a bit is set (1), the corresponding bit plane is read-enabled; if the bit is cleared (0), the corresponding bit plane is not displayed.

When used in conjunction with the WMSK16 command (write mask), the RMSK16 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 read and write masks can also be used to store multiple images in image memory and select them for display.

Note: The RMSK16 command is intended to be used with an 8-bit system. Use the RDMASK command with a 24-bit system in true color mode.

ASCII PARAMETER

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

RMSK16 RMSK16

FORTRAN PARAMETER

CALL RMSK16 (MASK)
CALL RMSK16S (MASK)

INTEGER*2 MASK

The RMSK16 subroutine uses only the high 8 bits of MASK.

The RMSK16S 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: 8-Bit System

```
! 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 1 (255,0,0: 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 table index 16 to
                          green.
! VAL8 16               ; Set current pixel value to 16 (0,255,0: 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.
```

RUNLEN RUNLEN

SYNTAX

ASCII	RUNLEN nrows, ncols, r, g, b, cnt, ...
FORTRAN Call	CALL RUNLEN (NROWS, NCOLS, IDATA)
Binary	[42] [highnrows][lownrows] [highncols][lowncols] ([r] [g] [b] [cnt]) ... (5 + 4 * number of runs bytes) 42 decimal = 052 octal = 2A hex

FUNCTION

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

Each pixel value is followed by a one byte count parameter, cnt, which specifies the number of horizontally contiguous pixels which are to be set to the given r,g,b value. The count is one less than the number of pixels to be set. If cnt = 0, one pixel is set, if cnt = 1, two pixels are set; the range is up to cnt = 255, where 256 pixels are set.

For an 8-bit system, only the red byte is used to determine the location in the look-up table. For this reason, the RUNLEN command is very inefficient and you should only use it if you require compatibility with other Model Ones. Otherwise, use the RUNLN8 command.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is 0 to 32,767.
r, g, b	8-bit values specifying the red, green, and blue components of the pixel value.
cnt	8-bit value specifying the number of horizontally contiguous pixels to be set to the r,g,b value; range is 0 to 255.

FORTRAN PARAMETERS

```
INTEGER*2     NROWS, NCOLS
LOCICAL*1     IDATA(1)
```

IDATA is a byte (8-bit) array which contains the run-length encoded data. The array must contain enough data to fill a window NROWS by NCOLS.

EXAMPLE: 24-Bit System with RGBTRU ON

```
! MOVABS 0 0                ; Move the current point to 0,0.

! RUNLEN 10 20 255,0,255 99 50,50,50 99

                            ; Draw a rectangle 10 rows high and 20 rows wide.
                              The top 5 rows are magenta (255,0,255) and the
                              bottom 5 rows are grey (50,50,50).
```

RUNLN8 RUNLN8

SYNTAX

ASCII	RUNLN8 nrows, ncols, val, cnt, ...
FORTRAN Call	CALL RUNLN8 (NROWS, NCOLS, IDATA)
Binary	[43] [highnrows][lownrows] [highncols][lowncols] ([val] [cnt]) ... (5 + 2 * number of runs bytes)
	43 decimal = 053 octal = 2B hex

FUNCTION

The RUNLN8 command transmits a run-length encoded image to the Model One. The array of transmitted data is nrows high and ncols wide. The location of the upper left corner of the array is defined by the current point (CREG 0). Pixels in image memory are filled left to right and top to bottom. Each pixel value is sent as an 8-bit quantity which is used as an index into the look-up table.

Each pixel value is followed by a one byte count parameter, cnt, which specifies the number of horizontally contiguous pixels which are to be set to the value given by val. The count should be one less than the number of pixels to be set. If cnt = 0, one pixel is set, if cnt = 1, two pixels are set; the range is up to cnt = 255, where 256 pixels are set.

The RUNLN8 command is intended for use with an 8-bit system. If you are using a 24-bit system in full color mode, you can use the RUNLEN command.

ASCII PARAMETERS

nrows, ncols	16-bit integers specifying the number of rows and columns in the array of data; range is 0 to 32,767.
val	8-bit parameter specifying the pixel value.
cnt	8-bit value specifying the number of horizontally contiguous pixels to be set to the pixel value; range is 0 to 255.

FORTRAN PARAMETERS

```
INTEGER*2      NROWS, NCOLS
LOCICAL*1      IDATA(1)
```

IDATA is a byte (8-bit) array which contains the run-length encoded data. The array must contain enough data to fill a window NROWS by NCOLS.

EXAMPLE

```
! MOVABS 0 0              ; Move the current point to 0,0.

! RUNLN8 10 20 3,99 48,99

                          ; Draw a rectangle 10 rows high and 20 rows wide.
                            The top 5 rows are blue (LUT index 3) and the
                            bottom 5 rows are red (LUT index 48).
```

SEGDEF SEGDEF

SYNTAX

ASCII	SEGDEF segment
FORTRAN Call	CALL SEGDEF (SEGID)
Binary	[220] [segment3][segment2][segment1][segment0] (5 bytes)
	220 decimal = 334 octal = DC hex

FUNCTION

The SEGDEF command opens a segment definition. Segment, a 32-bit number, specifies the new segment number. Segments can contain a variety of different commands, including

- commands that draw 2-D graphics primitives
- commands that move the current point
- commands that change the current color
- commands that change primitive-generation attributes, such as VECPAT and PRMFIL
- the SEGREF command to nest (reference) child segments, and
- the PICKID command.

Do not put readback commands (e.g., READBU, READP, etc.) in segments. Readback commands in a display list segment will hang the system when the segment is referenced.

You must close every segment definition with the SEGEND command. By default, segments are visible, pickable, and executable upon creation. If you want to change any of these attributes, use the SETATR command. To draw and pick segments after defining them, use the SEGREF command.

ASCII PARAMETER

segment The segment number; range is 00000000 to FBFFFFFF hex (32 bits). Segment numbers in the range FC000000 to FFFFFFFF (hex) are reserved.

FORTRAN PARAMETER

INTEGER*4 SEGID

SAVCFG SAVCFG

ASCII SYNTAX

SAVCFG

FUNCTION

The SAVCFG command saves the Model One port configurations defined with the SYSCFG command. SAVCFG saves all port configurations; any port configurations that were not changed by SYSCFG, however, are not changed by SAVCFG. In addition, SAVCFG saves definitions of special characters, and the current settings of the ALPHEM, GENLOCK, RGBTRU, and VIDFORM commands.

The SAVCFG command stores the current configuration in NVRAM (non-volatile RAM). The configuration saved with the SAVCFG command will be in effect when

- the system is powered on or off
- the RESET button is pressed, or
- the COLD (COLDstart) command is executed.

You can use the DFTCFG command to restore all ports to the default configuration, stored in PROM (programmable read-only memory).

To display the current configuration, use the DISCFG command.

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

EXAMPLE

```
! SYSCFG SERIAL HOSTSO PARITY N      ; Configures the serial port HOSTIO.

! SAVCFG                             ; Saves the configuration defined
                                      by the SYSCFG command.
```

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

SCROLL SCROLL

SYNTAX

ASCII	SCROLL flag
FORTRAN Call	CALL SCROLL (FLAG)
Binary	[202] [flag] (2 bytes)
	202 decimal = 312 octal = CA hex

FUNCTION

The SCROLL command controls whether the currently selected alphanumeric window scrolls when the text reaches the bottom of the window.

If flag = 1 or ON (the default), scrolling is done; if flag = 0 or OFF, scrolling is not done.

When scrolling is disabled, characters wrap from the end of the bottom line to the beginning of the top line.

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

ASCII PARAMETER

flag — Eight-bit parameter that specifies whether the window scrolls when the text reaches the bottom of the screen:

flag = 1 or ON (the default), scrolling is done;
flag = 0 or OFF, scrolling is not done.

FORTRAN PARAMETER

```
INTEGER*2     FLAG
```

EXAMPLE

```
! SCROLL ON          ; Enable scrolling of the text window.
```

SCRORG SCRORG

SYNTAX

ASCII	SCRORG x,y	
FORTRAN Call	SCRORG (IX, IY)	
Binary	[54] [highx] [lowx] [highy] [lowy]	(5 bytes)
	54 decimal = 066 octal = 36 hex	

FUNCTION

The SCRORG command sets the screen origin register (CREG 4) to the point specified by x,y. The screen origin specifies the coordinate in image memory that will be displayed at the center of the screen. This command is used to pan the displayed image.

Note: If you are using alphanumeric windows, note that window 0 is scrolled using the screen origin register. Therefore, changes to this register will change the placement of text within alphanumeric window 0.

ASCII PARAMETERS

x, y	16-bit parameters specifying the screen origin coordinate; range is -32,768 to 32,767.

FORTRAN PARAMETERS

INTEGER*2 IX, IY

EXAMPLE

```
! MOVABS 0 0          ; Move current point to 0,0.
! CIRCLE 50           ; Draw a circle in the center of the screen.
! SCRORG 300 300      ; Set screen origin to 300,300. Circle is
                         displayed in lower left corner of screen.
! SCRORG -300 -300    ; Set screen origin to -300,-300. Circle is
                         displayed in upper right corner of screen.
! SCRORG 0 0          ; Restore screen origin to 0,0. Circle is
                         displayed in center of screen.
```

SEGAPP SEGAPP

SYNTAX

ASCII	SEGAPP segment
FORTRAN Call	CALL SEGAPP (SEGID)
Binary	[219] [segment3] [segment2] [segment1] [segment0] (5 bytes)

219 decimal = 333 octal = DB hex

FUNCTION

The SEGAPP command reopens an existing segment definition for modification. The commands you input after SEGAPP are appended to the specified segment definition. After you have input all the commands you wish to add, use the SEGEND command to close the segment.

ASCII PARAMETER

segment	The number of the segment to which you want to append commands (32 bits).

FORTRAN PARAMETER

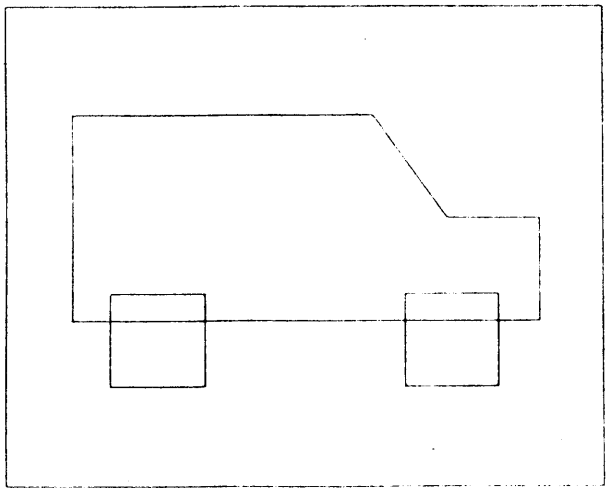
INTEGER*4	SEGID

EXAMPLE

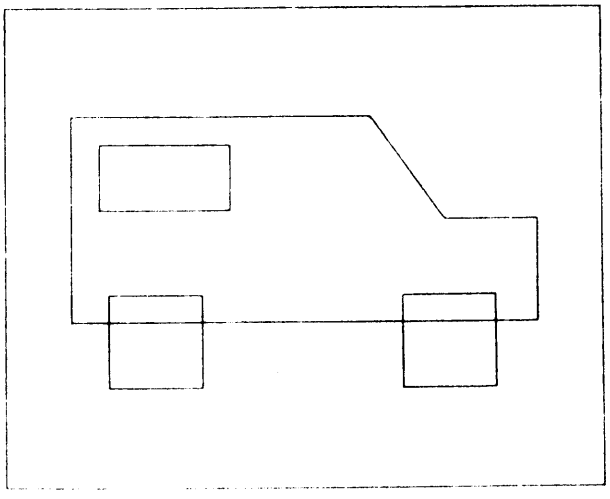
In this example, Segment 1 is reopened and a window is added to the truck. Refer to the example for the SEGDEF command to see the definition of Segment 1.

```
! SEGREF 1              ; Execute (draw) Segment 1, a truck with 2 wheels.
! SEGAPP 1              ; Reopen Segment 1 to append commands.
$ PUSH XFORM XF2DCP     ; Push current 2-D transformation and WCS current
                          point back onto the stack.
$ PICKID 40             ; Assign pick ID 40 to the window.
$ MOVABS -440 80        ; Move current point to lower left corner of
                          window.
$ RECREL 280 140        ; Draw window.
$ POP XFORM XF2DCP      ; Pop current current 2-D transformation and WCS
                          current point from the stack.
$ SEGEND                ; Close Segment 1.
! VAL8 0                ; Clear screen.
! FLOOD
! VAL8 63
! SEGREF 1              ; Execute (draw) appended Segment 1, now a truck
                          with 2 wheels and a window.
```

BEFORE THE APPEND



AFTER THE APPEND



SEGCOP SEGCOP

SYNTAX

ASCII SEGCOP segment2, segment1

FORTRAN Call CALL SEGCOP (SEGDST, SEGSRC)

Binary [231] [segment23] [segment22] [segment21] [segment20]
[segment13] [segment12] [segment11] [segment10]
(9 bytes)

231 decimal = 347 octal = E7 hex

FUNCTION

The SEGCOP command copies segment1 into segment2. If segment2 already exists, it is overwritten. In either case, segment1 remains unchanged.

ASCII PARAMETERS

segment1 The segment number of the source segment (32 bits).

segment2 The segment number of the destination segment (32 bits).

FORTRAN PARAMETERS

INTEGER*4 SEGDST, SEGSRC

EXAMPLE

```
! SEGCOP 3 2          ; Create Segment 3; copy Segment 2 into Segment 3.
```

SEGDEF SEGDEF

SYNTAX

ASCII	SEGDEF segment
FORTRAN Call	CALL SEGDEF (SEGID)
Binary	[220] [segment3] [segment2] [segment1] [segment0] (5 bytes)
	220 decimal = 334 octal = DC hex

FUNCTION

The SEGDEF command opens a segment definition. Segment, a 32-bit number, specifies the new segment number. Segments can contain a variety of different commands, including

- commands that draw 2-D graphics primitives
- commands that move the current point
- commands that change the current color
- commands that change primitive-generation attributes, such as VECPAT and PRMFIL
- the SEGREF command to nest (reference) child segments, and
- the PICKID command.

Do not put readback commands (e.g., READBU, READP, etc.) in segments. Readback commands in a display list segment will hang the system when the segment is referenced.

You must close every segment definition with the SEGEND command. By default, segments are both visible and pickable upon creation. If you want to change either of these attributes, use the SETATR command. To draw and pick segments after defining them, use the SEGREF command.

ASCII PARAMETER

segment The segment number; range is 00000000 to FBFFFFFF hex (32 bits). Segment numbers in the range FC000000 to FFFFFFFF (hex) are reserved.

FORTRAN PARAMETER

INTEGER*4 SEGID