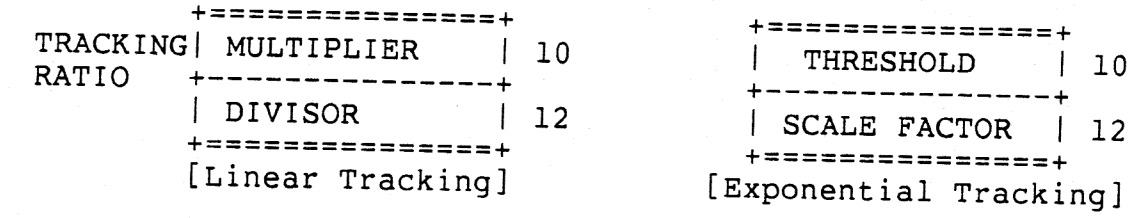
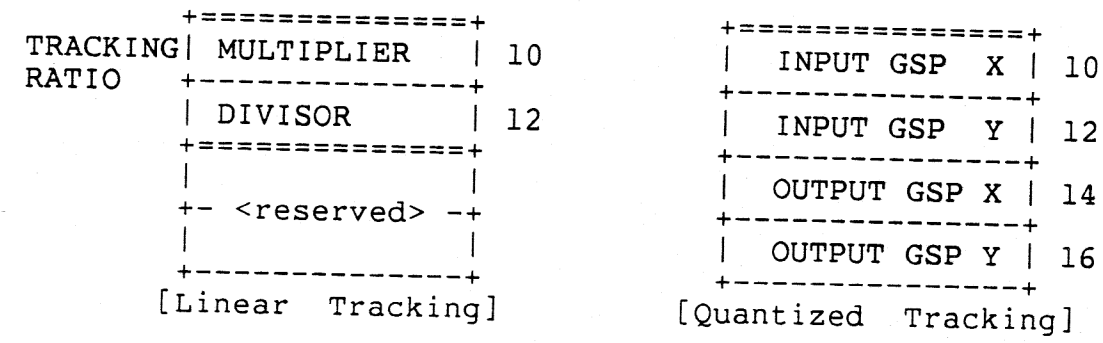
## 6.11 Set Mouse Characteristics Command Packet

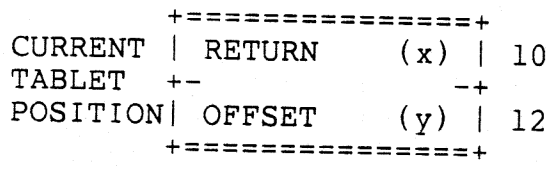



## 6.12 Set Tablet Characteristics Command Packet



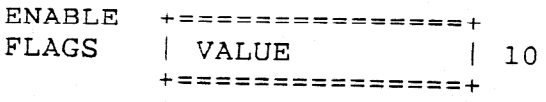

## 6.13 Get Tablet Position Command Packet

This command has a two-word field in which the display returns the current position of the tablet.

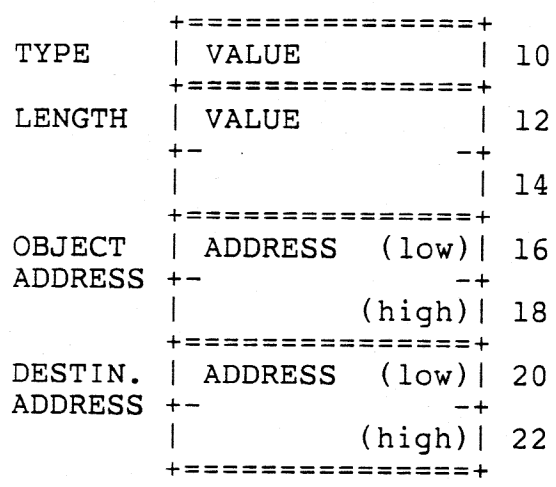



## 6.14 Set Pointing Device Event Reporting

This command enables or disables periodic reporting of mouse and/or tablet movement.

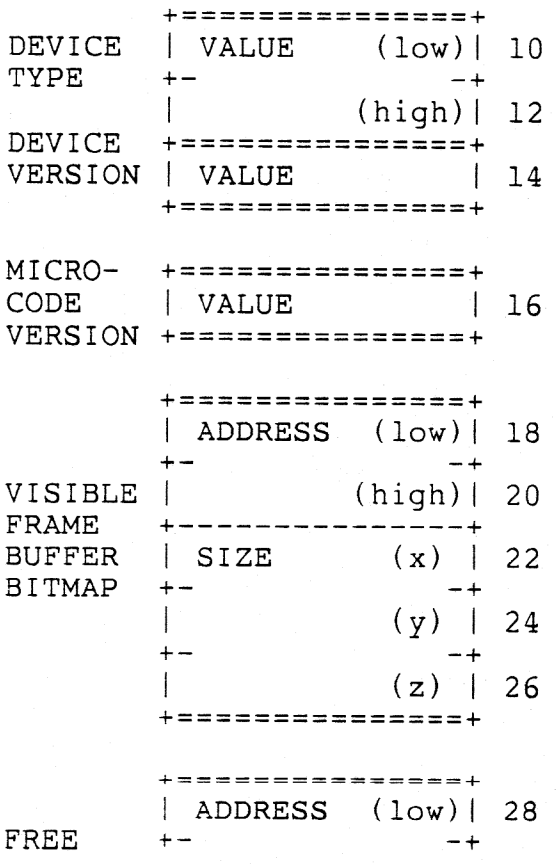


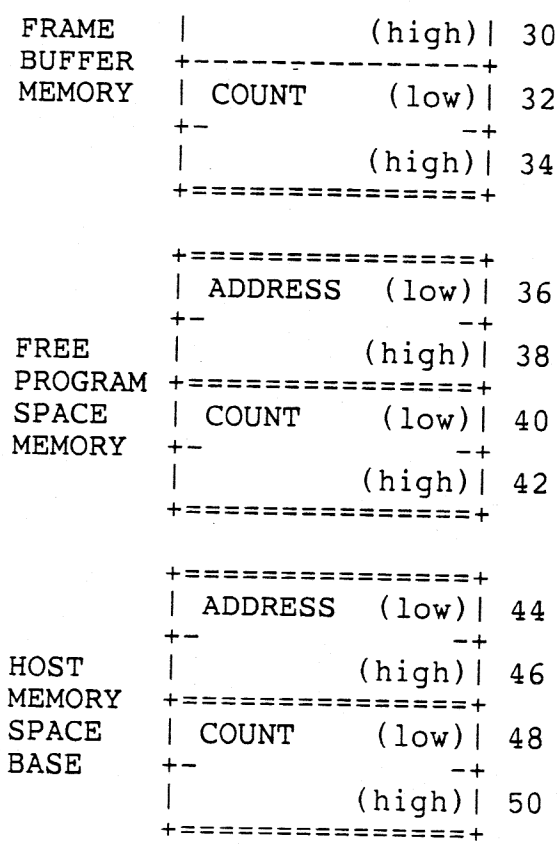
## 6.15 Move Object Command Packet




## 6.16 Report Status Command Packet

For the report status command, the packet contains no input parameters except for the opcode header. However, the packet contains space for the following information that is filled by the display.






## 6.17 No Operation Command Packet

This command has no parameters, and consists only of the packet opcode header.

# 7.0 CONSTANTS, OPCODES, MODIFIERS, AND ERROR CODES

This section specifies the constants used as opcodes, modifiers, and error codes for onyx commands and return codes.

## 7.1 Control And Status Register 0 Function Codes

The following function codes are defined for CSR0 functions.

Implementation-independent functions:

```
INIT = 1
SEND_PACKET = 2
START_DISPLAY = 3
ABORT = 4
EXECUTE_POWERUP_SEQUENCE = 5
```

VS100 Implementation-dependent functions:

```
BBA_ON = 16
BBA_OFF = 17
SET_INFINITE_RETRIES = 18
SET_FINITE_RETRIES = 19
```

## 7.2 Command Packet Operation Codes

The following are the values of the 8-bit operation codes specified in the low byte of the first word transmitted in a device command. These commands are all interperted by display microcode or firmware.

```
NO_OPERATION                = 0
COPY_AREA                   = 1
DRAW_CURVE                  = 2
PRINT_TEXT                  = 3
FLOOD_AREA                  = 4
LOAD_CURSOR                 = 5
SET_CURSOR_POSITION         = 6
ATTACH_CURSOR               = 7
GET_CURSOR_POSITION         = 8
MOVE_OBJECT                 = 9
REPORT_STATUS               = 10
FILL_AREA                   = 11
GET_MOUSE_POSITION          = 12
SET_MOUSE_CHARACTERISTICS   = 13
GET_TABLET_POSITION         = 14
SET_POINTING_DEV_REPORTING  = 15
SET_TABLET_CHARACTERISTICS  = 16
```

The following commands are interpreted only by display ROM, and are thus given different operation codes, even though there is some overlapping of the commands. This prohibits accidental interpretation of a command intended for a different machine state.

MOVE_OBJECT = 128
REPORT_STATUS = 129

## 7.3 Command Packet Operation Modifiers

The modifiers field specifies, for each command, which optional parameters are specified and what format is present for parameters with multiple formats. The following modifiers are defined below, listed separately for each command.

### 7.3.1 Copy Area Command Modifiers -

| PARAMETER | FIELD | VALUE | MEANING |
|---|---|---|---|
| SOURCE | mod<2:0> | 0 | constant |
| | | 1 | bitmap |
| | | 2 | halftone |
| SOURCE MASK | mod<5:3> | 0 | rectangle |
| | | 1 | bitmap |
| DEST. OFF. | mod<8:6> | | (not used) |
| MAP | mod<11:9> | 0 | identity map |
| | | 1 | source map address |
| | | 2 | source map literal |
| | | 3 | function code address |
| | | 4 | function code literal |
| CLIPPING RECTANGLES | mod<14:12> | 0 | none |
| | | 1 | literal rectangle |
| | | 2 | rectangle list addr. |

### 7.3.2 Draw Curve Command Modifiers -

| PARAMETER | FIELD | VALUE | MEANING |
|---|---|---|---|
| SOURCE | mod<2:0> | 0 | constant |
| | | 1 | bitmap |
| | | 2 | halftone |
| SOURCE MASK | mod<5:3> | 0 | rectangle |
| | | 1 | bitmap |
| DEST. OFF. | mod<8:6> | | (not used) |
| MAP | mod<11:9> | 0 | identity map |

| | | | |
|---|---|---|---|
| | | 1 | source map address |
| | | 2 | source map literal |
| | | 3 | function code address |
| | | 4 | function code literal |
| CLIPPING RECTANGLES | mod<14:12> | 0 | none |
| | | 1 | literal rectangle |
| | | 2 | rectangle list addr. |
| DRAWING MODE | mod<15> | 0 | Solid Segment |
| | | 1 | Dashed/Patterned Segment |
| PATTERN STATE | mod<17:16> | 0 | Literal pattern state |
| | | 1 | Indirect pattern state |
| | | 2 | Update literal pattern sta |
| | | 3 | Update indirect pattern st |
| PATTERN MODE | mod<19:18> | 0 | No secondary source (Dashe |
| | | 1 | Constant secondary source |
| | | 2 | Bitmap secondary source |
| | | 3 | Halftone secondary source |

### 7.3.3 Print Text Command Modifiers -

| PARAMETER | FIELD | VALUE | MEANING |
|---|---|---|---|
| SOURCE | mod<2:0> | 0 | constant |
| | | 1 | source font |
| | | 2 | halftone |
| MASK FONT | mod<5:3> | 0 | no mask |
| | | 1 | mask font supplied |
| DEST. OFF. | mod<8:6> | 0 | dest offset literal |
| | | 1 | dest offset indirect |
| | | 2 | update dest literal |
| | | 3 | update dest indirect |
| MAP | mod<11:9> | 0 | identity map |
| | | 1 | source map address |
| | | 2 | source map literal |
| | | 3 | function code address |
| | | 4 | function code literal |
| CLIPPING RECTANGLES | mod<14:12> | 0 | none |
| | | 1 | literal rectangle |
| | | 2 | rectangle list addr. |
| TEXT STRING | mod<15> | 0 | 8 bit characters |
| | | 1 | 16 bit characters |
| CONTROL STRING | mod<16> | 0 | no control string |
| | | 1 | control string |

### 7.3.4 Fill Area Command Modifiers -

| PARAMETER | FIELD | VALUE | MEANING |
|---|---|---|---|
| SOURCE | mod<2:0> | 0 | constant |
| | | 1 | (not used) |
| | | 2 | halftone |
| SOURCE MASK | mod<5:3> | | (not used) |
| DEST. OFF. | mod<8:6> | | (not used) |
| MAP | mod<11:9> | 0 | identity map |
| | | 1 | source map address |
| | | 2 | source map literal |
| | | 3 | function code address |
| | | 4 | function code literal |
| CLIPPING RECTANGLE | mod<14:12> | 0 | none |
| | | 1 | literal rectangle |

### 7.3.5 Flood Area Command Modifiers -

| PARAMETER | FIELD | VALUE | MEANING |
|---|---|---|---|
| SOURCE | mod<2:0> | 0 | constant |
| | | 1 | (not used) |
| | | 2 | halftone |
| SOURCE MASK | mod<5:3> | | (not used) |
| DEST. OFF. | mod<8:6> | | (not used) |
| MAP | mod<11:9> | | (not used) |
| CLIPPING RECTANGLE | mod<14:12> | 0 | none |
| | | 1 | literal rectangle |
| BOUNDARY MAP | mod<15> | 0 | literal |
| | | 1 | pointer |

### 7.3.6 Load Cursor Command Modifiers -

| PARAMETER | FIELD | VALUE | MEANING |
|---|---|---|---|
| SOURCE | mod<2:0> | 0 | constant |
| | | 1 | bitmap |
| | | 2 | halftone |

| | | | |
|---|---|---|---|
| SOURCE MASK | mod<5:3> | 0<br>1 | rectangle<br>bitmap |
| DEST. OFF. | mod<8:6> | | (not used) |
| MAP | mod<11:9> | 0<br>1<br>2<br>3<br>4 | identity map<br>source map address<br>source map literal<br>function code address<br>function code literal |

### 7.3.7 Set Mouse Characteristics Command Modifiers -

| PARAMETER | FIELD | VALUE | MEANING |
|---|---|---|---|
| TRACKING | mod<2:0> | 0<br>1 | Linear<br>Exponential |

### 7.3.8 Set Tablet Characteristics Command Modifiers -

| PARAMETER | FIELD | VALUE | MEANING |
|---|---|---|---|
| TRACKING | mod<2:0> | 0<br>1 | Linear<br>Quantized |

## 7.4 Interrupt Reason Values

The following are the values returned by the display in the Interrupt Reason Register (CSR1) following display interrupt. Interrupt reasons (bit 15 = 0) are unary encoded. Errors (bit 15 = 1) are binary encoded.

```
*
* WGA Completion Codes (15 reserved)
*

INT_ID          equ     $0001           ; Initialisation Done
INT_CD          equ     $0002           ; Command Done
INT_SE          equ     $0004           ; Started Executing
INT_BE          equ     $0008           ; Button Event
INT_CM          equ     $0010           ; Cursor Moved
INT_TM          equ     $0020           ; Tablet Moved
INT_MM          equ     $0040           ; Mouse Moved
INT_PD          equ     $0080           ; Powerup Done

* These are the error messages generated by the
* VS100, VS125 and VS300. Some errors are specific to
* one device. The following key indicates the error code's
* status for each device:
*
*       *   IS generated by this device
*       -   IS NOT generated by this device
*       +   New, was not defined in previous versions
*       !   Changed, Redefined from previous versions
*
* The key contains two characters. The first indicates the status
* for the VS100/125, and the second indicates the status for the VS300.
* For example:
*
*       *-  Error code is generated by VS100/125 only
*       -*  Error code is generated by VS300 only
*       **  Both VS100/125 and VS300 generate this code
*       !-  VS100/125 only, code has new definition
*       -+  VS300 only, new error code added
*       ++  VS100/125 and VS300, new error code added


* WGA Hardware Error Codes (32 reserved)
*
* Error Mnemonic        Value           Key  Description

NO_ERROR        equ     0               ; -+  Normal Successful Completion
ERR_BASE        equ     $8000           ; **  Error-Encountered bit
ERR_NYI         equ     ERR_BASE+0      ; **  Not Yet Implemented
ERR_IFC         equ     ERR_BASE+1      ; **  Invalid Function Code
ERR_ICC         equ     ERR_BASE+2      ; **  Invalid Command Code
ERR_RN          equ     ERR_BASE+3      ; **  Bus Error: Non-Existant Memo
ERR_RO          equ     ERR_BASE+4      ; *-  Bus Error: Retry Overflow
ERR_LD          equ     ERR_BASE+5      ; *-  Bus Error: Link Down
ERR_BE          equ     ERR_BASE+6      ; *-  Bus Error: Unexplained
```

```
ERR_AE          equ     ERR_BASE+7      ; **  Address Error
ERR_SI          equ     ERR_BASE+8      ; **  Spurious Interrupt
ERR_II          equ     ERR_BASE+9      ; *-  Illegal Instruction
ERR_BN          equ     ERR_BASE+10     ; *-  BBA NXM (Non-Existant Memory
ERR_BNI         equ     ERR_BASE+11     ; *-  BBA Not Installed
ERR_KBO         equ     ERR_BASE+12     ; *-  Keyboard Buffer Overflow
ERR_TBO         equ     ERR_BASE+13     ; *-  Tablet Buffer Overflow
ERR_BBO         equ     ERR_BASE+14     ; *-  Button Buffer Overflow
ERR_ITP         equ     ERR_BASE+15     ; *-  Invalid Tablet Packet


* WGA Packet Error Codes (32 reserved)
*
* Error Mnemonic        Value           Key  Description

ERR_ISRCMB      equ     ERR_BASE+32     ; **  Invalid SRC Modifier Bits
ERR_ISRCBW      equ     ERR_BASE+33     ; **  Invalid SRC Bitmap Width
ERR_ISRCBH      equ     ERR_BASE+34     ; **  Invalid SRC Bitmap Height
ERR_ISRCC       equ     ERR_BASE+35     ; *-  Invalid SRC Constant
ERR_ISRCBD      equ     ERR_BASE+36     ; **  Invalid SRC Bitmap Depth
ERR_ISRCD       equ     ERR_BASE+37     ; -+  Invalid SRC Bitmap Dimension

ERR_IMSKMB      equ     ERR_BASE+38     ; **  Invalid MSK Modifier Bits
ERR_IMSKBW      equ     ERR_BASE+39     ; **  Invalid MSK Bitmap Width
ERR_IMSKBH      equ     ERR_BASE+40     ; **  Invalid MSK Bitmap Height
ERR_IMSKBD      equ     ERR_BASE+41     ; **  Invalid MSK Bitmap Depth

ERR_IDSTMB      equ     ERR_BASE+44     ; **  Invalid DST-Offset Modifier

ERR_IDSTBW      equ     ERR_BASE+45     ; **  Invalid DST Bitmap Width
ERR_IDSTBH      equ     ERR_BASE+46     ; **  Invalid DST Bitmap Height
ERR_IDSTBD      equ     ERR_BASE+47     ; **  Invalid DST Bitmap Depth

ERR_NOAREA      equ     ERR_BASE+48     ; -+  No Resultant Area

ERR_IMAPMB      equ     ERR_BASE+50     ; **  Invalid Map Modifier Bits
ERR_IMAPFC      equ     ERR_BASE+51     ; -+  Invalid Map Function Code
ERR_ZIMAP       equ     ERR_BASE+52     ; -+  Depth Incompatible with Map
ERR_ZCIMAP      equ     ERR_BASE+53     ; -+  Depth Combination Incompatib

ERR_ICLPMB      equ     ERR_BASE+54     ; **  Invalid ClipR Modifier Bits
ERR_ICLPRC      equ     ERR_BASE+55     ; **  Invalid ClipR Count

ERR_SMC_ITC     equ     ERR_BASE+56     ; +-  Invalid Tracking Ratio
ERR_ITC_MULT    equ     ERR_BASE+57     ; -!  Invalid Tracking Multiplier
ERR_ITC_DIV     equ     ERR_BASE+58     ; -+  Invalid Tracking Divisor

ERR_ICD         equ     ERR_BASE+59     ; **  Invalid Cursor Device
ERR_MO_IBC      equ     ERR_BASE+60     ; **  Invalid Byte Count
ERR_MO_IOT      equ     ERR_BASE+61     ; **  Invalid Object Type
ERR_MO_IDT      equ     ERR_BASE+62     ; **  Invalid Device Type
ERR_IPC         equ     ERR_BASE+63     ; **  Invalid Path Count

*
* WGA Draw_Curve Error Codes (16 reserved)
```

```
*
* Error Mnemonic          Value              Key  Description

ERR_DC_IPC       equ     ERR_BASE+64      ; **  Invalid Path Count
ERR_DC_IPSL      equ     ERR_BASE+65      ; **  Invalid Pattern Length
ERR_DC_IPSM      equ     ERR_BASE+66      ; **  Invalid Pattern Multiplier
ERR_DC_ICF       equ     ERR_BASE+67      ; **  Invalid Closed Figure
ERR_DC_IPSP      equ     ERR_BASE+68      ; **  Invalid Pattern Position
ERR_DC_IPSMB     equ     ERR_BASE+69      ; **  Invalid Pattern String Modif
ERR_DC_IPMMB     equ     ERR_BASE+70      ; **  Invalid Pattern Mode Modifie
ERR_DC_IPSC      equ     ERR_BASE+71      ; **  Invalid Pattern Count
ERR_DC_ISSRCBW   equ     ERR_BASE+72      ; **  Invalid Second SRC Bitmap Wi
ERR_DC_ISSRCBH   equ     ERR_BASE+73      ; **  Invalid Second SRC Bitmap He
ERR_DC_ISSRCBD   equ     ERR_BASE+74      ; **  Invalid Second SRC Bitmap De
ERR_DC_ISSRCC    equ     ERR_BASE+75      ; *-  Invalid Second SRC Constant
ERR_DC_IDPM      equ     ERR_BASE+76      ; ++  Incompatible Drawing/Pattern

*
* WGA Print_Text Error Codes (16 reserved)
*
* Error Mnemonic          Value              Key  Description

ERR_PT_ICSL      equ     ERR_BASE+80      ; **  Invalid Control String Lengt
ERR_PT_ICSO      equ     ERR_BASE+81      ; **  Invalid Control String Opcod
ERR_PT_ICSP      equ     ERR_BASE+82      ; **  Invalid Control String Param
ERR_PT_ITSL      equ     ERR_BASE+83      ; **  Invalid Text String Length
ERR_PT_ICI       equ     ERR_BASE+84      ; **  Invalid Character Index
ERR_PT_TSE       equ     ERR_BASE+85      ; **  Text String Exhausted
ERR_PT_NFP       equ     ERR_BASE+86      ; **  No Font Present
ERR_PT_ISRCFW    equ     ERR_BASE+87      ; **  Invalid SRC Font width
ERR_PT_ISRCFH    equ     ERR_BASE+88      ; **  Invalid SRC Font height
ERR_PT_ISRCFD    equ     ERR_BASE+89      ; **  Invalid SRC Font depth
ERR_PT_IMSKFW    equ     ERR_BASE+90      ; **  Invalid MSK Font width
ERR_PT_IMSKFH    equ     ERR_BASE+91      ; **  Invalid MSK Font height
ERR_PT_IMSKFD    equ     ERR_BASE+92      ; **  Invalid MSK Font depth
ERR_PT_CSMF      equ     ERR_BASE+93      ; +*  Conflicting SRC/MSK Fonts

*
* WGA Flood_Area Error Codes (16 reserved)
*
* Error Mnemonic          Value              Key  Description

ERR_FA_ISRCB     equ     ERR_BASE+96      ; **  Invalid SRC Bitmap
ERR_FA_SPIOB     equ     ERR_BASE+98      ; **  Seed Point is on boundary
ERR_FA_SO        equ     ERR_BASE+99      ; **  Stack Overflow
ERR_FA_IBMMB     equ     ERR_BASE+100     ; **  Invalid Boundary Map Modifie

*
* WGA Fill_Polygon Error Codes (16 reserved)
*
* Error Mnemonic          Value              Key  Description

ERR_FP_ISRCB     equ     ERR_BASE+112     ; **  Invalid SRC Bitmap
ERR_FP_SO        equ     ERR_BASE+113     ; **  Stack Overflow
ERR_FP_IPC       equ     ERR_BASE+114     ; **  Invalid Point Count
```

```
ERR_FP_ICF      equ     ERR_BASE+115    ; **  Invalid Closed Figure
*
* WGA Powerup Error Codes (32 reserved)
*
* Error Mnemonic        Value             Key  Description

ERR_PASS        equ     ERR_BASE+128    ; *-  Base for test numbers
ERR_68K         equ     ERR_BASE+129    ; *-  68000 CPU
ERR_RC          equ     ERR_BASE+130    ; *-  ROM Checksum
ERR_PR          equ     ERR_BASE+131    ; *-  Program RAM
ERR_CRT         equ     ERR_BASE+132    ; *-  CRTC Register
ERR_TU          equ     ERR_BASE+133    ; *-  Tablet USART
ERR_KU          equ     ERR_BASE+134    ; *-  Keyboard USART
ERR_FOE         equ     ERR_BASE+135    ; *-  FOTR Electrical Loop Back
ERR_VTO         equ     ERR_BASE+136    ; *-  Vsync Time Out
ERR_SB          equ     ERR_BASE+137    ; *-  Screen Buffer
ERR_BS          equ     ERR_BASE+138    ; *-  BBA Scratchpad RAM
ERR_BC          equ     ERR_BASE+139    ; *-  BBA Copyarea Command
ERR_TTO         equ     ERR_BASE+140    ; *-  Tablet Time Out
ERR_FOO         equ     ERR_BASE+141    ; *-  FOTR Optical Loop Back
ERR_KTO         equ     ERR_BASE+142    ; *-  Keyboard Time Out
ERR_KST         equ     ERR_BASE+143    ; *-  Keyboard Self-Test

*
* WGA Load Cursor Error Codes (16 reserved)
*
* Error Mnemonic        Value             Key  Description

ERR_LDC_IATRV   equ     ERR_BASE+160    ; ++  Invalid Cursor Attribute Val
ERR_LDC_ICH     equ     ERR_BASE+161    ; ++  Invalid Cursor Height
ERR_LDC_ICW     equ     ERR_BASE+162    ; ++  Invalid Cursor Width
ERR_NOVALCUR    equ     ERR_BASE+163    ; ++  No Valid Cursor Defined
```

# 8.0 VAXSTATION 100 RESTRICTIONS

The following architectural restrictions apply to the VAXstation 100 implementation of the Workstation graphics architecture.

## 8.1 Number Of Planes

Since the VS100 has only one bit plane of framebuffer memory, the Z parameter may only have the value 1.

## 8.2 Halftone Representation

The VAXstation 100 uses only a single format for a halftone bitmap. The halftone pattern must be specified as a square bitmap 16 pixels on a side. Therefore, to use a "standard" four-by-four halftone pattern, the pattern is simply replicated horizontally and vertically to form a 16x16 pattern.

# 9.0 GENERAL IMPLEMENTATION RESTRICTIONS

The following restrictions apply to all VAX/UNIBUS implementations of the Workstation graphics architecture. Hence, the term "display" is used instead of "VS100".

## 9.1 Word Access I/O

All 16-bit word parameters must be word-aligned. Additionally, the display may only access host memory by words. Any byte strings of odd length, then, should be padded with an extra byte so that no "undefined" data is accessed.

## 9.2 UNIBUS Window Mapping

It is often the case that the display is asked to perform an operation between a source rectangle and a destination rectangle which overlap. That is, both source and destination bitmaps occupy the same (or nearly the same) area of memory. In this case, the display firmware must determine the proper memory-copy direction, so that no data is overwritten. It does this by comparing the base addresses of the two memory blocks in question. For this reason, it is important that the driver software (responsible for mapping UNIBUS memory to the display) maintain a one-to-one correspondence between areas of memory on each side of the UBW. Specifically, an area of VAX memory must never be made to appear to the display as two different areas of memory.

## 9.3 Bitmap Storage Requirements

The display represents a bitmap as a sequence of horizontal scan lines stored in contiguous memory locations. Each scan line must begin on a 16-bit word boundary. That is, although a bitmap can have any horizontal width in pixels, the storage in which the bitmap is kept must have sufficient space so that each horizontal line can be word-aligned. If the horizontal width in pixels is not evenly divisible by 16, the last bits in the last word of the storage for each horizontal line will be unused. For any bitmap of dimensions (X,Y) on the display, the storage requirement is ((X+15)/16)Y words.

## 9.4 Device Coordinate Management For WGA

The Device Position Registers, defined by the WGA, are used to hold the current XY position of the cursor. They are also used to report the current XY position of any pointing devices, e.g. mouse and/or tablet, which may be attached to the VAXstation.

The WGA also defines a phenomenon known as Event Reporting. If reporting is enabled for a particular device, it will interrupt the host at a maximum rate of 60 Hz. It places its XY position in the Cursor Position Registers and issues a "'device' moved" interrupt reason.

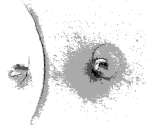
The following paragraphs enumerate the possible Event-Reporting situations, and how they are implemented.

[1] State: No Device Attached to the Cursor

[1a] Mouse Event Reporting set

On the Vsync interrupt, a service routine within the VS100 will read the current mouse XY position. If it has changed from the last time, it will be placed in the Device Position Registers. An interrupt is then sent to the host indicating "MOUSE MOVED". If the mouse has not moved, nothing happens. Note that since the mouse is not connected to anything here, its position is not confined to the visible screen. Its coordinates may vary from 32767 to -32768.

[1b] Tablet Event Reporting set



On the Vsync interrupt, a service routine will read the current tablet XY position. If it has changed from the last time, it will be placed in the Device Position Registers. It then issues an interrupt to the host indicating "TABLET MOVED". If the tablet has not moved, nothing happens. Note that the tablet position is not clipped to the visible screen boundaries unless it's attached to the cursor.

[1c] Both Mouse and Tablet Event Reporting set

In this case, both case [1a] and [1b] happen concurrently. However, only one device will be reported on any one Vsync. The devices will be polled in a round-robin-like fashion. On one Vsync, device (i) will be checked FIRST. If no action, device (i+1) is checked, and so on, until an 'active' one is found (or until there are no more devices). On the next Vsync, device (i+1) will be the first one checked, and so on.

This method has the following properties:

- o No more than 60 interrupts per second are sent to no matter how many devices are reporting.
- o No device will be 'locked out' by a higher-priorit or a more-rapidly-interrupting device.
- o Each individual device may still interrupt at 60Hz it has no 'competition'.
- o The worst case for device acknowledgement is 60/N N reporting devices.

[2] State: Mouse is Attached to the Cursor

[2a] Mouse Event Reporting set

All happens as in [1a], except that now the interrupt reason will be "CURSOR MOVED" instead of "MOUSE MOVED". Of course, since the mouse and cursor are attached, their XY positions will

always be the same.

This implies, of course, that the mouse XY position will now be clipped to the visible screen.

[2b] Tablet Event Reporting set

All happens as in case [1c], except that cases [2a] and [1b] are happening concurrently, instead of cases [1a] and [1b].

[2c] Both Mouse and Tablet Event Reporting set

This is the same as case [2b] above.

[3] State: Tablet is Attached to the Cursor

[3a] Mouse Event Reporting set

All happens as in case [1c], except that cases [1a] and [3b] are happening concurrently, instead of cases [1a] and [1b].

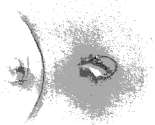
[3b] Tablet Event Reporting set

All happens as in [1b], except that now the interrupt reason will be "CURSOR MOVED" instead of "TABLET MOVED".

Since the tablet and cursor are attached, their XY positions will always be the same as long as they are within the visible screen. Should the tablet position stray from the visible screen, both the tablet and the cursor XY position will be clipped to the screen's boundaries.

[3c] Both Mouse and Tablet Event Reporting set

This is the same as case [3a] above.



## 9.5 Keyboard Interface

If the input data is a keycode, a transition bit is generated, the keyboard device code added, and the result is returned, where it will become a button event to the VAXen host. If the key is in Autorepeating/Down-Only mode, no up-transition events will be generated. An autorepeating key will thus appear as a succession of down-events. If the key is in Up/Down mode, both up and down transitions will be generated.

If the input data is an error message, or an acknowledgement, it is ignored by the VS100.

There are a few codes which are treated specially:

- The Metronome character, for example, is not passed to the host, but is used to repeat the most recently depressed character.
- The state of the control-key code is monitored. When the 'control' key is depressed, the firmware will automaticall issue a 'temporary autorepeat inhibit' to the keyboard to keep control keys from autorepeating.
- The 'Allups' keycode will generate up-transitions for all up/down-mode keys which are down at the time.
- Other special codes not related to actual keys will be discard

  Since the user is not allowed to change the mode of the keyboard divisions, Prefix-To-Keys-Down should never occur.

The keyboard divisions are initialized
According to the following defaults:

| Division | Mode |
|---|---|
| 1. Main array | Autorepeat down |
| 2. Numeric keypad | Autorepeat down |
| 3. Delete character | Autorepeat down |
| 4. Return and Tab characters | Down only |
| 5. Lock, Compose, and A10 | Up down |
| 6. Shift keys and Control key | Up down |
| 7. Horizontal arrow keys | Autorepeat down |
| 8. Vertical arrow keys | Autorepeat down |
| 9. Edit keypad keys | Autorepeat down |
| 10. Local function keys (G99-G04) | Autorepeat down |
| 11. Second function key set (G05-G09) | Autorepeat down |
| 12. Third function key set (G10-G14) | Autorepeat down |
| 13. HELP and MENU keys (G15-G16) | Down only |
| 14. Fifth function key set (G20-G23) | Autorepeat down |

In Up/Down mode, both up- and down-transitions are reported.
In Autorepeat and Down-Only mode, only down-transitions are reported.