5.0 CONTROL AND STATUS REGISTERS

The workstation graphics display communicates with the host via an interface that utilizes a set of control and status registers (CSRs). These CSRs are implemented as shared memory and not as hardware registers. Therefore, a strict protocol must be obeyed to avoid race conditions when using the CSRs. The CSRs and their programming are described in the following sections.

5.1 Control And Status Register (CSR0)

The control and status register is the main control register for transmitting commands to the display processor. Its format is:

```
                 15                      6  5      1  0
                +---------------------------------------+
CSR0            |      Reserved          |IE|FUNCTION|GO|
                +---------------------------------------+
                       Control and Status Register
```

where the fields have the following uses:

GO = CSR0<0> The GO bit is set by the host to indicate that the next command has been set up and the device should process the command.

FUNCTION = CSR0<5:1> The function field is set by the host to indicate the display command to execute, i.e., this is the command opcode. Functions 0 through 15 are implementation-independent definitions and 16 through 31 are implementation-dependent.

IE = CSR0<6> The interrupt enable bit, when set, allows the display to interrupt the host when a command is completed, a mouse event occurs, a keyboard event occurs, or an error occurs. There is only one interrupt vector; the host determines the event by scanning the interrupt reason flags. (Note: when the host initializes or reboots, it should clear IE until it is capable of receiving display interrupts.)

Use of CSR0 must follow a strict protocol. When GO is set to zero, the device is ready for a new command. Only the host can modify FUNCTION and GO when GO is zero. When GO is set, the device is processing a command. Only the device can modify FUNCTION once GO is set.

At initialization, the device sets GO and FUNCTION to zero to indicate that it is ready for processing. Host can then load FUNCTION and set GO to perform an operation. The device will clear both when it is done.

5.2 Interrupt Reason Register

The interrupt reason register is written by the display to indicate the event causing an interrupt. Interrupt reasons are bit-encoded; each bit indicates a different condition. Bit 15 of the interrupt reason register is an error bit. If set, the low-order 15 bits contain a display error code. Errors that also have bit 14 set are diagnostic errors.

The format of the Interrupt Reason Register is:

```
         15 14                                    0
        +-----------------------------------------+
CSR1    |E |D |    INTERRUPT REASON               |
        +-----------------------------------------+
```

Interrupt Reason Register

Like CSR0, use of CSR1 must be synchronized. When interrupt reason is zero, the device can complete a request. Only the device can write CSR1 when it is set to zero. When interrupt reason is nonzero, the host has not yet examined an event and the display may not write to CSR1. After the host has processed an interrupt, it sets CSR1 to zero. The device can then write the interrupt reason for the next event.

5.3 Device Event Register

The Device Event Register indicates events on keyboard or pointing device buttons attached to the display. The events are generally reported as up or down transitions on numbered keys. The format of the Device Event Register is:

```
         15         13 12     9  8  7             0
        +----------------------------------------+
CSR2    | Reserved   |   DEV   | T |  KEYCODE     |
        +----------------------------------------+
```

Keyboard Receive Register

where:

KEYCODE=CSR2<7:0> This field is written by the display to indicate on which key a transition has occurred. The key codes are integers from 0 to 255. The meaning of these codes depends on the particular device involved.

T=CSR2<8> The transition indicator specifies whether an up transition (T=0) or a down transition (T=1) has occurred. Not all of the button-event devices connected to the display may operate in exactly the same fashion, though. Some keys on the keyboard, for example, may only report down-transitions, while the buttons on the mouse report both up and down transitions. The user should refer to device-specific documentation for the particulars.

DEV=CSR2<12:9> The device field specifies the device responsible for generating the event. This field is encoded as follows:

- 0 = no device
- 1 = mouse
- 2 = keyboard
- 3 = tablet
- 4 = USART - AUX #1
- 5 = USART - console

5.4 Function Parameter Register

The function parameter register consists of two 16-bit CSRs, CSR3 and CSR4. This 32-bit register pair is loaded with the address of a packet to be transmitted to the display.

When a command packet (or list of linked commands) is completed (or aborted), the number of packets successfully completed is written into this register pair.

During initialization, the device writes in CSR3/CSR4 the location at which host memory is mapped within its local address space. The host must then offset all host memory addresses by this amount when communicating with the display.

```
         15                                     0
         +--------------------------------------+
CSR3     |          PACKET (LOW ADDRESS)         |
         +--------------------------------------+
CSR4     |          PACKET (HIGH ADDRESS)        |
         +--------------------------------------+
```

Function Parameter Register

5.5 Device Position Register

The Device Position Register contains the current position of the device whose movement generated the most recent interrupt. If the device is attached to the cursor, the interrupt reason will be "Cursor Moved"; the device position will be the same as the cursor position. If the device is not attached to the cursor, then the interrupt will indicate movement of that particular device. See also the sections on:

- Interrupt Reason Register
- Set Cursor Position
- Get Cursor Position
- Get Mouse Position
- Get Tablet Position
- Set Pointing Device Event Reporting

The format of the Device Position Register is:

```
          15                                       0
          +----------------------------------------+
CSR5      |                 XPOS                   |
          +----------------------------------------+
CSR6      |                 YPOS                   |
          +----------------------------------------+
```

Current Device Position

The Device Position Register should only be examined immediately following a device event interrupt. These interrupts will occur at most once every 1/60'th of a second. The display will update the Device Position Register prior to interrupting. Because the update requires two non-interlocked 16-bit writes, it would be possible for the host to read a new XPOS with an old YPOS if the registers were read at random times. Therefore, when servicing a device interrupt, the host should copy the device position to a local storage area.

5.6 Interrupt Vector Address Register

The Interrupt Vector Address Register is set by the host to the address of an entry in the UNIBUS interrupt vector block which contains the PC and PSL of the device interrupt service routine. When the device wants to interrupt the host, these values are used to invoke interrupt servicing.

```
          15                                       0
          +----------------------------------------+
CSR7      |      Interrupt Vector Address          |
          +----------------------------------------+
```

5.7 Initialization And Initial Display Requirements

The workstation graphics display must contain a ROM with code capable of processing a minimum of five immediate commands; that is, five function codes that can be loaded into the function field of CSR0. These commands are:

1. INIT (FUNCTION = 1). The INIT command causes the display micro-processor, if currently in operation, to halt execution and re-enter the initial ROM routine. An identical function

is performed on power-up of the display module. When the ROM is entered, it automatically stores the location at which host memory is mapped within the microprocessor's address space in the Function Parameter Registers, CSR3 and CSR4. When the host communicates a host memory address to the display, it must first add this value to the host address. This initial address is thus required by the host in order to transmit any host-mapped command packets to the display.

2. SEND PACKET (FUNCTION=2). The SEND PACKET command causes the display to read and process a command packet; the address of the command packet is loaded by the host into the Function Parameter Registers, CSR3 and CSR4, prior to setting FUNCTION CSR0 .

3. START DISPLAY (FUNCTION=3). The START DISPLAY command starts execution of the display micro-processor at the address specified in the Function Parameter Register (CSR3/CSR4). The start display command is executed during initialization to begin execution of display microcode. It causes the initial display ROM to invoke the firmware at the specified address.

   Start display is an immediate function and not a message. It is executed by loading the start display opcode into the CSR0 function field and setting the GO bit. The firmware responds with the "Started Executing" interrupt reason. The firmware start address is contained in the Function Parameter Register pair.

4. ABORT (FUNCTION=4). The ABORT function code stops any command being executed by the display. The display responds with the "command aborted" interrupt reason, and returns the number of successfully completed command packets in the Function Parameter Register.

5. EXECUTE_POWER_UP_SEQUENCE (FUNCTION=5). This function code causes the display to branch to its power-up sequence. This is a more "drastic" initialization than INIT. Some implementations may, for example, go in to a "self-test" mode here.

In addition, the initial display code must be capable of reading and processing two command packet opcodes as transmitted by a send packet request. These commands are:

1. Report Status - returns information about the display's status and addressing environment to the host,

2. Move Object - allows down-line loading of display micro-processor code into display local memory.

5.8 Aborting A Request

Since some commands can take seconds or minutes to complete and commands can be linked together, it must be possible to abort the current command sequence. To abort an operation, the host loads the ABORT function code into the CSR0 function field. Note that this violates the rules for access to CSR0. It is possible that after the ABORT is written and before the function field is examined by the device, the command will complete and the device will write a zero in the CSR0 function code. Therefore, when processing an abort, the host driver must be capable of accepting two interrupt reasons: aborted and command completion. The driver must also be able to ignore an aborted interrupt reason when no command is in progress.

6.0 COMMAND PACKET FORMATS

This section describes the formats of the display command packets. Each command packet has two parts, a header and a command message. The header format is common among all commands and has the following format:

```
 15     8 7      0
+----------------+
| qual. | opcode| 0
+----------------+
|   modifiers    | 2
+----------------+
|   modifiers    | 4
+----------------+
|      link      | 6
+----------------+
|      link      | 8
+----------------+
```

The opcode is an 8-bit value. Associated with the opcode is an 8-bit qualifiers field that contains command-independent qualifiers. In addition, many of the commands have command-dependent options in the specification of one or more parameters in the packet. For each parameter with specification options, the 32-bit opcode modifiers field indicates which option was chosen for that command. Although different options for a single parameter may require different size specifications, all packets for each opcode are of fixed size. Space is left in the packet for the largest size of each parameter offering several options. Thus, for each opcode, each parameter always starts at the same offset from the top of the packet, independent of the specification options chosen for the parameters.

The single command-independent qualifier currently defined is the Wait For Refresh qualifier, specified by bit zero of the 8-bit qualifiers field. If this bit is set to 1, the packet is processed as normal; however, physical screen operations are syncronized with the vertical blanking interrupt.

The link field is the 32-bit address of the next packet to be processed, if any. By linking packets, the host can initiate several commands with a single interaction. A zero value in the link field terminates the list. If any error occurs, the list is aborted at the point of error. The device returns the count of the number of packets successfully processed in the function parameter register, CSR3. For example, if the first packet is in error, a zero value will be returned, if the second is an error, a one will be returned, and so on.

6.1 Copy Area Command Packet

```
|                 15              0
|               +===============+          +================+         +===============+
|               | ADDRESS  (low)| 10       |    CONSTANT    |         | ADDRESS  (low)|
|               +-             -+          +----------------+         +-             -+
|               |         (high)| 12       |                |         |         (high)|
|   SOURCE      +---------------+          |                |         +---------------+
|   IMAGE       | SIZE      (x) | 14       |                |         | SIZE      (x) |
|   BITMAP      +-             -+          |   <reserved>   |         +-             -+
|               |           (y) | 16       |                |         |           (y) |
|               +-             -+          |                |         +-             -+
|               |           (z) | 18       |                |         |           (z) |
|               +===============+          +================+         +===============+
|   SOURCE      | OFFSET    (x) | 20       |                |         | ALIGNMENT (x) |
|   OFFSET      +-             -+          +-  <reserved>  -+         +-             -+
|               |           (y) | 22       |                |         | OFFSET    (y) |
|               +===============+          +================+         +===============+
|                  [bitmap]                   [constant]                 [halftone]
|
|               +===============+          +================+
|               | ADDRESS  (low)| 24       |                |
|               +-             -+          |                |
|               |         (high)| 26       |                |
|               +---------------+          |                |
|               | SIZE      (x) | 28       |                |
|               +-             -+          |   <reserved>   |
|               |           (y) | 30       |                |
|   SOURCE      +-             -+          |                |
|   MASK        |           (z) | 32       |                |
|               +---------------+          |                |
|               | OFFSET    (x) | 34       |                |
|               +-             -+          |                |
|               |           (y) | 36       |                |
|               +---------------+          +----------------+
|               | EXTENT    (x) | 38       | EXTENT     (x) |
|               +-             -+          |-              -|
|               |           (y) | 40       |            (y) |
|               +===============+          +================+
|                 [sub-bitmap]                [rectangle]
|          +================+
|          | ADDRESS   (low)|  42
|          +-              -+
|          |         (high)|  44
| DESTIN.  +----------------+
| IMAGE    | SIZE       (x) |  46
| BITMAP   +-              -+
|          |            (y) |  48
|          +-              -+
|          |            (z) |  50
|          +================+
| DESTIN.  | OFFSET     (x) |  52
| OFFSET   +-              -+
|          |            (y) |  54
|          +================+
|
|          +================+        +================+
|          | ADDRESS   (low)|  56    | LITERAL MAP    |
| MAP      +-              -+        +-              -+
|          |         (high)|  58    |                |
|          +================+        +================+
|            [map address]          [literal map table]
|
|          +================+        +================+
|          |ADDRESS    (low)|  60    | OFFSET     (x) |
| CLIPPING+-              -+        +-              -+
| RECTANG.|          (high)|  62    |            (y) |
|          +----------------+        +----------------+
|          | COUNT          |  64    | EXTENT     (x) |
|          +----------------+        +-              -+
|          |   <reserved>   |  66    |            (y) |
|          +================+        +================+
|           [rectangle list]        [literal rectangle]
```

6.2 Draw Curve Command Packet

The draw curve command packet is identical to the copy area command packet with the following additional fields:

```
                +================+
                | ADDRESS   (low)| 68
                +-             -+
PATH            |          (high)| 70
                +----------------+
                | COUNT          | 72
                +================+

                +================+
                | PATTERN LENGTH| 74
                +----------------+
PATTERN         |    PATTERN     | 76
STRING          +----------------+
                | PATTERN MULT.  | 78
                +================+

                +================+         +================+
                | PATTERN POSIT.| 80       | ADDRESS   (low)|
PATTERN         +----------------+         +-             -+
STATE           | PATTERN COUNT  | 82      |          (high)|
                +================+         +================+
                   [literal]               [state pointer]

                +================+         +================+         +================+
                | ADDRESS   (low)| 84      |    CONSTANT    |         | ADDRESS   (low)|
                +-             -+          +----------------+         +-             -+
                |          (high)| 86      |                |         |          (high)|
SECOND          +----------------+         |                |         +----------------+
SOURCE          | SIZE       (x) | 88      |                |         | SIZE       (x) |
                +-             -+          |   <reserved>   |         +-             -+
                |            (y) | 90      |                |         |            (y) |
                +-             -+          |                |         +-             -+
                |            (z) | 92      |                |         |            (z) |
                +================+         +================+         +================+
                | OFFSET     (x) | 94      |                |         | ALIGNMENT (x)  |
OFFSET          +-             -+          +- <reserved>  -+          +-             -+
                |            (y) | 96      |                |         | OFFSET     (y) |
                +================+         +================+         +================+
                   [bitmap]                   [constant]                 [halftone]
```

6.3 Print Text Command Packet

The print text command is similar to the copy area command, and has the following format. The differences are in specification of font (instead of source mask), the alternative specification of initial destination offset (instead of destination offset), and the additional parameters.

```
              15             0
             +================+        +================+        +================+
             | ADDRESS   (low)| 10     |    CONSTANT    |        | ADDRESS   (low)|
             +-             -+         +----------------+        +-             -+
             |          (high)| 12     |                |        |          (high)|
SOURCE       +----------------+        |                |        +----------------+
IMAGE        |                | 14     |                |        | SIZE       (x) |
             |                |        |   <reserved>   |        +-             -+
             |                | 16     |                |        |            (y) |
             |   <reserved>   |        |                |        +-             -+
             |                | 18     |                |        |            (z) |
             +================+        +================+        +================+
             |                | 20     |                |        | ALIGNMENT  (x) |
             +-  <reserved>  -+        +-  <reserved>  -+        +-             -+
             |                | 22     |                |        | OFFSET     (y) |
             +================+        +================+        +================+
               [source font]              [constant]               [halftone]

             +================+
             | ADDRESS   (low)| 24
             +-             -+
             |          (high)| 26
             +----------------+
             |                | 28
             +-             -+
             |                | 30
MASK         +-             -+
FONT         |                | 32
             +-             -+
             |   <reserved>   | 34
             +-             -+
             |                | 36
             +-             -+
             |                | 38
             +-             -+
             |                | 40
             +================+

             +================+
             | ADDRESS   (low)| 42
             +-             -+
             |          (high)| 44
DESTIN.      +----------------+
IMAGE        | SIZE       (x) | 46
BITMAP       +-             -+
             |            (y) | 48
             +-             -+
|         |            (z) | 50
|         +===============+
|
|         +===============+        +===============+
| INITIAL | OFFSET     (x) | 52     | ADDRESS  (low)|
| DESTIN. +-             -+        +-             -+
| OFFSET  |            (y) | 54     |         (high)|
|         +===============+        +===============+
|            [literal]              [offset pointer]
|
|         +===============+        +===============+
|         | ADDRESS  (low)| 56     | LITERAL MAP   |
| MAP     +-             -+        +-             -+
|         |         (high)| 58     |               |
|         +===============+        +===============+
|           [map address]         [literal map table]
|
|         +===============+        +===============+
|         |ADDRESS   (low)| 60     | OFFSET    (x) |
| CLIPPING+-             -+        +-             -+
| RECTANG.|         (high)| 62     |           (y) |
|         +---------------+        +---------------+
|         | COUNT         | 64     | EXTENT    (x) |
|         +---------------+        +-             -+
|         |   <reserved>  | 66     |           (y) |
|         +===============+        +===============+
|          [rectangle list]       [literal rectangle]
|
|         +===============+
|         | ADDRESS  (low)| 68
| TEXT    +-             -+
| STRING  |         (high)| 70
|         +---------------+
|         | COUNT         | 72
|         +===============+
|
|         +===============+
|         | ADDRESS  (low)| 74
| CONTROL +-             -+
| STRING  |         (high)| 76
|         +---------------+
|         | COUNT         | 78
|         +===============+
|
| INTER-  +===============+
| CHAR    | COUNT         | 80
| PAD     +===============+
|
| SPACE   +===============+
| PAD     | COUNT         | 82
|         +===============+
|
```

6.4 Fill Area Command Packet

The command packet for the fill area command has the following format:

```
          15             0
         +===============+         +===============+
         |    CONSTANT   | 10      | ADDRESS  (low)|
         +---------------+         +-            -+
         |               | 12      |        (high)|
SOURCE   |               |         +---------------+
IMAGE    |               | 14      | SIZE      (x) |
BITMAP   |  <reserved>   |         +-             -+
         |               | 16      |           (y) |
         |               |         +-             -+
         |               | 18      |           (z) |
         +===============+         +===============+
         |               | 20      | ALIGNMENT (x) |
         +- <reserved>  -+         +-             -+
         |               | 22      | OFFSET    (y) |
         +---------------+         +---------------+
            [constant]                [halftone]

         +===============+
         | ADDRESS  (low)| 24
         +-             -+
         |        (high)| 26
DESTIN.  +---------------+
IMAGE    | SIZE      (x) | 28
BITMAP   +-             -+
         |           (y) | 30
         +-             -+
         |           (z) | 32
         +===============+
DESTIN.  | OFFSET    (x) | 34
OFFSET   +-             -+
         |           (y) | 36
         +===============+

         +===============+         +===============+
         | ADDRESS  (low)| 38      | LITERAL MAP   |
MAP      +-             -+         +-             -+
         |        (high)| 40      |               |
         +===============+         +===============+
          [map address]          [literal map table]

         +===============+
         | OFFSET    (x) | 42
         +-             -+
CLIPPING |           (y) | 44
RECT.    +---------------+
         | EXTENT    (x) | 46
         +-             -+
         |           (y) | 48
         +===============+
|
|           +================+
|           | ADDRESS   (low)|  50
|           +-             -+
|   PATH    |          (high)|  52
|           +----------------+
|           | COUNT          |  54
|           +================+
|
|
```

6.5 Flood Area Command Packet

The command packet for the flood area command has the following format:

```
                 15              0
                +================+          +================+
                |    CONSTANT    | 10       | ADDRESS   (low)|
                +----------------+          +-              -+
                |                | 12       |          (high)|
        SOURCE  |                |          +----------------+
        IMAGE   |                | 14       | SIZE        (x) |
        BITMAP  |   <reserved>   |          +-              -+
                |                | 16       |             (y) |
                |                |          +-              -+
                |                | 18       |             (z) |
                +================+          +----------------+
                |                | 20       | ALIGNMENT (x)  |
                +-  <reserved>  -+          +-              -+
                |                | 22       | OFFSET     (y) |
                +----------------+          +----------------+
                    [constant]                 [halftone]

                +================+
                | ADDRESS   (low)| 24
                +-              -+
                |          (high)| 26
        DESTIN. +----------------+
        IMAGE   | SIZE       (x) | 28
        BITMAP  +-              -+
                |            (y) | 30
                +-              -+
                |            (z) | 32
                +================+

                +================+
        SEED    | OFFSET     (x) | 34
        POINT   +-              -+
                |            (y) | 36
                +================+

                +================+
                | OFFSET     (x) | 38
                +-              -+
        CLIPPING|            (y) | 40
        RECT.   +----------------+
                | EXTENT     (x) | 42
                +-              -+
                |            (y) | 44
                +================+

                +================+          +================+
                | ADDRESS   (low)| 46       |BOUNDARY (low)  |
        BOUNDARY+-              -+          +-              -+
        MAP     |          (high)| 48       |BOUNDARY (high)|
|
|        +===============+        +===============+
|        [boundary address]       [literal boundary]
|
```

6.6 Load Cursor Command Packet

```
                 15             0
              +================+       +================+         +================+
              | ADDRESS   (low)|  10   |    CONSTANT    |         | ADDRESS   (low)|
              +-              -+       +----------------+         +-              -+
CURSOR        |          (high)|  12   |                |         |          (high)|
SOURCE        +----------------+       |                |         +----------------+
IMAGE         | SIZE        (x)|  14   |                |         | SIZE        (x)|
BITMAP        +-              -+       |   <reserved>   |         +-              -+
              |             (y)|  16   |                |         |             (y)|
              +-              -+       |                |         +-              -+
              |             (z)|  18   |                |         |             (z)|
              +================+       +================+         +================+
SOURCE        | OFFSET      (x)|  20   |                |         | ALIGNMENT   (x)|
OFFSET        +-              -+       +- <reserved>   -+         +-              -+
              |             (y)|  22   |                |         | OFFSET      (y)|
              +================+       +================+         +================+
                  [bitmap]                [constant]                  [halftone]

              +================+       +================+
              | ADDRESS   (low)|  24   |                |
              +-              -+       |                |
              |          (high)|  26   |                |
              +----------------+       |                |
              | SIZE        (x)|  28   |                |
              +-              -+       |   <reserved>   |
CURSOR        |             (y)|  30   |                |
SOURCE        +-              -+       |                |
MASK          |             (z)|  32   |                |
              +----------------+       |                |
              | OFFSET      (x)|  34   |                |
              +-              -+       |                |
              |             (y)|  36   |                |
              +----------------+       +----------------+
              | EXTENT      (x)|  38   | EXTENT      (x)|
              +-              -+       |-              -|
              |             (y)|  40   |             (y)|
              +================+       +================+
                [sub-bitmap]              [rectangle]

              +================+       +================+
              | ADDRESS   (low)|  42   | LITERAL MAP    |
MAP           +-              -+       +-              -+
              |          (high)|  44   |                |
              +================+       +================+
                [map address]        [literal map table]

CURSOR        +================+
ATTRIB.       | VALUE          |  46
              +================+
```

6.7 Attach Cursor Command Packet

```
        +================+
DEVICE  | VALUE          | 10
        +================+
```

6.8 Set Cursor Position Command Packet

```
        +================+
        | OFFSET     (x) | 10
LOCATION+-             -+
        |            (y) | 12
        +================+
```

6.9 Get Cursor Position Command Packet

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

```
        +================+
CURRENT | RETURN     (x) | 10
CURSOR  +-             -+
POSITION| OFFSET     (y) | 12
        +================+
```

6.10 Get Mouse Position Command Packet

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

```
        +================+
CURRENT | RETURN     (x) | 10
MOUSE   +-             -+
POSITION| OFFSET     (y) | 12
        +================+
```

6.11 Set Mouse Characteristics Command Packet

```
         +================+            +================+
TRACKING| MULTIPLIER     | 10           |  THRESHOLD     | 10
RATIO    +----------------+            +----------------+
         | DIVISOR        | 12          | SCALE FACTOR   | 12
         +================+            +================+
         [Linear Tracking]           [Exponential Tracking]
```

6.12 Set Tablet Characteristics Command Packet

```
         +================+          +================+
TRACKING| MULTIPLIER     | 10        |  INPUT GSP  X | 10
RATIO    +----------------+          +----------------+
         | DIVISOR        | 12        |  INPUT GSP  Y | 12
         +================+          +----------------+
         |                |           |  OUTPUT GSP X | 14
         +- <reserved> -+           +----------------+
         |                |           |  OUTPUT GSP Y | 16
         +----------------+          +----------------+
        [Linear  Tracking]         [Quantized  Tracking]
```

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.

```
         +================+
CURRENT | RETURN     (x) | 10
TABLET   +-               -+
POSITION| OFFSET     (y) | 12
         +================+
```

6.14 Set Pointing Device Event Reporting

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

```
ENABLE   +================+
FLAGS    | VALUE          | 10
         +================+
```