Section 6. Mouse

Description

The Mouse is a cursor-positioning device that uses a ball and two encoders to indicate *x* and *y* movement to the system. Two push-button switches transmit their states directly to the system. The Mouse is connected to the system with a 1.8-meter (6-foot), shielded, four-conductor cable. The ball is removable for cleaning.

The following are descriptions of the four modes of operation:

Mode	Description
Reset	In this mode a self-test is initiated during power-on or by a Reset command. Upon satisfactory completion of the diagnostics, a completion code (hex AA) and an ID Code (hex 00) are transmitted to the system. The following defaults are set: sampling rate of 100 reports per second, linear scaling, stream mode, resolution of four counts per millimeter, and the Mouse is disabled. Any commands sent before the transmission of the completion code and the ID byte are ignored. The Mouse sends an error code of hex FC followed by an ID code of hex 00 immediately following a failure to complete the diagnostics. At this time, the Mouse is disabled and awaits further command input from the system.
Stream	In this mode, a data report is transmitted to the system if a switch is pressed or released, or if at least one count of movement has been detected. The maximum rate of transfer is the programmed sample rate. No transmissions occur if the Mouse is motionless unless a switch is operated, in which case, the incremental movement report is zero.
Remote	In this mode, data is transmitted only in response to a Read Data command.
Wrap	In this mode, any byte of data sent by the system, except hex EC and hex FF, is returned by the Mouse.

Programming Considerations

This section describes the Mouse commands and discusses the interface.

Commands

The ACK (hex FA) is always the first response to any valid input received from the system other than a Set Wrap Mode or Resend command, and supersedes all other Mouse output. If an interruption occurs during output of an ACK, the Mouse discards the ACK and accepts and responds to the new command. The ACK response is considered part of a special protocol between the system and the Mouse, and is not stored in any buffered internal memory. It is transmitted when required, then discarded.

The following lists all the valid commands.

Figure 6-1. Mouse Commands

Hex Code	Command
FF	Reset
FE	Resend
F6	Set Default
F5	Disable
F4	Enable
F3	Set Sampling Rate
F2	Read Device Type
F0	Set Remote Mode
EE	Set Wrap Mode
EC	Reset Wrap Mode
EB	Read Data
EA	Set Stream Mode
E9	Status Request
E8	Set Resolution
E7	Set Scaling 2:1
E6	Reset Scaling

The following describes valid commands:

Hex Code	Description
FF	**Reset:** This command causes the Mouse to enter the reset mode and do an internal self-test.
FE	**Resend:** Any time the Mouse receives an invalid command, it returns a Resend command to the system. The Mouse continues operating in the state it was in before receiving the invalid command. The count accumulators are cleared after receiving any command other than a Resend. The system sends this command when it detects an error in any transmission from the Mouse. The Resend command is sent following the Mouse transmission and before the system enables the interface, allowing the next output. When the Mouse receives a Resend command, it retransmits its last packet of data. If the last packet was a Resend command, it transmits the packet just prior to the Resend command. In the Stream mode, if a Resend command is received by the Mouse immediately following a 3-byte data packet transmission to the system, the Mouse resends the 3-byte data packet prior to clearing the count accumulators. This command also is described in "Error Handling" on page 6-10.
F6	**Set Default:** This command reinitializes all conditions to the power-on default state. Following receipt of this command, the Mouse sets up for a sampling rate of 100 reports per second, linear scaling, Stream mode, four counts per millimeter resolution, and disabled. No further action occurs until another command is sent from the system.
F5	**Disable:** This command is used in the Stream mode to stop transmissions initiated from the Mouse. It responds to all other commands while disabled. If the Mouse is in the Stream mode, it must be disabled before sending it any command that requires a response.
F4	**Enable:** This command is used in the Stream mode to begin transmissions.

F3, XX **Set Sampling Rate:** In the Stream mode, this command sets the sampling rate to the value indicated by the byte shown in the following figure.

Figure 6-2. Sampling Rate

Second Byte (in Hex)	Sample Rate
0A	10 per Second
14	20 per Second
28	40 per Second
3C	60 per Second
50	80 per Second
64	100 per Second
C8	200 per Second

F2 **Read Device Type:** This command always receives a response of hex 00.

F0 **Set Remote Mode:** This command sets the Remote mode. Data values are reported only in response to a Read Data command.

EE **Set Wrap Mode:** This command sets the Wrap mode. This mode remains until hex FF or hex EC is received.

EC **Reset Wrap Mode:** This command resets the Wrap mode. The Mouse returns to the previous mode of operation after receiving this command.

Note: If the Mouse is in the Stream mode and the Wrap mode is entered, the Reset Mode command causes the Mouse to reenter the Stream Mode in a disabled state.

EB **Read Data:** This command requests that all data defined in the data packet format be transmitted. This command is executed in either remote or stream mode. The data is transmitted even if there has been no movement since the last report or the switch status is unchanged. Following a Read Data command, the accumulators are cleared after a data transmission.

EA **Set Stream Mode:** This command sets the Stream mode.

E9 **Status Request:** When this command is issued by the system, the Mouse responds with a 3-byte status report as follows.

Figure 6-3. Status Request Format

Byte	Bit	Description
1	7	Reserved
	6	0 = Stream Mode, 1 = Remote Mode
	5	0 = Disabled, 1 = Enabled
	4	0 = Scaling 1:1, 1 = Scaling 2:1
	3	Reserved
	2	1 = Left Button Pressed
	1	Reserved
	0	1 = Right Button Pressed
2	7 - 0	Current Resolution Setting
3	7 - 0	Current Sampling Rate

E8, XX **Set Resolution:** The Mouse provides four resolutions selected by the second byte of this command as follows.

Figure 6-4. Set Resolution

Second Byte (in Hex)	Resolution (Counts per mm)
00	1
01	2
02	4
03	8

E7 **Set Scaling 2:1:** Scaling is used to provide a course or fine tracking response. At the end of a sample interval in the Stream mode, the current *x* and *y* data values are converted to new values. The sign bits are not involved in this conversion. The relationship between the input and output counts follows.

Figure 6-5. Set Scaling 2:1

Input	Output
0	0
1	1
2	1
3	3
4	6
5	9
N (≥6)	2.0 x N

2:1 scaling is performed only in Stream mode. In response to a Read Data command, the current value before conversion is sent.

E6 **Reset Scaling:** This command restores scaling to 1:1.

Data Report

When operating in Stream mode, a data report is sent at the end of a sample interval if a button remains pressed or is released during the interval, or if at least one count of Mouse movement has occurred since the last report.

If a button is pressed during a sample interval, it is reported as pressed at the end of the interval. If a button remains pressed, no further reports are transmitted until it is released unless there is further Mouse movement to report. When movement is to be reported and there has been no change in the button status during the last interval, the buttons are reported in their current state (1 = pressed, 0 = not pressed). If a button is pressed and released during a sample interval, it is reported as pressed at the end of the interval. Any transmission reporting button status change can also include travel data.

In the Remote mode, a data report is sent in response to a Read Data command. The buttons are reported in their current state at the time of transmission.

The following data report format is valid for both the Stream and Remote modes and is three bytes long.

Figure 6-6. Data Packet Report Format

Byte	Bit	Description
1	7	*y* Data Overflow 1 = Overflow
	6	*x* Data Overflow 1 = Overflow
	5	*y* Data sign 1 = Negative
	4	*x* Data sign 1 = Negative
	3	Reserved
	2	Reserved
	1	Right Button Status 1 = Pressed
	0	Left Button Status 1 = Pressed
2	7 - 0	*X* Data
3	7 - 0	*y* Data

The data values are in binary and the least significant bit (LSB) indicates 0.25 millimeter of movement when operating with linear scaling at four counts per millimeter resolution. Negative values of *x* and *y* data are expressed in twos complement where 0.25 millimeter movement in the negative direction is expressed with all bits set to 1 and the sign bit set to 1. The full movement number is a 9-bit twos-complement number.

The count accumulators do not wrap around. If a count during a sample period is greater than the format allows, the maximum count is reported and the overflow bit for that coordinate is set. After a transmission, the accumulators are cleared to 0.

Error Handling

The Mouse issues a Resend command (hex FE) following receipt of an invalid input or any input with incorrect polarity. If two invalid inputs are received in succession, an error code of hex FC is sent to the system.

Following a system transmission, a response is sent within 25 milliseconds if the system requires a response or if an error is detected in the transmission. If the Mouse is in the Stream mode, the system disables the Mouse before issuing any command requiring a response. When a command requiring a response is issued by the system, another command should not be issued until either the response is received or 25 milliseconds has elapsed. No more than four non-response commands should be sent in succession.

Data Transmission

During a data transmission, CLK is used to clock serial data. The Mouse generates the clocking signal when sending data to and receiving data from the system. The system requests the Mouse receive system data output by forcing the DATA line low and allowing CLK to go high.

Communication is bi-directional using the CLK and DATA signal lines. The signal for each of these lines comes from open collector devices, allowing either the Mouse or the system to drive a line low. During a non-transmission state, CLK and DATA are both held high.

Data Output: When the Mouse is ready to transmit data, it must first check for its own Inhibit or system request-to-send status on the CLK and DATA lines. If CLK is low (inhibit status), data is continuously updated and no transmissions are started. If CLK is high and DATA is low (request-to-send), data is updated. Data is received from the system and no transmissions are started by the Mouse until CLK and DATA are both high.

If CLK and DATA are both high, the Mouse proceeds to output zero start-bits, eight data bits, a parity bit, and a stop bit if a transmission is required. Data is valid prior to the falling edge of CLK and beyond

the rising edge of CLK. During transmission, the Mouse checks for line contention by checking for a low level on CLK at intervals not to exceed 100 milliseconds. Contention occurs when the system drives CLK low to inhibit the Mouse output after the Mouse has started a transmission. If this occurs before the rising edge of the tenth clock (parity bit), the Mouse internally stores the data packet in the its buffer and returns DATA and CLK to a high level. If the contention does not occur by the tenth clock, the transmission is completed.

Following a transmission, the system inhibits the Mouse by holding CLK low until it can service the input or until the system receives a request to send a response from the Mouse. The system raises CLK to allow the next transmission.

Data Input: When the system is ready to send data, it first checks to see if the Mouse is transmitting data. If the Mouse is transmitting, the system can override the output by driving CLK low prior to the tenth clock. If the Mouse transmission is beyond the tenth clock, the system receives the data.

If the Mouse is not transmitting or if the system chooses to override the output, the system drives CLK low for a period of not less than 100 microseconds while preparing for output. When the system is ready to output 0 start bit (DATA line is low), it allows CLK to go to an active level. The Mouse checks for this state not to exceed every 10 milliseconds.

If request-to-send is detected, the Mouse clocks in 11 bits. Following the tenth clock, the Mouse checks for DATA being high, and if found, then drives DATA low (line control bit), and clocks once more. This signals the system to return to the ready state when it can accept input or go to the Inhibit mode until ready. If DATA is low following clock 10, a framing error has occurred and the Mouse continues to clock until DATA is high, then clocks the line control bit and requests a resend.

For each system command or data transmission that requires a response, the system waits for the Mouse to respond before sending its next output. The response must be within 20 milliseconds, unless the system inhibits the Mouse output or inhibits the data transmissions from the system that require a response.

If the system initiates a command or data transmission and the response is invalid or has a parity error, the system resends the command or data. If after two retries the response is still invalid or has a parity error, the system resets the Mouse.

Figure 6-7. Data Frame

Bit	Function
Start Bit	Always 0
0	Least Significant Bit (LSB)
1 - 6	Data Bits 1-6
7	Most Significant Bit (MSB)
Parity Bit	Odd Parity
Stop Bit	Always 1

Mouse Device Driver Interface

The function of the device driver is to allow the Mouse to operate with applications that use the interface designed by Microsoft® . The device driver can only be loaded using DOS commands and cannot be loaded as a DOS device driver from the CONFIG.SYS file.

Applications access the device driver by issuing an interrupt hex 33. The device driver determines which function to perform by the value in the AX register. Parameters are passed from the calling application to the device driver in the BX, CX and DX registers (Function 16 also uses the SI and DI registers). High-level programming languages can access the device driver by making a call to the entry point of the device driver.

To make a call from a BASIC program you must:

- Assign the offset and segment address of the software to a pair of integer variables in your program. The entry offset and segment address are in memory. To get these values, insert the following statements into your program:

```
10 DEF SEG=0
20 MSEG=256*PEEK(51*4+3)+PEEK(51*4+2)
30 MOUSE=256*PEEK(51*4+1)+PEEK(51*4)
40 IF MSEG OR MOUSE THEN 60
50 PRINT "Mouse Driver not found":END
60 DEF SEG=MSEG
70 IF PEEK (MOUSE)=207 THEN 50 '207 is IRET
80 'Mouse driver is there, continue
```

  Be sure these statements appear before making any calls to Mouse functions.

Microsoft is a registered trademark of the Microsoft Corporation.

- Use the CALL statement to make the call. The statement should have the form:

  ```
  CALL MOUSE (M1%,M2%,M3%,M4%)
  ```

  where MOUSE is the variable containing the entry offset of the Mouse software and M1%, M2%, M3%, and M4% are the names of the integer variables you have chosen for parameters in this call (constants and non-integer variables are not allowed). All four parameters must appear in the CALL statement, even if no value is assigned to one or more of them.

  Set the values for M1%, M2%, M3% and M4% to the AX, BX, CX, and DX registers described in the device driver functions.

  To ensure the variables are integer variables, use the percent sign (%) as part of all the variable names. You may also use the DEFINT statement at the beginning of your program. For example, the statement

  ```
  10 DEFINT A-Z
  ```

  defines all variables as integer. If this statement appears at the beginning of the program, the variable names do not need to include the percent sign.

  An application gets status by continuously polling the device driver for Mouse movement and button position. The device driver also supports the ability of an application to hook in a subroutine to be called whenever a Mouse event occurs. Status is passed to the subroutine when it is called from the device driver.

To interface with the device driver from an assembler language program, set up the registers specified in the following figures according to the desired function and issue an interrupt hex 33 as specified in the following figures.

Call to Device Driver

The device driver receives data from the Mouse whenever the Mouse is moved or whenever a button position changes. During initial setup of the Mouse and the device driver, a pointer to a subroutine in the device driver is passed to BIOS through INT 15 AH = C2 AL = 7. The subroutine is called whenever Mouse movement is detected or a button position changes.

The following data is passed to the device driver subroutine on the stack.

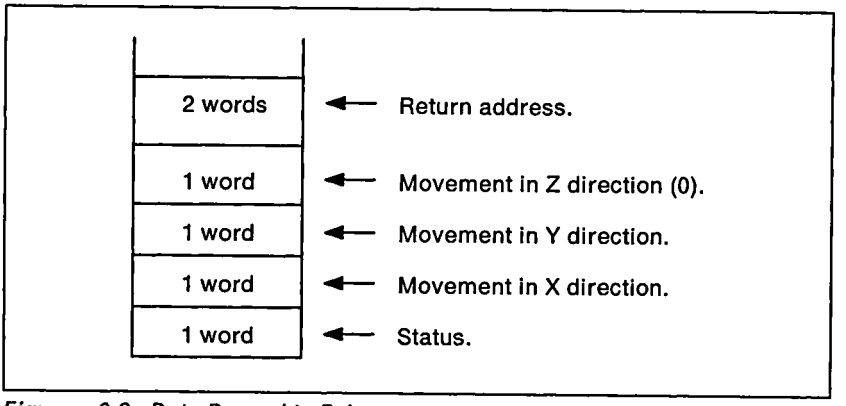

Figure 6-8. Data Passed to Driver

The data is read from the stack and processed. While the data is being processed, no more data will be sent by the Mouse BIOS. If a device driver function is currently in progress the data is lost.

Device Driver Functions

Function 0: Installed Flag and Reset: This function initializes variables and initializes the Mouse if it is attached. If the Mouse is not attached, the device driver sends an unsuccessful return code to the calling program.

Figure 6-9. Function 0: Installed Flag and Reset

Input		Output	
Register	**Value**	**Register**	**Value**
AX	0	AX	Mouse Status 0 = Unsuccessful -1 = Successful
		BX	Number of Buttons 2 If successful 0 Otherwise

The following device driver variables are initialized.

Figure 6-10. Device Driver Variables

Description	**Value**
xy Movements	0
Button Presses	0
Button Releases	0
Cursor Position at Last Button Press	0
Cursor Position at Last Button Release	0
Cursor Position	Screen Center
Conditional Off Region	Hex 07FFF
Scaling Factor (Horizontal)	8
Scaling Factor (Vertical)	16
Text Cursor	Inverted Box
Graphics Cursor	Arrow
Hot Spot	(-1,-1)
Min/Max Cursor Position	Depends on Video Mode

The Mouse is initialized to the following values.

Figure 6-11. Mouse Initial Values

Description	**Value**
Sample Rate	100 Reports per Second
Resolution	8 Counts per mm
Data Package Size	3 Bytes
Scaling	2:1

Function 1: Show Cursor: The internal cursor flag is incremented; if it is equal to 0, the cursor is displayed. This function always increments the cursor flag. Upon Reset the flag is set to -1, the first show will display a cursor. Additional show calls will increment the cursor flag to positive values.

Note: Hide Cursor calls should be paired with Show Cursor calls.

Figure 6-12. Function 1: Show Cursor

Input		Output	
Register	**Value**	**Register**	**Value**
AX	1	—	—

Function 2: Hide Cursor: The internal cursor flag is decremented and the cursor is removed from the screen if it is displayed. For every Hide Cursor function call made, a Show Cursor function call must be made to increment the cursor flag.

Figure 6-13. Function 2: Hide Cursor

Input		Output	
Register	**Value**	**Register**	**Value**
AX	2	—	—

Function 3: Get Position and Button Status: The current horizontal and vertical cursor positions and the button status are returned to the calling program. Both horizontal and vertical cursor positions, are returned as "virtual screen coordinates." They might not map directly into the video (graphics) mode.

Figure 6-14. Function 3: Get Position and Button Status

Input		Output	
Register	**Value**	**Register**	**Value**
AX	3	BX	Button Status Bit 1 1 = Right Button Pressed 0 = Right Button Released Bit 0 1 = Left Button Pressed 0 = Left Button Released
		CX	Cursor Position (Horizontal)
		DX	Cursor Position (Vertical)

Function 4: Set Cursor Position: This function sets the horizontal and vertical cursor positions.

Figure 6-15. Function 4: Set Cursor Position

Input		Output	
Register	**Value**	**Register**	**Value**
AX	4	—	—
CX	New Horizontal Coordinate	—	—
DX	New Vertical Coordinate	—	—

Function 5: Get Button Press Information: This function returns the following information, for the specified Mouse button, to the calling program:

- The status of the button
- The number of times the button was pressed since the last call to this function
- The cursor position of the last button pressed.

Figure 6-16. Function 5: Get Button Press Information

Input		Output	
Register	**Value**	**Register**	**Value**
AX	5	AX	Button Status
BX	Button (0 = Left, 1 = Right)	BX	Count of Button Presses
		CX	Cursor (Horizontal) at Last Press
		DX	Cursor (Vertical) at Last Press

Function 6: Get Button Release Information: This function returns the following information, for the specified Mouse button, to the calling program:

- The status of the button
- The number of times the button was released since the last call to this function
- The cursor position of the last button release.

Figure 6-17. Function 6: Get Button Release Information

Input		Output	
Register	**Value**	**Register**	**Value**
AX	6	AX	Button Status
BX	Button (0 = Left, 1 = Right)	BX	Count of Button Releases
		CX	Cursor (Horizontal) at Last Release
		DX	Cursor (Vertical) at Last Release

Function 7: Set Minimum and Maximum Horizontal Position: This function sets the minimum and maximum values on the x-axis. If the minimum value specified is larger than the maximum value, the values are exchanged.

If the maximum value is larger than allowed, the value is set to the maximum value allowed. This also applies to the minimum value. The minimum and maximum values are defined by the graphic mode.

Figure 6-18. Function 7: Set Minimum and Maximum Horizontal Position

Input		Output	
Register	**Value**	**Register**	**Value**
AX	7	—	—
CX	Minimum Position	—	—
DX	Maximum Position	—	—

Function 8: Set Minimum and Maximum Vertical Position: This function sets the minimum and maximum values on the *y*-axis. If the minimum value specified is larger than the maximum value, the values are exchanged.

If the maximum value is larger than allowed, the value is set to the maximum value allowed. This also applies to the minimum value. The minimum and maximum values are defined by the graphic mode.

Figure 6-19. Function 8: Set Minimum and Maximum Vertical Position

Input		Output	
Register	**Value**	**Register**	**Value**
AX	8	—	—
CX	Minimum Position	—	—
DX	Maximum Position	—	—

Function 9: Set Graphics Cursor Block: This function is used to define the screen mask, cursor mask, and hot spot for the graphics cursor. The cursor masks are 16-bit by 16-bit matrixes used to determine the shape and color of the graphics cursor. In a 640 by XXX (XXX can be 200, 350, or 480) graphics mode, one bit in each mask represents a PEL. In a 320 by XXX (XXX can be 200 or 350) graphics mode, two horizontal bits represent a PEL.

Figure 6-20. Function 9: Set Graphics Cursor Block

Input		Output	
Register	**Value**	**Register**	**Value**
AX	9	—	—
BX	Horizontal Cursor Hot Spot (-16 to 16)	—	—
CX	Vertical Cursor Hot Spot (-16 to 16)	—	—
DX	Pointer to Screen and Cursor Masks	—	—

The graphics cursor is defined as follows for graphics modes less than mode 7.

Figure 6-21. Graphics Modes Less Than 7

Screen Mask	Cursor Mask	Result On Screen
0	0	0
0	1	1
1	0	Unchanged
1	1	Inverted

The graphics cursor is defined as follows for graphics modes greater than mode 7.

Figure 6-22. Graphics Modes Greater Than 7

Screen Mask	Cursor Mask	Result On Screen
0	0	Black
0	1	White
1	0	Unchanged
1	1	Unchanged

Function 10: Set Text Cursor: This function defines the cursor and screen mask for the text cursor and defines the text cursor as the hardware or software cursor.

Figure 6-23. Function 10: Set Text Cursor

Input		Output	
Register	**Value**	**Register**	**Value**
AX	10	—	—
BX	Cursor Select 0 = Software text cursor 1 = Hardware text cursor	—	—
CX	Screen Mask Value or Scan Line Start	—	—
DX	Cursor Mask Value or Scan Line Stop	—	—

Function 11: Read Motion Counters: This function returns the horizontal (*x*) and vertical (*y*) counts since the last time this function was called. The *x* and *y* are units of distance approximately equal to 0.5 millimeter (0.02 inch). A positive count indicates movement to the right or down. A negative count indicates movement to the left or up. The count is always within the range of -32768 to 32767.

Figure 6-24. Function 11: Read Motion Counters

Input		Output	
Register	**Value**	**Register**	**Value**
AX	11	CX	Count (Horizontal)
		DX	Count (Vertical)

Function 12: Set User-Defined Subroutine Input Mask: This function allows the calling program to set up a subroutine that will be called by the device driver when a condition in the mask occurs.

Figure 6-25. Function 12: Set User-Defined Subroutine Input Mask

Input		Output	
Register	**Value**	**Register**	**Value**
AX	12	—	—
CX	Call Mask	—	—
DX	Address Offset to Subroutine	—	—

The Mouse hardware interrupts automatically stop execution of the program and call the specified subroutine when one or more of the conditions defined by the Call Mask occur. When the subroutine finishes, the program continues execution from the point of interruption.

The Call Mask is an integer value that defines conditions that cause an interrupt. Each bit in the Call Mask corresponds to a specific condition, as shown in the following.

Figure 6-26. Call Mask Bit Definitions

Mask Bit	Condition
15 - 5	Not Used
4	Right Button Released
3	Right Button Pressed
2	Left Button Released
1	Left Button Pressed
0	Cursor Position Changes

To enable the subroutine, set the corresponding Call Mask bit to 1. To disable the subroutine, set the corresponding bit to 0. Function 0 automatically disables all interrupts.

Note: Before the program ends, restore initial values of the Call Mask and the subroutine address.

Function 13: Light Pen Emulation Mode On: When the Light Pen Emulation mode is turned on, the Mouse emulates a light pen. The cursor position represents the light pen position; pressing both buttons represents the light pen being pressed down.

Figure 6-27. Function 13: Light Pen Emulation Mode On

Input		Output	
Register	**Value**	**Register**	**Value**
AX	13	—	—

Function 14: Light Pen Emulation Mode Off: This function disables light pen emulation.

Figure 6-28. Function 14: Light Pen Emulation Mode Off

Input		Output	
Register	**Value**	**Register**	**Value**
AX	14	—	—

Function 15: Set xy/PEL Ratio: This function sets the horizontal and vertical ratios of Mouse movement to cursor movement (8 PEL). The default value for the horizontal ratio is 8 *xy* to 8 PEL. This ratio moves the cursor all the way across the screen when the Mouse moves 81 millimeters (3.2 inch). The default value for the vertical ratio is 16 *xy* to 8 PEL. This ratio moves the cursor all the way down the screen when the Mouse moves 50.1 millimeters (2.0 inch).

Figure 6-29. Function 15: Set xy/PEL Ratio

Input		Output	
Register	**Value**	**Register**	**Value**
AX	15	—	—
CX	*x* per 8 PEL Ratio (Horizontal)	—	—
DX	*y* per 8 PEL Ratio (Vertical)	—	—

Function 16: Conditional Off: This function defines an area on the screen that erases the cursor. Function 1 (Show Cursor) must be called after calling this function.

Figure 6-30. Function 16: Conditional Off

Input		Output	
Register	**Value**	**Register**	**Value**
AX	16	—	—
CX	Upper *x* Screen Coordinate	—	—
DX	Upper *y* Screen Coordinate	—	—
SI	Lower *x* Screen Coordinate	—	—
DI	Lower *y* Screen Coordinate	—	—

Function 19: Set Double Speed Threshold: This function sets a threshold speed (in *xy*'s per second), which if exceeded, causes the cursor to move twice the distance on the screen.

Figure 6-31. Function 19: Set Double Speed Threshold

Input		Output	
Register	**Value**	**Register**	**Value**
AX	19	—	—
DX	Threshold Speed in Movements per Second	—	—

Function 20: Swap User Interrupt Vector: This function sets new values for the Call Mask and the subroutine address and returns the values previously specified.

Figure 6-32. Function 20: Swap User Interrupt Vector

Input		Output	
Register	**Value**	**Register**	**Value**
AX	20	—	—
CX	New User Interrupt Event Mask	CX	Old User Interrupt Event Mask
ES:DX	New User Interrupt Vector	ES:DX	Old User Interrupt Vector

The Mouse hardware interrupts automatically stop execution of the program and call the specified subroutine when one or more of the conditions defined by the Call Mask occur. When the subroutine finishes, the program continues execution from the point of interruption.

The Call Mask is an integer value that defines conditions that cause an interrupt. Each bit in the Call Mask corresponds to a specific condition, as shown in the following.

Figure 6-33. Call Mask Bit Definitions

Mask Bit	Condition
15 - 5	Not Used
4	Right Button Released
3	Right Button Pressed
2	Left Button Released
1	Left Button Pressed
0	Cursor Position Changes

When the program makes a call to the subroutine, it loads the following information into the microprocessor registers.

Figure 6-34. Register Information

Register	Information
AX	Condition Mask (Similar to the Call Mask except a bit is set only if the condition has occurred)
BX	Button State (see Figure 14)
CX	Cursor Coordinate (Horizontal)
DX	Cursor Coordinate (Vertical)
DI	Horizontal Counts
SI	Vertical Counts

Note: The DS register contains the Mouse device driver data segments. The subroutine is responsible for setting the DS register as needed.

To enable the subroutine, set the corresponding Call Mask bit to 1 and pass the mask in the CX register. To disable the subroutine, set the corresponding bit to 0 and pass the mask in the CX register. Function 0 automatically disables all interrupts.

Note: Before the program ends, restore the initial values of the Call Mask and the subroutine address.

Function 21: Query Save State Storage Requirements: This function returns the size of the buffer required to store the current state of the Mouse device driver. It is used with functions 22 and 23 to temporarily interrupt a program using the Mouse and execute another program also using the Mouse.

Figure 6-35. Function 21: Query Save State Storage Requirements

Input		Output	
Register	**Value**	**Register**	**Value**
AX	21	—	—
—	—	BX	Buffer Size Required for Functions 22 and 23

Function 22: Save Mouse Driver State: This function saves the current Mouse device driver state in a buffer allocated by the program. It is used with functions 21 and 23 to temporarily interrupt a program using the Mouse and execute another program that also using the Mouse.

Before calling Function 22, the program should call Function 21 to determine the buffer size required for saving the Mouse device driver state, then allocate the appropriate amount of memory.

Figure 6-36. Function 22: Save Mouse Driver State

Input		Output	
Register	**Value**	**Register**	**Value**
AX	22	—	—
ES:DX	Pointer to the Buffer	—	—

Function 23: Restore Mouse Driver State: This function restores the last Mouse device driver state saved by Function 22. It is used with functions 21 and 22 to temporarily interrupt a program that also uses the Mouse. To restore the Mouse device driver state saved by function 22, call Function 23 at the end of the interrupt program.

Figure 6-37. Function 23: Restore Mouse Driver State

Input		Output	
Register	**Value**	**Register**	**Value**
AX	23	—	—
ES:DX	Pointer to the Buffer	—	—

Function 24: Set Alternate Mouse User Subroutine: This function allows the calling program to set up a subroutine that will be called by the device driver when a condition in the Event Mask occurs. This function is similar to Function 12 except the Event Mask can also include a combination of certain keystrokes. Up to three routines can be defined by calling this function. After the third call an error is returned.

Figure 6-38. Function 24: Set User-Defined Subroutine Input Mask

Input		Output	
Register	**Value**	**Register**	**Value**
AX	12	—	—
CX	Call Mask	—	—
DX	Address Offset to Subroutine	—	—

The register used when entering user subroutines are shown in the following figure.

Figure 6-39. Register Conventions

Register	Definition
AX	Condition Mask
BX	Button State (see Figure 14)
CX	Cursor Coordinate (Horizontal)
DX	Cursor Coordinate (Vertical)
SI	Delta *x* and *y* Movement (Horizontal)
DI	Delta *x* and *y* Movement (Vertical)

Note: The DS register contains the Mouse device driver data segments. The subroutine is responsible for setting the DS register as needed.

Bit definitions for the Event Mask are shown in the following figure.

Figure 6-40. Event Mask Bit Definitions

Bits	Definition
15 - 8	Reserved
7	Alt Key Pressed During Button Event
6	Ctrl Key Pressed During Button Event
5	Shift Key Pressed During Button Event
4	Right Button Released
3	Right Button Pressed
2	Left Button Released
1	Left Button Pressed
0	Cursor Moved

Note: When bits 7 - 5 are set, the corresponding shift state must be active to allow other events to cause the user subroutine to be called.

Function 25: Get User Alternate Interrupt Vector: This function returns a pointer to the subroutine defined by Function 24. The Event Mask condition in the CX register must match the Event Mask used to define the subroutine in Function 24. If no match is found, a 0 is returned in the CX register.

Figure 6-41. Function 25: Get User Alternate Interrupt Vector

Input		Output	
Register	**Value**	**Register**	**Value**
AX	25	—	—
—	—	BX:DX	User Interrupt Vector (or Undefined)
CX	User Interrupt Event Mask Shift Bits	CX	User Interrupt Event Mask (0 if No Match)

Function 26: Set Mouse Sensitivity: This function sets the mouse to cursor-movement sensitivity by defining the number of *x*'s and *y*'s that are equal to a single PEL. It also sets the double-speed threshold for the mouse. The sensitivity value must be in the range of 1 to 100, where a value of 1 specifies an *xy*-to-PEL ratio of 1:1. The default value is 50. These values are not reset by a Mouse Reset function call. If a value greater than 100 is passed in the BX or CX register, the number is truncated to 100.

Figure 6-42. Function 26: Set Mouse Sensitivity

Input		Output	
Register	**Value**	**Register**	**Value**
AX	26	—	—
BX	Horizontal *x* and *y* Coordinates per PEL	—	—
CX	Vertical *x* and *y* Coordinates per PEL	—	—
DX	Double Speed Threshold	—	—

Function 27: Get Mouse Sensitivity: This function returns the value set by Function 26.

Figure 6-43. Function 27: Get Mouse Sensitivity

Input		Output	
Register	**Value**	**Register**	**Value**
AX	27	—	—
—	—	BX	Horizontal *x* and *y* Coordinates per PEL
—	—	CX	Vertical *x* and *y* Coordinates per PEL
—	—	DX	Double-Speed Threshold

Function 29: Set CRT Page Number: This function specifies which CRT page the Mouse cursor will be displayed on.

Figure 6-44. Function 29: Set CRT Page Number

Input		Output	
Register	**Value**	**Register**	**Value**
AX	29	—	—
BX	CRT Page for Cursor Display	—	—

Function 30: Get CRT Page Number: This function returns the number of the CRT page the Mouse cursor will be displayed on.

Figure 6-45. Function 30: Get CRT Page Number

Input		Output	
Register	**Value**	**Register**	**Value**
AX	30	—	—
—	—	BX	CRT Page for Cursor Display

Function 31: Disable Mouse Driver: This function restores all interrupt vectors (except the interrupt hex 33 vector) used by the Mouse device driver to their values before the Mouse device driver was installed. The value returned in ES:BX can be used to restore the value of interrupt hex 33 vector. The Mouse device driver uses interrupt hex 10. It also uses interrupt hex 71 for systems with the 8086 microprocessor, or interrupt hex 74 for systems with the 80286 or 80386 microprocessor. AX is set to -1 if the Mouse device driver is unable to restore one or more of the vectors it is using.

Figure 6-46. Function 31: Disable Mouse Driver

Input		Output	
Register	**Value**	**Register**	**Value**
AX	31	—	—
—	—	AX	31 if Disable Was Successful -1 if Disable Was Unsuccessful
—	—	ES:BX	Previous Function 31 Vector

Function 32: Enable Mouse Driver: This function reinstalls the interrupt vector values used by the Mouse device driver. The pointer to the Mouse device driver is reinstalled through interrupt hex 15.

Note: This function will rechain any vectors unchained as a result of calls made to Function 32.

Figure 6-47. Function 32: Enable Mouse Driver

Input		Output	
Register	**Value**	**Register**	**Value**
AX	32	—	—

Function 33: Software Reset: This function is identical to Function 0, except that the Mouse is not reset.

Figure 6-48. Function 33: Software Reset

Input		Output	
Register	**Value**	**Register**	**Value**
AX	33	—	—
—	—	AX	-1 if the Mouse Driver is Installed 33 if the Mouse Driver is Not Installed
—	—	BX	2 if AX = -1

Function 36: Get Driver Version, Mouse Type, and IRQ Number: This function gets the version number of the Mouse driver, the type of Mouse it requires, and the number of the interrupt-request (IRQ) type. For example, a function call 36 to version 6.10 would return the value 0610 (binary coded decimal) in the BX register.

The Mouse type is contained in the CH register. A value of 1 indicates a bus Mouse, a value of 4 indicates a Personal System/2® Mouse.

The value for the interrupt-request type is contained in the CL register.

Figure 6-49. Function 36: Get Driver Version, Mouse Type, and IRQ Number

Input		Output	
Register	**Value**	**Register**	**Value**
AX	36	BH	Major Version Number
—	—	BL	Minor Version Number
—	—	CH=1	Bus Mouse
—	—	CH=2	Serial Mouse
—	—	CH=4	Personal System/2 Mouse
—	—	CL=0	Personal System/2 Value

Video - Supported Modes: The device driver supports all VGA (video graphics array) video modes. In graphics modes hex 0D through hex 13, the cursor is displayed only in black or white. (See Function 9 on page 6-20 for more detail about graphics cursors).

All VGA registers that are altered by the device driver during cursor displaying and erasing are returned to the state they were in when the device driver code was called. Modes hex 0D through hex 12 map video using bit planes.

Personal System/2 is a registered trademark of the International Business Machines Corporation.

Connector

The Mouse is connected to the system by a 6-pin connector. Figure 6-50 shows the pin configuration and signal assignments.

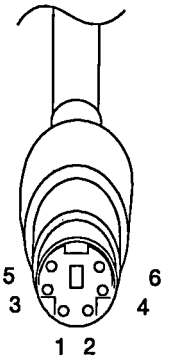

Figure 6-50. Mouse Connector Pin Assignments

Pin	Signal
1	Data
2	No Connection
3	Ground
4	+ 5 V dc
5	Clock
6	No Connection

Specifications

The following are specifications for the Mouse.

Programmable Resolution: (counts per millimeter) 1, 2, 4 (default), or 8.

Programmable Sampling Rate: (reports per second) 10, 20, 30, 40, 60, 80, 100 (default), or 200.

Data Modes: Stream (default), Remote, or Wrap.

Scaling: 1:1, 2:1.

Power: +5 V dc, ±10%, 70 milliamperes (maximum).

Maximum Tracking Speed: ≥ 200 millimeters per second.

Size	**Millimeters**	**Inches**
• Length:	110	4.3
• Depth:	66	2.6
• Heigth:	32	1.3

Weight	**Kilograms**	**Pounds**
	1.23	0.5