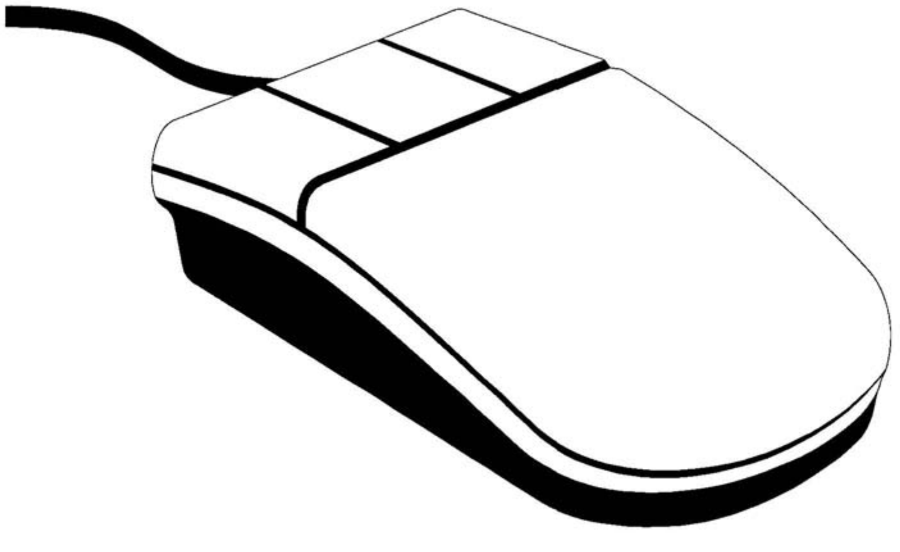
CHAPTER 11

Communications: Interfacing to the PS/2 Mouse



11 Communications: Interfacing to the PS/2 Mouse

Just like the PS/2 keyboard, the PS/2 mouse uses the PS/2 synchronous bi-directional serial communication protocol described in section 10.4 and shown in Figures 10.1 and 10.2. Internally, the mouse contains a ball that rolls two slotted wheels. The wheels are connected to two optical encoders. The two encoders sense x and y motion by counting pulses when the wheels move. It also contains two or three pushbuttons that can be read by the system and a single-chip microcontroller. The microcontroller in the mouse sends data packets to the computer reporting movement and button status.

It is necessary for the computer or in this case the FLEX chip to send the mouse an initialization command to have it start sending mouse data packets. This makes interfacing to the mouse more difficult than interfacing to the keyboard. As seen in Table 11.1, the command value needed for initialization after power up is F4, enable streaming mode.

Table 11.1 PS/2 Mouse Commands.

Commands Sent to Mouse	Hex Value
Reset Mouse	FF
Mouse returns AA, 00 after self-test	
Resend Message	FE
Set to Default Values	F6
Enable Streaming Mode Mouse starts sending data packets at default rate	F4
Disable Streaming Mode	F5
Set sampling rate XX is number of packets per second	F3, XX
Read Device Type	F2
Set Remote Mode	EE
Set Wrap Mode Mouse returns data sent by system	EC
Read Remote Data Mouse sends 1 data packet	EB
Set Stream Mode	EA
Status Request Mouse returns 3-bytes with current settings	E9
Set Resolution XX is 0, 1, 2, 3	E8, XX
Set Scaling 2 to 1	E7
Reset Scaling	E6

Table 11.2 PS/2 Mouse Messages.

Messages Sent by Mouse	Hex Value
Resend Message	FE
Two bad messages in a row	FC
Mouse Acknowledge Command Sent by Mouse after each command byte	FA
Mouse passed self-test	AA

After streaming mode is enabled, the mouse sends data to the system in three byte data packets that contain motion and pushbutton status. The format of a three-byte mouse data packet is seen in Table 11.3.

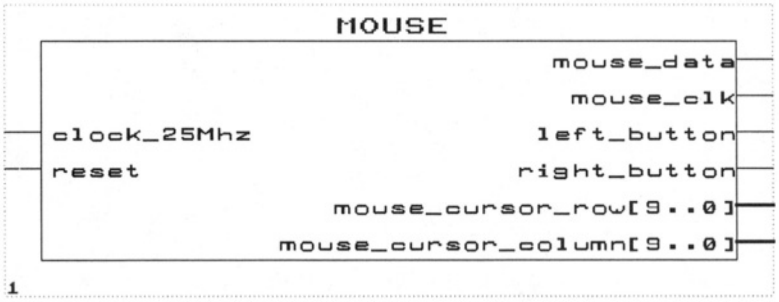
Table 11.3 PS/2 Mouse Data Packet Format.

	MSB							LSB
Bit	7	6	5	4	3	2	1	0
Byte 1	*Yo*	*Xo*	*Ys*	*Xs*	1	M	R	L
Byte 2	X7	X6	X5	X4	X3	X2	X1	X0
Byte 3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

L = Left Key Status bit	(For buttons 1 = Pressed and 0 = Released)
M = Middle Key Status bit	(This bit is reserved in the standard PS/2 mouse protocol, but some three button mice use the bit for middle button status.)
R = Right Key Status bit	
X7 – X0 = Moving distance of X in two's complement	(Moving Left = Negative; Moving Right = Positive)
Y7 – Y0 = Moving distance of Y in two's complement	(Moving Up = Positive; Moving Down = Negative)
Xo = X Data Overflow bit	(1 = Overflow)
Yo = Y Data Overflow bit	(1 = Overflow)
Xs = X Data sign bit	(1 = Negative)
Ys = Y Data sign bit	(1 = Negative)

11.1 The Mouse UP1core

The UP1core function Mouse is designed to provide a simple interface to the mouse. This function initializes the mouse and then monitors the mouse data transmissions. It outputs a mouse cursor address and button status. The internal operation of the Mouse UP1core is rather complex and the fundamentals are described in the section that follows. Like the other UP1core functions, it is written in VHDL and source code is provided.



To interface to the mouse, a clock filter, serial-to-parallel conversion and parallel-to-serial conversion with two shift registers is required along with a state machine to control the various modes. See Chapter 10 on the PS/2 keyboard for an example of a clock filter design.

11.2 Mouse Initialization

Two lines are used to interface to the mouse, clock and data. The lines must be tri-state bi-directional, since at times they are driven by the mouse and at other times by the FLEX chip. All clock, data, and handshake signals share two tri-state, bi-directional lines, clock and data. These two lines must be declared bi-directional when pin assignments are made and they must have tri-state outputs in the interface. The mouse actually has open collector outputs that can be simulated by using a tri-state output. The mouse always drives the clock signal for any serial data exchanges. The FLEX chip can inhibit mouse transmissions by pulling the clock line Low at any time.

The FLEX chip drives the data line when sending commands to the mouse. When the mouse sends data to the FLEX chip it drives the data line. The tri-state bi-directional handshaking is described in more detail in the IBM PS/2 Technical Reference manual. A simpler version with just the basics for operation with the UP 1 is presented here. Just like the keyboard, the mouse interface is more reliable if a clock filter is used on the clock line.

At power-up, the mouse runs a self-test and sends out the codes AA and 00. The clock and data FLEX chip outputs are tri-stated before downloading the UP 1, so they float High. High turns out to be ready to send for mouse data, so AA and 00 are sent out prior to downloading and need not be considered in the interface. This assumes that the mouse is plugged in before applying power to the UP 1 board and downloading the design.

The default power-up mode is streaming mode disabled. To get the mouse to start sending 3-byte data packets, the streaming mode must be turned on by sending the enable streaming mode command, F4, to the mouse from the FLEX chip. The clock tri-state line is driven Low by the FLEX for at least 60us to inhibit any data transmissions from the mouse. This is the only case when the FLEX chip should ever drive the clock line. The data line is then driven Low by the FLEX chip to signal that the system has a command to send the mouse.

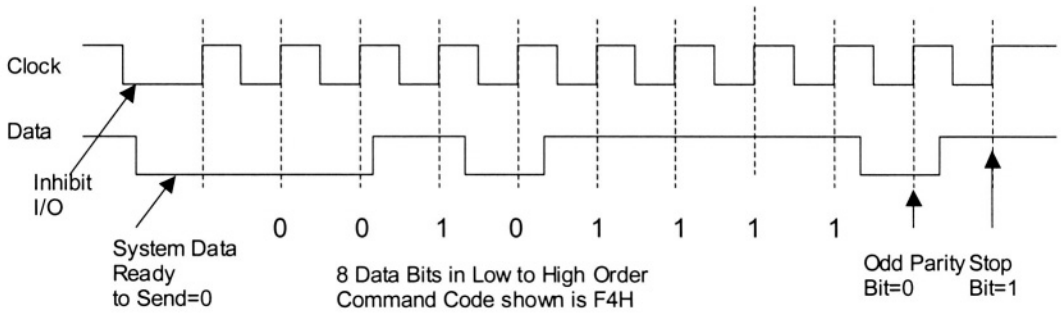


Figure 11.1 Transmission of Mouse Initialization Command.

The clock line is driven High for two clocks at 25Mhz and then tri-stated to simulate an open collector output. This reduces the rise time and reflections on the mouse cable that might be seen by the fast FLEX chip logic as the clock line returns to the High state. As an alternative, the mouse clock input to the FLEX could be briefly disabled while the clock line returns to the High state.

Next the mouse, seeing data Low and clock High, starts clocking in the serial data from the FLEX chip. The data is followed by an odd parity bit and a High stop bit. The handshake signal of the data line starting out Low takes the place of the start bit when sending commands to the mouse.

With the FLEX chip clock and data drivers both tri-stated, the mouse then responds to this message by sending an acknowledge message code, FA, back to the FLEX chip. Data from the mouse includes a Low start bit, eight data bits, an odd parity bit, and a High stop bit. The mouse, as always, drives the clock line for the serial data transmission. The mouse is now initialized.

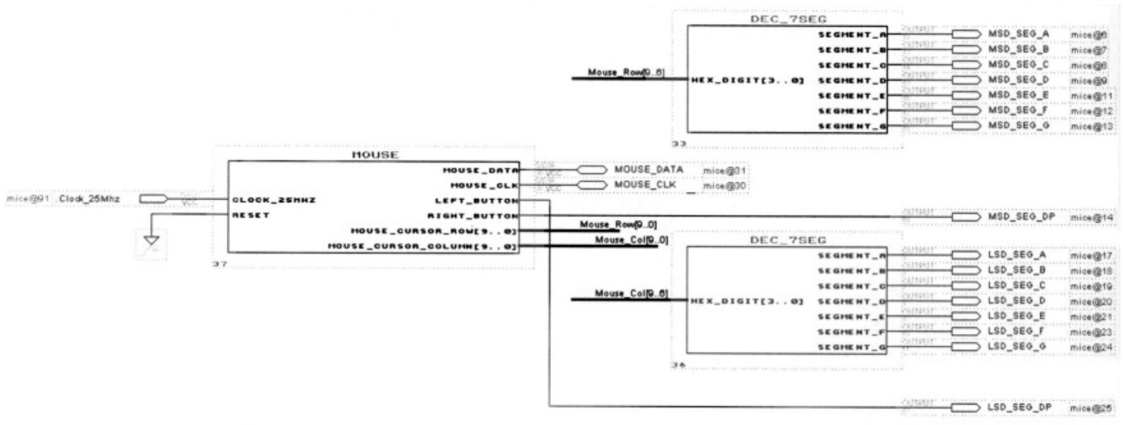
11.3 Mouse Data Packet Processing

As long as the FLEX chip clock and data drivers remain tri-stated, the mouse then starts sending 3-byte data packets at the power-up default sampling rate of 100 per second. Bytes 2 and 3 of the data packet contain X and Y motion values as was seen in Table 11.2. These values can be positive or negative, and they are in two's complement format.

For a video cursor such as is seen in the PC, the motion value will need to be added to the current value every time a new data packet is received. Assuming 640 by 480 pixel resolution, two 10-bit registers containing the current cursor row and column addresses are needed. These registers are updated every packet by adding the sign extended 8-bit X and Y motion values found in bytes 2 and 3 of the data packet. The cursor normally would be initialized to the center of the video screen at power-up.

11.4 An Example Design Using the Mouse UP1core

In this example design, the mouse drives the seven-segment LED displays. The most-significant display contains the high bits of the cursor row address and the least-significant display contains the high bits of the cursor column address. The left button and the right button drive the MSD and LSD decimal points. The mouse cursor powers up to the center position of the 640 by 480 video screen. Note that the mouse clock and data pins must be bi-directional.



11.5 For Additional Information

The IBM PS/2 Hardware Interface Technical Reference Manual, IBM Corporation, 1988, contains information on the mouse in the Keyboard and Auxiliary Device Controller Chapter.

11.6 Laboratory Exercises

1. Generate a video display that has a moving cursor controlled by the mouse using the Mouse and VGA_Sync UP1cores. Use the mouse buttons to change the color of the cursor.

2. Use the mouse as input to a video etch-a-sketch. Use a monochrome 64 by 64 1-bit pixel RAM in your video design. Display a cursor. To draw a line, the left mouse button should be held down.

3. Use the mouse as an input device in another design with video output or a simple video game such as pong, breakout, or Tetris.

4. Write a mouse driver in Verilog. Use the mouse information provided in sections 11.2 and 11.3.