13. The PS/2 Mouse

Mice come in various flavours - serial mice, PS/2 mice, busmice, USB mice. Below a little about mice using the PS/2 protocol, since these also use the keyboard controller.

A mouse has a number of buttons (1-5 is common) and must report button presses. It has some way of detecting motion, and must report the amount of movement in the X and Y direction, usually as differences with the previously reported position, in a (dx,dy) pair. Touchpads can also report absolute position.

Reports come in the form of mouse packets of between 1 and 8 bytes. Various protocols are in use.

13.1 Modes

A PS/2 mouse can be in *stream mode* (the default). In this mode it produces a stream of packets indicating mouse movements and button presses. Or it can be in *remote mode*. In this mode the mouse only sends a packet when the host requests one, using the **eb** command. Finally, it can be in *echo* ("wrap") *mode*, in which everything the host sends is echoed back, until either a reset (**ff**) or clear echo mode (**ec**) is received.

13.2 Scaling

Scaling can be set to 1:1 or 2:1. This affects stream mode only. In 2:1 scaling: If the unscaled absolute value of dx or dy is 6 or more, it is doubled. Otherwise, for the unscaled value 0,1,2,3,4,5,6, the scaled value 0,1,1,3,6,9,12 is sent.

13.3 PS/2 mouse protocol

The default protocol

The standard PS/2 protocol uses 3-byte packets, as follows:

Yovfl	Xovfl	dy8	dx8	1	Middle Btn	Right Btn	Left Btn
dx7	dx6	dx5	dx4	dx3	dx2	dx1	dx0
dy7	dy6	dy5	dy4	dy3	dy2	dy1	dy0

It gives the movement in the X and Y direction in 9-bit two's complement notation (range -256 to +255) and an overflow indicator. It also gives the status of the three mouse buttons. When this protocol is used, the **f2** Read mouse ID command is answered by **00**.

Intellimouse

The Microsoft Intellimouse uses the above protocol until scrolling wheel mode is activated by sending the magic sequence **f3 c8 f3 64 f3 50** (set sample rate 200, 100, 80). In this mode, the Read mouse ID command returns **03**, and 4-byte packets are used:

Yovfl	Xovfl	dy8	dx8	1	Middle Btn	Right Btn	Left Btn
dx7	dx6	dx5	dx4	dx3	dx2	dx1	dx0
dy7	dy6	dy5	dy4	dy3	dy2	dy1	dy0
dz3	dz3	dz3	dz3	dz3	dz2	dz1	dz0

Here the last byte gives the movement of the scrolling wheel in 4-bit two's complement notation (range -8 to +7) and the leading four bits are just copies of the sign bit.

Intellimouse Explorer mouse

The Explorer mouse protocol allows for scrolling wheel and five buttons. It is activated by first sending the magic sequence for Intellimouse, and then, when the Intellimouse ID has been seen, sending the magic sequence **f3 c8 f3 c8 f3 50** (set sample rate 200, 200, 80). In this mode, the Read mouse ID command returns **04**, and 4-byte packets are used:

Yovfl	Xovfl	dy8	dx8	1	Middle Btn	Right Btn	Left Btn
dx7	dx6	dx5	dx4	dx3	dx2	dx1	dx0
dy7	dy6	dy5	dy4	dy3	dy2	dy1	dy0
0	0	5th Btn	4th Btn	dz3	dz2	dz1	dz0

Lots of other protocols occur, and only incomplete data is known about most of them. Some examples.

Typhoon mouse

The Typhoon optical mouse is reported to send 6-byte packets. Bytes 1-3 are as for the default PS/2 protocol. Byte 4 equals byte 1. Byte 5 gives the Z axis movement, one of **ff**, **00**, **01**. Byte 6 is 0. Of course the idea is that this packet looks like two ordinary packets and ordinary PS/2 mouse drivers will handle it. The 6-byte mode is activated by sending the magic sequence **f3 c8 f3 64 f3 50 f3 3c f3 28 f3 14** (set sample rate 200, 100, 80, 60, 40, 20). It is recognized by the ID **08**.

13.4 Mouse Commands

Every command or data byte sent to the mouse (except for the resend command **fe**) is ACKed with **fa**. If the command or data is invalid, it is NACKed with **fe**. If the next byte is again invalid, the reply is ERROR: **fc**.

*Command **d0**: Read extended ID*

Read up to 256 bytes.

*Commands **d1**-**df**: Vendor unique commands*

*Command **d1**: Logitech PS/2++ command*

This command was to be used, followed by an arbitrary data sequence. Now replaced by the sliced commands using **e8**.

*Command **e1**: Read secondary ID*

> Replies with two bytes. An IBM TrackPoint returns **01** as first byte, and a second byte depending on the model.

*Command **e2**: IBM TrackPoint command*

> Followed by several parameter bytes. For details, see ykt3dext.pdf.

*Command **e6**: Set mouse scaling to 1:1*

> Often ingredient in magic sequences.

*Command **e7**: Set mouse scaling to 2:1*

> Often ingredient in magic sequences.

*Command **e8**: Set mouse resolution*

> This command is followed by a byte indicating the resolution (0, 1, 2, 3: 1, 2, 4, 8 units per mm, respectively). It is used in magic sequences to transport two bits, so that four of these are needed to send a byte to the mouse. See below.

*Command **e9**: Status request*

> This command returns three bytes:
>
> First a status byte: Bit 7: unused, 0. Bit 6: 0: stream mode, 1: remote mode. Bit 5: 0: disabled, 1: enabled. Bit 4: 0: scaling set to 1:1, 1: scaling set to 2:1. Bit 3: unused, 0. Bit 2: 1: left button pressed. Bit 1: 1: middle button pressed. Bit 0: 1: right button pressed.
>
> Then a resolution byte: 0, 1, 2, 3: 1, 2, 4, 8 units per mm, respectively.
>
> Finally a sample rate (in Hz).
>
> See below for special Synaptics Touchpad handling.

*Command **ea**: Set stream mode*

*Command **eb**: Read data*

> Read a mouse packet. Needed in remote mode to ask the mouse for data. Also functions in stream mode.

*Command **ec**: Clear echo mode*

*Command **ee**: Set echo mode*

*Command **f0**: Set remote mode*

*Command **f2**: Read mouse ID*

> (Only supported on some systems.) This command reads a 1-byte mouse ID. The reply is a single byte **00**. Wait at least 10ms after issuing this command.
>
> For the keyboard reply, see above.
>
> BallPoint (trackball) devices return a single byte **02**, Intellimouse returns **03**, Explorer Mouse returns **04**, 4d Mouse returns **06**, 4dplus Mouse returns **08**,as does the Typhoon mouse.

*Command **f3**: Set mouse sample rate*

> (Only supported on some systems.) Set mouse sample rate in Hz. If the given sampling rate is acceptable the ACK is **fa**. Otherwise the NACK is **fe**, and the host can correct. If it is incorrect again **fc** is sent. Correct values are, e.g., 10, 20, 40, 60, 80, 100, 200.

*Command **f4**: Mouse enable*

> The stream mode mouse data reporting is disabled after a reset and after the disable command. This command enables it again.

*Command **f5**: Mouse disable*

> This stops mouse data reporting in stream mode. In stream mode, this command should be sent before sending any other commands.

*Command **f6**: Set defaults*

> If this command is recognized, a reset is done (set sampling rate 100 Hz, resolution 4 counts/mm, stream mode, disabled, scaling 1:1), but no diagnostics are performed. For some enhanced mice that require a magic sequence to get into enhanced mode, this command will reset them to default PS/2 mode.

*Command **fe**: Resend*

> If this command is recognized, the last mouse packet (possibly several bytes) is resent. There is no ACK to this command, but if the last reply was ACK, it is sent.

*Command **ff**: Mouse reset*

> A self-test is performed. When OK, the response is **aa 00**. On error, the response is **fc 00**. The mouse is reset to default PS/2 mode.

13.5 Sliced parameters

For more advanced mouse modes it is necessary to send data to the mouse. There is now a commonly accepted way.

First Logitech tried to use the **d1** command followed by an arbitrary data sequence. While the IBM specs reserve **d1-df** for vendor unique commands, it turns out that not all BIOSes will transmit such codes. So Logitech drops the **d1** and uses the sequence **e8** *aa* **e8** *bb* **e8** *cc* **e8** *dd* to transmit the byte *aabbccdd*, where *aa*, *bb*, *cc*, *dd* are 2-bit quantities. In this way an arbitrarily long sequence of bytes can be transmitted.

For synchronization purposes it is possible to separate such groups of four **e8** commands by an **e6** command. Indeed, such separation may be required: Synaptics Touchpads react to **e9** or **f3** commands preceded by precisely four **e8** commands.

Magic knock

For example, the "magic knock" **d1 39 db** that sets a device that understands it in PS/2++ mode, becomes **e8 00 e8 03 e8 02 e8 01 e6 e8 03 e8 01 e8 02 e8 03**, abbreviated {E8}0321 {E6} {E8}3123. Note that 0321 and 3123 do not have repeated symbols. If they had, too intelligent intermediate hardware transmitting these sequences might see a superfluous command and suppress it.

Magic unknock

PS/2++ mode is cleared again by the "magic unknock" {E8} 0323 or D1 3B from an external device, and {E8} 0321 or D1 39 from an internal device. (These commands differ so that in setups where the same commands are sent to internal and external devices, they can be commanded separately.)

For a decription of the PS/2++ format, see ps2ppspec.htm.

13.6 Synaptics Touchpad

A few sketchy details. For nice precise information, get the Synaptics interfacing guide.

Status request

When preceded by an 8-bit request number encoded via four **e8** commands, the **e9** status request returns modified output, somewhat dependent on the Touchpad model.

*Request **00**: Identify Touchpad*

This request returns three bytes, of which the middle one is the constant **47**. This is the way to recognize a Touchpad. The low order four bits of the third word contain the major model version number, the first word contains the minor version number, and the high order four bits of the third word contain the (obsolete) model code.

*Request **01**: Read Touchpad Modes*

This request returns three bytes, of which the first two are the constants **3b** and **47**. The last byte is the mode byte

ABS	Rate	-	-	Baud/Sleep	DisGest	PackSize	Wmode

Here ABS indicates *absolute mode* (instead of the default relative mode).

Rate is 0 for 40 packets/sec, 1 for 80 packets/sec. The PS/2 sampling rate value is ignored.

Baud/Sleep indicates the baud rate when used with a serial protocol (0: 1200 baud, 1: 9600 baud). It must be set whenever ABS or Rate is set. When used with the PS/2 protocol this bit indicates *sleep mode* - a low power mode in which finger activity is ignored and only button presses are reported.

DisGest is the "disable gestures" bit. When set, we have classical mouse behaviour. When cleared, "tap" and "drag" processing is enabled.

PackSize is used for the serial protocol only (and then chooses between 6-, 7- and 8-byte packets, also depending on the Wmode bit).

Wmode is used in absolute mode only. When set the packets also contain the W value. (This value indicates the amount of contact: 0: two-finger contact, 1: three-finger contact, 2: pen contact, 3: reserved, 4-7: ordinary finger contact, 8-15: wide finger or palm contact.)

This described Touchpad 4.x. Earlier models had up to four mode bytes. This request would return mode bytes 1 and 2 in the first and last result byte, and request **02** would return mode bytes 3 and 4.

*Request **02**: Read Capabilities*

This request returns three bytes, of which the middle one is the constant **47**. The first and third byte are the high-order and low-order parts of the capability word. (Thus on Touchpad 4.x. On earlier models mode bytes 3 and 4 are returned.)

This capability word has 16 bits. Bit 15 indicates that capabilities are supported. Bit 4 indicates that Sleep is supported (for the PS/2 protocol). Bit 3 indicates that four buttons (Left, Right, Up, Down) are supported. Bit 1 indicates that multi-finger detection is supported. Bit 0 indicates that palm detection is supported.

*Request **03**: Read Model ID*

*Request **06**: Read Serial Number Prefix*

*Request **07**: Read Serial Number Suffix*

*Request **08**: Read Resolution*

Mode setting

When preceded by an 8-bit request number encoded via four **e8** commands, the **f3** **14** (set sample rate 20) command sets the mode byte to the encoded number. (Thus on Touchpads 4.x. Older models have more mode bytes and several such commands.)

13.7 Vendor extensions

There is a complicated forest of "magic sequences" that enable vendor extensions. Recognizing all of these is a very obscure activity.

(Moreover, recognizing these may be counterproductive: if the mouse has special capabilities which are activated by a special sequence, and it is connected to the computer via a KVM switch that does not know about this special protocol, then switching away and back will leave the mouse in the non-special state. This leads to non-functioning mice.)

A 2002 Logitech file describes the following procedure for recognizing the mouse type:

Stage 1: Send **ff**: reset. The reply is ignored. (Most common is **aa** **00**.)

Stage 2: Send **f3 0a f2**: set sample rate and ask for ID. If the reply is **02**, we have a trackball - it has its own protocol. (The usual reply is **00**.)

Stage 3: Send **e8 00 e6 e6 e6 e9**: set resolution and scaling (three times), and request status. The reply consists of three bytes *s1 s2 s3*. An old-fashioned mouse would report 0 in the second status byte *s2* (since that is the resolution and we just set it).

If *s2* is nonzero then: *s2* is the number of buttons, *s3* is the firmware revision, *s1* has the firmware ID (device type) bits 6-0 in bits 3-0,6-4, while bit 7 of s1 indicates support for the **e7 e7 e7 e9** command.

If *s1*=**d0** and *s2*=**03** and *s3*=**c8**, suspect Synaptics.

If *s1* and *s2* are zero but *s3* equals **0a**, suspect Alps. (*s3*=**0a** is as expected, but *s1*=0 is not)

Stage 4: If bit 7 of *s1* is set, or if we suspect Alps, send **e8 00 e7 e7 e7 e9**: set resolution and scaling (three times), and request status. The reply consists of three bytes *t1 t2 t3*. Of course, we already know that this is not an old-fashioned mouse.

If *t2*=**01** and FirmwareID < 0x10 and *t1* >> 6 = 1, then conclude that we have a Cordless MouseMan (RA12).

If *t2*=**01** and FirmwareID < 0x10 and *t1* >> 6 = 3, then conclude that we have a Cordless MouseMan (RB24).

Other cases with *t2*=**01** are for new cordless mice.

If we suspect Synaptics and *t2*=0 and *t3*=**0a**, then conclude that we have a Synaptics touchpad.

If we suspect Alps and *t1*=**33**, then conclude that we have an Alps touchpad.

Stage 5: If we don't know the type yet, send **f3 c8 f3 64 f3 50 f2**: Set sampling rate to 200, 100, 80 Hz, and ask for ID. The reply is a single byte. If we get 3, conclude that we have an IntelliMouse. (And this sequence is the initialization sequence for the IntelliMouse.)

Stage 6: Send **ff**: reset. Now the device is no longer in any special state.

Stage 7: If we don't know the type yet, send **e8 00 e8 00 e8 00 e8 00 e9**: set resolution to 0 (four times), and ask for status. The reply consists of three bytes *u1 u2 u3*. If *u2*=**47** and *u3*=**13**, then conclude that we have a new Synaptics touchpad.

Stage 7a: At this point we can narrow down to model type. If the thing is Synaptics or Alps, then Logitech is no longer interested. If it has 3 buttons, FirmwareID 1 and firmware revision **50**, then conclude that it is a Logitech Mouseman.

Stage 8: If we think it is a touchpad, detect whether it has programmable RAM. Send **e6 e8 00 e8 00 e8 00 e8 00 e9**. The reply consists of three bytes *v1 v2 v3*. If *v1*=**06** and *v2*=**00**, then conclude that we have a Touchpad TP3 with programmable RAM.

Stage 9: Test whether the device understands the Logitech PS/2++ protocol. Send the "magic knock" **f5 e8 00 e8 03 e8 02 e8 01 e6 e8 03 e8 01 e8 02 e8 03 f4**. Check whether the device replies with an extended report.

Next Previous Contents