PS/2 Mouse

From OSDev Wiki

Also be sure to read Mouse Input.

Contents

- 1 Overview
- 2 Mouse Device Over PS/2
- 3 Mouse Extensions
- 4 See Also
  - 4.1 Articles
  - 4.2 External Links
  - 4.3 Implementations

Overview

The PS/2 Mouse is a device that talks to a PS/2 controller using serial communication. Ideally, each different type of PS/2 controller driver should provide some sort of standard/simple "send byte/receive byte" interface, and the PS/2 Mouse driver would use this interface without caring about lower level details (like what type of PS/2 controller the device is plugged into).

Mouse Device Over PS/2

Here is the table of command a generic PS/2 compatible mouse understands:

Standard PS/2 Mouse Commands

Byte	Data byte	Description
0xFF	None	Reset
0xFE	None	Resend
0xF6	None	Set Defaults
0xF5	None	Disable Data Reporting
0xF4	None	Enable Data Reporting
0xF3	Sample rate, ranges from 10-200.	Set Sample Rate
0xF2	None	Get Device ID. See Detecting PS/2 Device Types for the response bytes.
0xF0	None	Set Remote Mode
0xEE	None	Set Wrap Mode
0xEC	None	Reset Wrap Mode
0xEB	None	Read Data
0xEA	None	Set Stream Mode
0xE9	None	Status Request
0xE8	**Byte** / **Resolution** 00 / 1 count/mm 01 / 2 count/mm 02 / 4 count/mm 03 / 8 count/mm	Set Resolution

The most common command reply is *0xFA* from the master (mouse), which means acknowledge. You may then get a variable number of bytes afterwards depending on the command. You may also receive other command replies which may state that the master (mouse) has encountered an error decoding your command. For a more detailed list check out some of the links above or look through the Linux source tree.

First, you have to enable the mouse on the PS/2 bus. This requires sending one byte which is clocked over the PS/2 interface. You will then get a response regarding the result. By sending *0xF4* (Enable Data Reporting) the mouse should reply back with a *0xFA* which means acknowledgement. Then afterwards as the mouse pointer is moved it will send back the generic packet format like below. Unless you enable an enhanced mode for the mouse (non-standard) this is what you will get when ever the mouse is moved.

Generic PS/2 Mouse Packet Bits

BYTE	7	6	5	4	3	2	1	0
0	yo	xo	ys	xs	ao	bm	br	bl
1	xm							
2	ym							

Code	Description
yo	Y-Axis Overflow
xo	X-Axis Overflow
ys	Y-Axis Sign Bit (9-Bit Y-Axis Relative Offset)
xs	X-Axis Sign Bit (9-Bit X-Axis Relative Offset)
ao	Always One
bm	Button Middle (Normally Off = 0)
br	Button Right (Normally Off = 0)
bl	Button Left (Normally Off = 0)
xm	X-Axis Movement Value
ym	Y-Axis Movement Value

Each X and Y axis value is relative. The mouse device does not track it's location in absolute coordinates. This should also be apparent by the 9-bit values. Instead, it sends back saying I moved this far to the left, to the right, down, or up. To keep track of a mouse position you need to accumulate these relative offsets into a absolute position offset in your code:

```
mouse_x = mouse_x + mouse_packet_rel_x
mouse_y = mouse_y + mouse_packet_rel_y
```

Being these 9-bit values are signed the above pseudo would work.

Also, if you simply read the *xm* or *ym* fields you will get an 8-bit unsigned value. Which, if used as unsigned will yield incorrect behavior. If you convert it into a signed 8-bit value you will get behavior that is similar to correct, but strange artifacts will appear when the mouse is moved fast. The correct way to produce a 9-bit or greater signed value is as follows:

```
    state = first_byte
    d = second_byte
    rel_x = d - ((state << 4) & 0x100)
    d = third_byte
    rel_y = d - ((state << 3) & 0x100)
```

The pseudo code above will cause *((state << 4) & 0x100)* to equal *0x100* only if the signed bit (9'th bit stored in the first byte) is set. If the 9'th bit is set then the value is deemed negative, but the value in *second_byte* is not stored in one or two's complement form. It is instead stored as a positive 8-bit value. So, if *second_byte* is say a *2* then it will become *2 minus 0* since the negative (9'th bit) is off. But, if it is on then it will become *2 minus 0x100* which will produce the twos complement, or *-2*. It will also cause the register to be correctly sign extended no matter it's size.

Mouse Extensions

Here, an example of mouse that supports extensions. To maintain backwards compatibility you should have to activate these features through the PS/2 bus. Various mouse devices use different ways. Linux mouse drivers for example sometimes handle multiple different devices which all share the same standard packet format above, or at least support the compatibility mode described above.

IntelliMouse Explorer								
BYTE	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
3	vs	hs						

See Also

Articles

- PS/2
- "8042" PS/2 Controller
- PS/2 Keyboard

External Links

- www.Computer-Engineering.org/ps2mouse (http://www.computer-engineering.org/ps2mouse/)

Implementations

- Linux (http://lxr.linux.no/#linux+v3.5.4/drivers/input/mouse/psmouse-base.c) (C,GPL)