Elantech Touchpad Driver

Extra information for hardware version 1 found and provided by Steve Havelka

Version 2 (EeePC) hardware support based on patches received from Woody at Xandros and forwarded to me by user StewieGriffin at the eeeuser.com forum

Introduction

Currently the Linux Elantech touchpad driver is aware of four different hardware versions unimaginatively called version 1,version 2, version 3 and version 4. Version 1 is found in "older" laptops and uses 4 bytes per packet. Version 2 seems to be introduced with the EeePC and uses 6 bytes per packet, and provides additional features such as position of two fingers, and width of the touch. Hardware version 3 uses 6 bytes per packet (and for 2 fingers the concatenation of two 6 bytes packets) and allows tracking of up to 3 fingers. Hardware version 4 uses 6 bytes per packet, and can combine a status packet with multiple head or motion packets. Hardware version 4 allows tracking up to 5 fingers.

Some Hardware version 3 and version 4 also have a trackpoint which uses a separate packet format. It is also 6 bytes per packet.

The driver tries to support both hardware versions and should be compatible with the Xorg Synaptics touchpad driver and its graphical configuration utilities.

Note that a mouse button is also associated with either the touchpad or the trackpoint when a trackpoint is available. Disabling the Touchpad in xorg (TouchPadOff=0) will also disable the buttons associated with the touchpad.

Additionally the operation of the touchpad can be altered by adjusting the contents of some of its internal registers. These registers are represented by the driver as sysfs entries under /sys/bus/serio/drivers/psmouse/serio? that can be read from and written to.

Currently only the registers for hardware version 1 are somewhat understood. Hardware version 2 seems to use some of the same registers but it is not known whether the bits in the registers represent the same thing or might have changed their meaning.

On top of that, some register settings have effect only when the touchpad is in relative mode and not in absolute mode. As the Linux Elantech touchpad driver always puts the hardware into absolute mode not all information mentioned below can be used immediately. But because there is no freely available Elantech documentation the information is provided here anyway for completeness sake.

Extra knobs

Currently the Linux Elantech touchpad driver provides three extra knobs under /sys/bus/serio/drivers/psmouse/serio? for the user.

- debug

  Turn different levels of debugging ON or OFF.

  By echoing "0" to this file all debugging will be turned OFF.

  Currently a value of "1" will turn on some basic debugging and a value of "2" will turn on packet debugging. For hardware version 1 the default is OFF. For version 2 the default is "1".

  Turning packet debugging on will make the driver dump every packet received to the syslog before processing it. Be warned that this can generate quite a lot of data!

- paritycheck

  Turns parity checking ON or OFF.

  By echoing "0" to this file parity checking will be turned OFF. Any non-zero value will turn it ON. For hardware version 1 the default is ON. For version 2 the default it is OFF.

  Hardware version 1 provides basic data integrity verification by calculating a parity bit for the last 3 bytes of each packet. The driver can check these bits and reject any packet that appears corrupted. Using this knob you can bypass that check.

  Hardware version 2 does not provide the same parity bits. Only some basic data consistency checking can be done. For now checking is disabled by default. Currently even turning it on will do nothing.

- crc_enabled

  Sets crc_enabled to 0/1. The name "crc_enabled" is the official name of this integrity check, even though it is not an actual cyclic redundancy check.

  Depending on the state of crc_enabled, certain basic data integrity verification is done by the driver on hardware version 3 and 4. The driver will reject any packet that appears corrupted. Using this knob, The state of crc_enabled

can be altered with this knob.

Reading the crc_enabled value will show the active value. Echoing "0" or "1" to this file will set the state to "0" or "1".

Differentiating hardware versions

To detect the hardware version, read the version number as param[0].param[1].param[2]:

```
4 bytes version: (after the arrow is the name given in the Dell-provided driver)
02.00.22 => EF013
02.06.00 => EF019
```

In the wild, there appear to be more versions, such as 00.01.64, 01.00.21, 02.00.00, 02.00.04, 02.00.06:

```
6 bytes:
02.00.30 => EF113
02.08.00 => EF023
02.08.XX => EF123
02.0B.00 => EF215
04.01.XX => Scroll_EF051
04.02.XX => EF051
```

In the wild, there appear to be more versions, such as 04.03.01, 04.04.11. There appears to be almost no difference, except for EF113, which does not report pressure/width and has different data consistency checks.

Probably all the versions with param[0] <= 01 can be considered as 4 bytes/firmware 1. The versions < 02.08.00, with the exception of 02.00.30, as 4 bytes/firmware 2. Everything >= 02.08.00 can be considered as 6 bytes.

Hardware version 1

Registers

By echoing a hexadecimal value to a register it contents can be altered.

For example:

```
echo -n 0x16 > reg_10
```

- reg_10:

```
bit   7   6   5   4   3   2   1   0
      B   C   T   D   L   A   S   E

      E: 1 = enable smart edges unconditionally
      S: 1 = enable smart edges only when dragging
      A: 1 = absolute mode (needs 4 byte packets, see reg_11)
      L: 1 = enable drag lock (see reg_22)
      D: 1 = disable dynamic resolution
      T: 1 = disable tapping
      C: 1 = enable corner tap
      B: 1 = swap left and right button
```

- reg_11:

```
bit   7   6   5   4   3   2   1   0
      1   0   0   H   V   1   F   P

      P: 1 = enable parity checking for relative mode
      F: 1 = enable native 4 byte packet mode
      V: 1 = enable vertical scroll area
      H: 1 = enable horizontal scroll area
```

- reg_20:

```
single finger width?
```

- reg_21:

```
scroll area width (small: 0x40 ... wide: 0xff)
```

- reg_22:

```
drag lock time out (short: 0x14 ... long: 0xfe;
                    0xff = tap again to release)
```

- reg_23:

```
tap make timeout?
```

- reg_24:

```
tap release timeout?
```

- reg_25:

```
smart edge cursor speed (0x02 = slow, 0x03 = medium, 0x04 = fast)
```

- reg_26:

```
smart edge activation area width?
```

Native relative mode 4 byte packet format

byte 0:

```
bit   7   6   5   4   3   2   1   0
      c   c  p2  p1   1   M   R   L

      L, R, M = 1 when Left, Right, Middle mouse button pressed
         some models have M as byte 3 odd parity bit
      when parity checking is enabled (reg_11, P = 1):
         p1..p2 = byte 1 and 2 odd parity bit
      c = 1 when corner tap detected
```

byte 1:

```
bit   7   6   5   4   3   2   1   0
     dx7 dx6 dx5 dx4 dx3 dx2 dx1 dx0

      dx7..dx0 = x movement;   positive = right, negative = left
      byte 1 = 0xf0 when corner tap detected
```

byte 2:

```
bit   7   6   5   4   3   2   1   0
     dy7 dy6 dy5 dy4 dy3 dy2 dy1 dy0

      dy7..dy0 = y movement;   positive = up,    negative = down
```

byte 3:

```
parity checking enabled (reg_11, P = 1):

   bit   7   6   5   4   3   2   1   0
         w   h  n1  n0  ds3 ds2 ds1 ds0

         normally:
            ds3..ds0 = scroll wheel amount and direction
                       positive = down or left
                       negative = up or right
         when corner tap detected:
            ds0 = 1 when top right corner tapped
            ds1 = 1 when bottom right corner tapped
            ds2 = 1 when bottom left corner tapped
            ds3 = 1 when top left corner tapped
         n1..n0 = number of fingers on touchpad
            only models with firmware 2.x report this, models with
            firmware 1.x seem to map one, two and three finger taps
            directly to L, M and R mouse buttons
         h = 1 when horizontal scroll action
         w = 1 when wide finger touch?

otherwise (reg_11, P = 0):

   bit   7   6   5   4   3   2   1   0
        ds7 ds6 ds5 ds4 ds3 ds2 ds1 ds0

         ds7..ds0 = vertical scroll amount and direction
                    negative = up
                    positive = down
```

Native absolute mode 4 byte packet format

EF013 and EF019 have a special behaviour (due to a bug in the firmware?), and when 1 finger is touching, the first 2 position reports must be discarded. This counting is reset whenever a different number of fingers is reported.

byte 0:

```
firmware version 1.x:
```

```
   bit   7   6   5   4   3   2   1   0
         D   U  p1  p2   1  p3   R   L

         L, R = 1 when Left, Right mouse button pressed
         p1..p3 = byte 1..3 odd parity bit
         D, U = 1 when rocker switch pressed Up, Down

firmware version 2.x:

   bit   7   6   5   4   3   2   1   0
        n1  n0  p2  p1   1  p3   R   L

         L, R = 1 when Left, Right mouse button pressed
         p1..p3 = byte 1..3 odd parity bit
         n1..n0 = number of fingers on touchpad
```

byte 1:

```
firmware version 1.x:

   bit   7   6   5   4   3   2   1   0
         f   0  th  tw  x9  x8  y9  y8

         tw = 1 when two finger touch
         th = 1 when three finger touch
         f  = 1 when finger touch

firmware version 2.x:

   bit   7   6   5   4   3   2   1   0
         .   .   .   .  x9  x8  y9  y8
```

byte 2:

```
bit   7   6   5   4   3   2   1   0
     x7  x6  x5  x4  x3  x2  x1  x0

      x9..x0 = absolute x value (horizontal)
```

byte 3:

```
bit   7   6   5   4   3   2   1   0
     y7  y6  y5  y4  y3  y2  y1  y0

      y9..y0 = absolute y value (vertical)
```

Hardware version 2

Registers

By echoing a hexadecimal value to a register it contents can be altered.

For example:

```
echo -n 0x56 > reg_10
```

- reg_10:

```
   bit   7   6   5   4   3   2   1   0
         0   1   0   1   0   1   D   0

         D: 1 = enable drag and drop
```

- reg_11:

```
   bit   7   6   5   4   3   2   1   0
         1   0   0   0   S   0   1   0

         S: 1 = enable vertical scroll
```

- reg_21:

```
   unknown (0x00)
```

- reg_22:

```
   drag and drop release time out (short: 0x70 ... long 0x7e;
                           0x7f = never i.e. tap again to release)
```

Native absolute mode 6 byte packet format

Parity checking and packet re-synchronization

There is no parity checking, however some consistency checks can be performed.

For instance for EF113:

```
SA1= packet[0];
A1 = packet[1];
B1 = packet[2];
SB1= packet[3];
C1 = packet[4];
D1 = packet[5];
if( (((SA1 & 0x3C) != 0x3C) && ((SA1 & 0xC0) != 0x80)) || // check Byte 1
    (((SA1 & 0x0C) != 0x0C) && ((SA1 & 0xC0) == 0x80)) || // check Byte 1 (one finger pressed)
    (((SA1 & 0xC0) != 0x80) && (( A1 & 0xF0) != 0x00)) || // check Byte 2
    (((SB1 & 0x3E) != 0x38) && ((SA1 & 0xC0) != 0x80)) || // check Byte 4
    (((SB1 & 0x0E) != 0x08) && ((SA1 & 0xC0) == 0x80)) || // check Byte 4 (one finger pressed)
    (((SA1 & 0xC0) != 0x80) && (( C1 & 0xF0) != 0x00))  ) // check Byte 5
       // error detected
```

For all the other ones, there are just a few constant bits:

```
if( ((packet[0] & 0x0C) != 0x04) ||
    ((packet[3] & 0x0f) != 0x02) )
       // error detected
```

In case an error is detected, all the packets are shifted by one (and packet[0] is discarded).

One/Three finger touch

byte 0:

```
bit   7   6   5   4   3   2   1   0
      n1  n0  w3  w2   .   .   R   L

      L, R = 1 when Left, Right mouse button pressed
      n1..n0 = number of fingers on touchpad
```

byte 1:

```
bit   7   6   5   4   3   2   1   0
      p7  p6  p5  p4 x11 x10 x9  x8
```

byte 2:

```
bit   7   6   5   4   3   2   1   0
      x7  x6  x5  x4  x3  x2  x1  x0

      x11..x0 = absolute x value (horizontal)
```

byte 3:

```
bit   7   6   5   4   3   2   1   0
      n4  vf  w1  w0   .   .   .  b2

      n4 = set if more than 3 fingers (only in 3 fingers mode)
      vf = a kind of flag ? (only on EF123, 0 when finger is over one
           of the buttons, 1 otherwise)
      w3..w0 = width of the finger touch (not EF113)
      b2 (on EF113 only, 0 otherwise), b2.R.L indicates one button pressed:
             0 = none
             1 = Left
             2 = Right
             3 = Middle (Left and Right)
             4 = Forward
             5 = Back
             6 = Another one
             7 = Another one
```

byte 4:

```
bit   7   6   5   4   3   2   1   0
     p3  p1  p2  p0  y11 y10 y9  y8

      p7..p0 = pressure (not EF113)
```

byte 5:

```
bit   7   6   5   4   3   2   1   0
     y7  y6  y5  y4  y3  y2  y1  y0

      y11..y0 = absolute y value (vertical)
```

Two finger touch

Note that the two pairs of coordinates are not exactly the coordinates of the two fingers, but only the pair of the lower-left and upper-right coordinates. So the actual fingers might be situated on the other diagonal of the square defined by these two points.

byte 0:

```
bit   7   6   5   4   3   2   1   0
     n1  n0  ay8 ax8  .   .   R   L

      L, R = 1 when Left, Right mouse button pressed
      n1..n0 = number of fingers on touchpad
```

byte 1:

```
bit   7   6   5   4   3   2   1   0
     ax7 ax6 ax5 ax4 ax3 ax2 ax1 ax0

      ax8..ax0 = lower-left finger absolute x value
```

byte 2:

```
bit   7   6   5   4   3   2   1   0
     ay7 ay6 ay5 ay4 ay3 ay2 ay1 ay0

      ay8..ay0 = lower-left finger absolute y value
```

byte 3:

```
bit   7   6   5   4   3   2   1   0
      .   .  by8 bx8  .   .   .   .
```

byte 4:

```
bit   7   6   5   4   3   2   1   0
     bx7 bx6 bx5 bx4 bx3 bx2 bx1 bx0

      bx8..bx0 = upper-right finger absolute x value
```

byte 5:

```
bit   7   6   5   4   3   2   1   0
     by7 by8 by5 by4 by3 by2 by1 by0

      by8..by0 = upper-right finger absolute y value
```

Hardware version 3

Registers

- reg_10:

```
     bit   7   6   5   4   3   2   1   0
           0   0   0   0   R   F   T   A

          A: 1 = enable absolute tracking
          T: 1 = enable two finger mode auto correct
          F: 1 = disable ABS Position Filter
          R: 1 = enable real hardware resolution
```

Native absolute mode 6 byte packet format

1 and 3 finger touch shares the same 6-byte packet format, except that 3 finger touch only reports the position of the center of all three fingers.

Firmware would send 12 bytes of data for 2 finger touch.

Note on debounce: In case the box has unstable power supply or other electricity issues, or when number of finger changes, F/W would send "debounce packet" to inform driver that the hardware is in debounce status. The debouce packet has the following signature:

```
byte 0: 0xc4
byte 1: 0xff
byte 2: 0xff
byte 3: 0x02
byte 4: 0xff
byte 5: 0xff
```

When we encounter this kind of packet, we just ignore it.

One/Three finger touch

byte 0:

```
bit   7   6   5   4   3   2   1   0
     n1  n0  w3  w2   0   1   R   L

     L, R = 1 when Left, Right mouse button pressed
     n1..n0 = number of fingers on touchpad
```

byte 1:

```
bit   7   6   5   4   3   2   1   0
     p7  p6  p5  p4 x11 x10  x9  x8
```

byte 2:

```
bit   7   6   5   4   3   2   1   0
     x7  x6  x5  x4  x3  x2  x1  x0

     x11..x0 = absolute x value (horizontal)
```

byte 3:

```
bit   7   6   5   4   3   2   1   0
      0   0  w1  w0   0   0   1   0

      w3..w0 = width of the finger touch
```

byte 4:

```
bit   7   6   5   4   3   2   1   0
     p3  p1  p2  p0  y11 y10 y9  y8

     p7..p0 = pressure
```

byte 5:

```
bit   7   6   5   4   3   2   1   0
     y7  y6  y5  y4  y3  y2  y1  y0

     y11..y0 = absolute y value (vertical)
```

Two finger touch

The packet format is exactly the same for two finger touch, except the hardware sends two 6 byte packets. The first packet contains data for the first finger, the second packet has data for the second finger. So for two finger touch a total of 12 bytes are sent.

Hardware version 4

Registers

- reg_07:

```
bit   7   6   5   4   3   2   1   0
      0   0   0   0   0   0   0   A

      A: 1 = enable absolute tracking
```

Native absolute mode 6 byte packet format

v4 hardware is a true multitouch touchpad, capable of tracking up to 5 fingers. Unfortunately, due to PS/2's limited bandwidth, its packet format is rather complex.

Whenever the numbers or identities of the fingers changes, the hardware sends a status packet to indicate how many and which fingers is on touchpad, followed by head packets or motion packets. A head packet contains data of finger id, finger position (absolute x, y values), width, and pressure. A motion packet contains two fingers' position delta.

For example, when status packet tells there are 2 fingers on touchpad, then we can expect two following head packets. If the finger status doesn't change, the following packets would be motion packets, only sending delta of finger position, until we receive a status packet.

One exception is one finger touch. when a status packet tells us there is only one finger, the hardware would just send head packets afterwards.

Status packet

byte 0:

```
bit   7   6   5   4   3   2   1   0
      .   .   .   .   0   1   R   L

      L, R = 1 when Left, Right mouse button pressed
```

byte 1:

```
bit   7   6   5   4   3   2   1   0
      .   .   . ft4 ft3 ft2 ft1 ft0

      ft4 ft3 ft2 ft1 ft0 ftn = 1 when finger n is on touchpad
```

byte 2:

```
not used
```

byte 3:

```
bit   7   6   5   4   3   2   1   0
      .   .   .   1   0   0   0   0

      constant bits
```

byte 4:

```
bit   7   6   5   4   3   2   1   0
      p   .   .   .   .   .   .   .

      p = 1 for palm
```

byte 5:

```
not used
```

Head packet

byte 0:

```
bit   7   6   5   4   3   2   1   0
     w3  w2  w1  w0   0   1   R   L

     L, R = 1 when Left, Right mouse button pressed
     w3..w0 = finger width (spans how many trace lines)
```

byte 1:

```
bit   7   6   5   4   3   2   1   0
     p7  p6  p5  p4 x11 x10  x9  x8
```

byte 2:

```
bit   7   6   5   4   3   2   1   0
     x7  x6  x5  x4  x3  x2  x1  x0

     x11..x0 = absolute x value (horizontal)
```

byte 3:

```
bit   7   6   5   4   3   2   1   0
    id2 id1 id0   1   0   0   0   1

    id2..id0 = finger id
```

byte 4:

```
bit   7   6   5   4   3   2   1   0
     p3  p1  p2  p0  y11 y10 y9  y8

     p7..p0 = pressure
```

byte 5:

```
bit   7   6   5   4   3   2   1   0
     y7  y6  y5  y4  y3  y2  y1  y0

     y11..y0 = absolute y value (vertical)
```

Motion packet

byte 0:

```
bit   7   6   5   4   3   2   1   0
```

```
      id2 id1 id0   w   0   1   R   L

      L, R = 1 when Left, Right mouse button pressed
      id2..id0 = finger id
      w = 1 when delta overflows (> 127 or < -128), in this case
      firmware sends us (delta x / 5) and (delta y  / 5)
```

byte 1:

```
  bit   7   6   5   4   3   2   1   0
       x7  x6  x5  x4  x3  x2  x1  x0

       x7..x0 = delta x (two's complement)
```

byte 2:

```
  bit   7   6   5   4   3   2   1   0
       y7  y6  y5  y4  y3  y2  y1  y0

       y7..y0 = delta y (two's complement)
```

byte 3:

```
  bit   7   6   5   4   3   2   1   0
      id2 id1 id0   1   0   0   1   0

      id2..id0 = finger id
```

byte 4:

```
  bit   7   6   5   4   3   2   1   0
       x7  x6  x5  x4  x3  x2  x1  x0

       x7..x0 = delta x (two's complement)
```

byte 5:

```
  bit   7   6   5   4   3   2   1   0
       y7  y6  y5  y4  y3  y2  y1  y0

       y7..y0 = delta y (two's complement)

       byte 0 ~ 2 for one finger
       byte 3 ~ 5 for another
```

Trackpoint (for Hardware version 3 and 4)

Registers

No special registers have been identified.

Native relative mode 6 byte packet format

Status Packet

byte 0:

```
  bit   7   6   5   4   3   2   1   0
        0   0  sx  sy   0   M   R   L
```

byte 1:

```
  bit   7   6   5   4   3   2   1   0
      ~sx   0   0   0   0   0   0   0
```

byte 2:

```
  bit   7   6   5   4   3   2   1   0
      ~sy   0   0   0   0   0   0   0
```

byte 3:

```
  bit   7   6   5   4   3   2   1   0
        0   0 ~sy ~sx   0   1   1   0
```

byte 4:

```
  bit   7   6   5   4   3   2   1   0
       x7  x6  x5  x4  x3  x2  x1  x0
```

byte 5:

```
bit   7   6   5   4   3   2   1   0
     y7  y6  y5  y4  y3  y2  y1  y0


      x and y are written in two's complement spread
          over 9 bits with sx/sy the relative top bit and
          x7..x0 and y7..y0 the lower bits.
      ~sx is the inverse of sx, ~sy is the inverse of sy.
      The sign of y is opposite to what the input driver
          expects for a relative movement
```