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

Contents

1. Introduction

Currently the Linux Elantech touchpad driver is aware of two different
hardware versions unimaginatively called version 1 and version 2. 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.

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

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.

///

2. Extra knobs

Currently the Linux Elantech touchpad driver provides two 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.

   It is not known yet whether hardware version 2 provides the same parity
   bits. Hence checking is disabled by default. Currently even turning it on
   will do nothing.

///

3. Hardware version 1

3.1 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?
```

3.2 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
```

3.3 Native absolute mode 4 byte packet format

```
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)
```

```
////////////////////////////////////////////////////////////////////////////////
```

```
4. Hardware version 2
   ==================


4.1 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)


4.2 Native absolute mode 6 byte packet format
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


4.2.1 One finger touch
      ~~~~~~~~~~~~~~~~


byte 0:

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

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

byte 1:

   bit   7   6   5   4   3   2   1   0
         .   .   .   .   .  x10 x9  x8
```

```
byte 2:

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

         x10..x0 = absolute x value (horizontal)

byte 3:

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

byte 4:

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

byte 5:

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

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

4.2.2 Two finger touch
~~~~~~~~~~~~~~~~

```
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 = numbers 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 = first 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 = first 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 = second 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 = second finger absolute y value
```