

Sentelic Touchpad

Copyright: © 2002-2011 Sentelic Corporation.
Last update: Dec-07-2011

Finger Sensing Pad Intellimouse Mode (scrolling wheel, 4th and 5th buttons)

A. MSID 4: Scrolling wheel mode plus Forward page(4th button) and Backward page (5th button)

1. Set sample rate to 200;
2. Set sample rate to 200;
3. Set sample rate to 80;
4. Issuing the "Get device ID" command (0xF2) and waits for the response;
5. FSP will respond 0x04.

```
Packet 1
Bit 7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0
BYTE |-----|BYTE |-----|BYTE |-----|BYTE |-----|
1   |Y|X|Y|x|1|M|R|L| 2   |X|X|X|X|X|X|X|X| 3   |Y|Y|Y|Y|Y|Y|Y|Y| 4   | | |B|F|W|W|W|W|
    |-----|      |-----|      |-----|      |-----|
```

Byte 1: Bit7 => Y overflow
Bit6 => X overflow
Bit5 => Y sign bit
Bit4 => X sign bit
Bit3 => 1
Bit2 => Middle Button, 1 is pressed, 0 is not pressed.
Bit1 => Right Button, 1 is pressed, 0 is not pressed.
Bit0 => Left Button, 1 is pressed, 0 is not pressed.
Byte 2: X Movement(9-bit 2's complement integers)
Byte 3: Y Movement(9-bit 2's complement integers)
Byte 4: Bit3~Bit0 => the scrolling wheel's movement since the last data report.
valid values, -8 ~ +7
Bit4 => 1 = 4th mouse button is pressed, Forward one page.
0 = 4th mouse button is not pressed.
Bit5 => 1 = 5th mouse button is pressed, Backward one page.
0 = 5th mouse button is not pressed.

B. MSID 6: Horizontal and Vertical scrolling

- Set bit 1 in register 0x40 to 1

FSP replaces scrolling wheel's movement as 4 bits to show horizontal and vertical scrolling.

```
Packet 1
Bit 7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0
BYTE |-----|BYTE |-----|BYTE |-----|BYTE |-----|
1   |Y|X|Y|x|1|M|R|L| 2   |X|X|X|X|X|X|X|X| 3   |Y|Y|Y|Y|Y|Y|Y|Y| 4   | | |B|F|r|l|u|d|
    |-----|      |-----|      |-----|      |-----|
```

Byte 1: Bit7 => Y overflow
Bit6 => X overflow
Bit5 => Y sign bit
Bit4 => X sign bit
Bit3 => 1
Bit2 => Middle Button, 1 is pressed, 0 is not pressed.
Bit1 => Right Button, 1 is pressed, 0 is not pressed.
Bit0 => Left Button, 1 is pressed, 0 is not pressed.
Byte 2: X Movement(9-bit 2's complement integers)
Byte 3: Y Movement(9-bit 2's complement integers)
Byte 4: Bit0 => the Vertical scrolling movement downward.
Bit1 => the Vertical scrolling movement upward.
Bit2 => the Horizontal scrolling movement leftward.
Bit3 => the Horizontal scrolling movement rightward.
Bit4 => 1 = 4th mouse button is pressed, Forward one page.
0 = 4th mouse button is not pressed.
Bit5 => 1 = 5th mouse button is pressed, Backward one page.
0 = 5th mouse button is not pressed.

C. MSID 7

FSP uses 2 packets (8 Bytes) to represent Absolute Position. so we have PACKET NUMBER to identify packets.

If PACKET NUMBER is 0, the packet is Packet 1. If PACKET NUMBER is 1, the packet is Packet 2. Please count this number in program

MSID6 special packet will be enable at the same time when enable MSID 7.

Absolute position for STL3886-G0

1. Set bit 2 or 3 in register 0x40 to 1
2. Set bit 6 in register 0x40 to 1

```
Packet 1 (ABSOLUTE POSITION)
Bit 7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0
BYTE |-----|BYTE|-----|BYTE|-----|BYTE|-----|
  1  |0|1|V|1|1|M|R|L|  2  |X|X|X|X|X|X|X|X|  3  |Y|Y|Y|Y|Y|Y|Y|Y|  4  |r|l|d|u|X|X|Y|Y|
    |-----|      |-----|      |-----|      |-----|
```

Byte 1: Bit7~Bit6 => 00, Normal data packet
=> 01, Absolute coordination packet
=> 10, Notify packet
Bit5 => valid bit
Bit4 => 1
Bit3 => 1
Bit2 => Middle Button, 1 is pressed, 0 is not pressed.
Bit1 => Right Button, 1 is pressed, 0 is not pressed.
Bit0 => Left Button, 1 is pressed, 0 is not pressed.
Byte 2: X coordinate (xpos[9:2])
Byte 3: Y coordinate (ypos[9:2])
Byte 4: Bit1~Bit0 => Y coordinate (xpos[1:0])
Bit3~Bit2 => X coordinate (ypos[1:0])
Bit4 => scroll up
Bit5 => scroll down
Bit6 => scroll left
Bit7 => scroll right

```
Notify Packet for G0
Bit 7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0
BYTE |-----|BYTE|-----|BYTE|-----|BYTE|-----|
  1  |1|0|0|1|1|M|R|L|  2  |C|C|C|C|C|C|C|C|  3  |M|M|M|M|M|M|M|M|  4  |0|0|0|0|0|0|0|0|
    |-----|      |-----|      |-----|      |-----|
```

Byte 1: Bit7~Bit6 => 00, Normal data packet
=> 01, Absolute coordination packet
=> 10, Notify packet
Bit5 => 0
Bit4 => 1
Bit3 => 1
Bit2 => Middle Button, 1 is pressed, 0 is not pressed.
Bit1 => Right Button, 1 is pressed, 0 is not pressed.
Bit0 => Left Button, 1 is pressed, 0 is not pressed.
Byte 2: Message Type => 0x5A (Enable/Disable status packet)
Mode Type => 0xA5 (Normal/Icon mode status)
Byte 3: Message Type => 0x00 (Disabled)
=> 0x01 (Enabled)
Mode Type => 0x00 (Normal)
=> 0x01 (Icon)
Byte 4: Bit7~Bit0 => Don't Care

Absolute position for STL3888-Ax

```
Packet 1 (ABSOLUTE POSITION)
Bit 7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0
BYTE |-----|BYTE|-----|BYTE|-----|BYTE|-----|
  1  |0|1|V|A|1|L|0|1|  2  |X|X|X|X|X|X|X|X|  3  |Y|Y|Y|Y|Y|Y|Y|Y|  4  |x|x|y|y|X|X|Y|Y|
    |-----|      |-----|      |-----|      |-----|
```

Byte 1: Bit7~Bit6 => 00, Normal data packet
=> 01, Absolute coordination packet
=> 10, Notify packet
=> 11, Normal data packet with on-pad click
Bit5 => Valid bit, 0 means that the coordinate is invalid or finger up.
When both fingers are up, the last two reports have zero valid bit.
Bit4 => arc
Bit3 => 1
Bit2 => Left Button, 1 is pressed, 0 is released.
Bit1 => 0
Bit0 => 1
Byte 2: X coordinate (xpos[9:2])
Byte 3: Y coordinate (ypos[9:2])
Byte 4: Bit1~Bit0 => Y coordinate (xpos[1:0])
Bit3~Bit2 => X coordinate (ypos[1:0])
Bit5~Bit4 => y1_g

```

        Bit7~Bit6 => x1_g

Packet 2 (ABSOLUTE POSITION)
Bit 7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0
BYTE |-----|BYTE |-----|BYTE |-----|BYTE |-----|
1   |0|1|V|A|1|R|1|0| 2   |X|X|X|X|X|X|X|X| 3   |Y|Y|Y|Y|Y|Y|Y|Y| 4   |x|x|y|y|X|X|Y|Y|
    |-----|      |-----|      |-----|      |-----|

Byte 1: Bit7~Bit6 => 00, Normal data packet
        => 01, Absolute coordinates packet
        => 10, Notify packet
        => 11, Normal data packet with on-pad click
        Bit5 => Valid bit, 0 means that the coordinate is invalid or finger up.
                When both fingers are up, the last two reports have zero valid
                bit.
        Bit4 => arc
        Bit3 => 1
        Bit2 => Right Button, 1 is pressed, 0 is released.
        Bit1 => 1
        Bit0 => 0
Byte 2: X coordinate (xpos[9:2])
Byte 3: Y coordinate (ypos[9:2])
Byte 4: Bit1~Bit0 => Y coordinate (xpos[1:0])
        Bit3~Bit2 => X coordinate (ypos[1:0])
        Bit5~Bit4 => y2_g
        Bit7~Bit6 => x2_g

```

```

Notify Packet for STL3888-Ax
Bit 7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0
BYTE |-----|BYTE |-----|BYTE |-----|BYTE |-----|
1   |1|0|1|P|1|M|R|L| 2   |C|C|C|C|C|C|C|C| 3   |0|0|F|F|0|0|0|i| 4   |r|l|d|u|0|0|0|0|
    |-----|      |-----|      |-----|      |-----|

Byte 1: Bit7~Bit6 => 00, Normal data packet
        => 01, Absolute coordinates packet
        => 10, Notify packet
        => 11, Normal data packet with on-pad click
        Bit5 => 1
        Bit4 => when in absolute coordinates mode (valid when EN_PKT_GO is 1):
                0: left button is generated by the on-pad command
                1: left button is generated by the external button
        Bit3 => 1
        Bit2 => Middle Button, 1 is pressed, 0 is not pressed.
        Bit1 => Right Button, 1 is pressed, 0 is not pressed.
        Bit0 => Left Button, 1 is pressed, 0 is not pressed.
Byte 2: Message Type => 0xB7 (Multi Finger, Multi Coordinate mode)
Byte 3: Bit7~Bit6 => Don't care
        Bit5~Bit4 => Number of fingers
        Bit3~Bit1 => Reserved
        Bit0 => 1: enter gesture mode; 0: leaving gesture mode
Byte 4: Bit7 => scroll right button
        Bit6 => scroll left button
        Bit5 => scroll down button
        Bit4 => scroll up button
                * Note that if gesture and additional button (Bit4~Bit7)
                happen at the same time, the button information will not
                be sent.
        Bit3~Bit0 => Reserved

```

Sample sequence of Multi-finger, Multi-coordinate mode:

notify packet (valid bit == 1), abs pkt 1, abs pkt 2, abs pkt 1, abs pkt 2, ..., notify packet (valid bit == 0)

Absolute position for STL3888-B0

```

Packet 1 (ABSOLUTE POSITION)
Bit 7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0
BYTE |-----|BYTE |-----|BYTE |-----|BYTE |-----|
1   |0|1|V|F|1|0|R|L| 2   |X|X|X|X|X|X|X|X| 3   |Y|Y|Y|Y|Y|Y|Y|Y| 4   |r|l|u|d|X|X|Y|Y|
    |-----|      |-----|      |-----|      |-----|

Byte 1: Bit7~Bit6 => 00, Normal data packet
        => 01, Absolute coordinates packet
        => 10, Notify packet
        => 11, Normal data packet with on-pad click
        Bit5 => Valid bit, 0 means that the coordinate is invalid or finger up.
                When both fingers are up, the last two reports have zero valid
                bit.
        Bit4 => finger up/down information. 1: finger down, 0: finger up.
        Bit3 => 1

```

Bit2 => finger index, 0 is the first finger, 1 is the second finger.
 Bit1 => Right Button, 1 is pressed, 0 is not pressed.
 Bit0 => Left Button, 1 is pressed, 0 is not pressed.
 Byte 2: X coordinate (xpos[9:2])
 Byte 3: Y coordinate (ypos[9:2])
 Byte 4: Bit1~Bit0 => Y coordinate (xpos[1:0])
 Bit3~Bit2 => X coordinate (ypos[1:0])
 Bit4 => scroll down button
 Bit5 => scroll up button
 Bit6 => scroll left button
 Bit7 => scroll right button

Packet 2 (ABSOLUTE POSITION)

Bit 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BYTE -----				BYTE -----				BYTE -----				BYTE -----											
1 0 1 V F 1 1 R L				2 X X X X X X X X				3 Y Y Y Y Y Y Y Y				4 r l u d X X Y Y											
-----				-----				-----				-----											

Byte 1: Bit7~Bit6 => 00, Normal data packet
 => 01, Absolute coordination packet
 => 10, Notify packet
 => 11, Normal data packet with on-pad click
 Bit5 => Valid bit, 0 means that the coordinate is invalid or finger up.
 When both fingers are up, the last two reports have zero valid bit.
 Bit4 => finger up/down information. 1: finger down, 0: finger up.
 Bit3 => 1
 Bit2 => finger index, 0 is the first finger, 1 is the second finger.
 Bit1 => Right Button, 1 is pressed, 0 is not pressed.
 Bit0 => Left Button, 1 is pressed, 0 is not pressed.
 Byte 2: X coordinate (xpos[9:2])
 Byte 3: Y coordinate (ypos[9:2])
 Byte 4: Bit1~Bit0 => Y coordinate (xpos[1:0])
 Bit3~Bit2 => X coordinate (ypos[1:0])
 Bit4 => scroll down button
 Bit5 => scroll up button
 Bit6 => scroll left button
 Bit7 => scroll right button

Notify Packet for STL3888-B0:

Bit 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BYTE -----				BYTE -----				BYTE -----				BYTE -----																			
1 1 0 1 P 1 1 M R L				2 C C C C C C C C				3 0 0 F F 0 0 0 i				4 r l u d 0 0 0 0																			
-----				-----				-----				-----																			

Byte 1: Bit7~Bit6 => 00, Normal data packet
 => 01, Absolute coordination packet
 => 10, Notify packet
 => 11, Normal data packet with on-pad click
 Bit5 => 1
 Bit4 => when in absolute coordinates mode (valid when EN_PKT_GO is 1):
 0: left button is generated by the on-pad command
 1: left button is generated by the external button
 Bit3 => 1
 Bit2 => Middle Button, 1 is pressed, 0 is not pressed.
 Bit1 => Right Button, 1 is pressed, 0 is not pressed.
 Bit0 => Left Button, 1 is pressed, 0 is not pressed.
 Byte 2: Message Type => 0xB7 (Multi Finger, Multi Coordinate mode)
 Byte 3: Bit7~Bit6 => Don't care
 Bit5~Bit4 => Number of fingers
 Bit3~Bit1 => Reserved
 Bit0 => 1: enter gesture mode; 0: leaving gesture mode
 Byte 4: Bit7 => scroll right button
 Bit6 => scroll left button
 Bit5 => scroll up button
 Bit4 => scroll down button
 * Note that if gesture and additional button(Bit4~Bit7)
 happen at the same time, the button information will not
 be sent.
 Bit3~Bit0 => Reserved

Sample sequence of Multi-finger, Multi-coordinate mode:

notify packet (valid bit = 1), abs pkt 1, abs pkt 2, abs pkt 1, abs pkt 2, ..., notify packet (valid bit = 0)

Absolute position for STL3888-Cx and STL3888-Dx

Single Finger, Absolute Coordinate Mode (SFAC)

Bit 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
-------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

BYTE	1	2	3	4
	0 1 0 P 1 M R L	X X X X X X X X	Y Y Y Y Y Y Y Y	r l B F X X Y Y

Byte 1: Bit7~Bit6 => 00, Normal data packet
=> 01, Absolute coordinates packet
=> 10, Notify packet

Bit5 => Coordinate mode(always 0 in SFAC mode):
0: single-finger absolute coordinates (SFAC) mode
1: multi-finger, multiple coordinates (MFMC) mode

Bit4 => 0: The LEFT button is generated by on-pad command (OPC)
1: The LEFT button is generated by external button
Default is 1 even if the LEFT button is not pressed.

Bit3 => Always 1, as specified by PS/2 protocol.
Bit2 => Middle Button, 1 is pressed, 0 is not pressed.
Bit1 => Right Button, 1 is pressed, 0 is not pressed.
Bit0 => Left Button, 1 is pressed, 0 is not pressed.

Byte 2: X coordinate (xpos[9:2])
Byte 3: Y coordinate (ypos[9:2])
Byte 4: Bit1~Bit0 => Y coordinate (xpos[1:0])
Bit3~Bit2 => X coordinate (ypos[1:0])
Bit4 => 4th mouse button(forward one page)
Bit5 => 5th mouse button(backward one page)
Bit6 => scroll left button
Bit7 => scroll right button

Multi Finger, Multiple Coordinates Mode (MFMC):

Bit 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
BYTE -----	BYTE -----	BYTE -----	BYTE -----
1 0 1 1 P 1 F R L	2 X X X X X X X X	3 Y Y Y Y Y Y Y Y	4 r l B F X X Y Y

Byte 1: Bit7~Bit6 => 00, Normal data packet
=> 01, Absolute coordination packet
=> 10, Notify packet

Bit5 => Coordinate mode (always 1 in MFMC mode):
0: single-finger absolute coordinates (SFAC) mode
1: multi-finger, multiple coordinates (MFMC) mode

Bit4 => 0: The LEFT button is generated by on-pad command (OPC)
1: The LEFT button is generated by external button
Default is 1 even if the LEFT button is not pressed.

Bit3 => Always 1, as specified by PS/2 protocol.
Bit2 => Finger index, 0 is the first finger, 1 is the second finger.
If bit 1 and 0 are all 1 and bit 4 is 0, the middle external button is pressed.

Bit1 => Right Button, 1 is pressed, 0 is not pressed.
Bit0 => Left Button, 1 is pressed, 0 is not pressed.

Byte 2: X coordinate (xpos[9:2])
Byte 3: Y coordinate (ypos[9:2])
Byte 4: Bit1~Bit0 => Y coordinate (xpos[1:0])
Bit3~Bit2 => X coordinate (ypos[1:0])
Bit4 => 4th mouse button(forward one page)
Bit5 => 5th mouse button(backward one page)
Bit6 => scroll left button
Bit7 => scroll right button

When one of the two fingers is up, the device will output four consecutive MFMC#0 report packets with zero X and Y to represent 1st finger is up or four consecutive MFMC#1 report packets with zero X and Y to represent that the 2nd finger is up. On the other hand, if both fingers are up, the device will output four consecutive single-finger, absolute coordinate(SFAC) packets with zero X and Y.

Notify Packet for STL3888-Cx/Dx:

Bit 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
BYTE -----	BYTE -----	BYTE -----	BYTE -----
1 1 0 0 P 1 M R L	2 C C C C C C C C	3 0 0 F F 0 0 0 i	4 r l u d 0 0 0 0

Byte 1: Bit7~Bit6 => 00, Normal data packet
=> 01, Absolute coordinates packet
=> 10, Notify packet

Bit5 => Always 0

Bit4 => 0: The LEFT button is generated by on-pad command(OPC)
1: The LEFT button is generated by external button
Default is 1 even if the LEFT button is not pressed.

Bit3 => 1
Bit2 => Middle Button, 1 is pressed, 0 is not pressed.
Bit1 => Right Button, 1 is pressed, 0 is not pressed.
Bit0 => Left Button, 1 is pressed, 0 is not pressed.

Byte 2: Message type:
0xba => gesture information

```

0xc0 => one finger hold-rotating gesture
Byte 3: The first parameter for the received message:
0xba => gesture ID (refer to the 'Gesture ID' section)
0xc0 => region ID
Byte 4: The second parameter for the received message:
0xba => N/A
0xc0 => finger up/down information

```

Sample sequence of Multi-finger, Multi-coordinates mode:

notify packet (valid bit == 1), MFMC packet 1 (byte 1, bit 2 == 0), MFMC packet 2 (byte 1, bit 2 == 1), MFMC packet 1, MFMC packet 2, ..., notify packet (valid bit == 0)

That is, when the device is in MFMC mode, the host will receive interleaved absolute coordinate packets for each finger.

FSP Enable/Disable packet

```

Bit 7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0      7 6 5 4 3 2 1 0
BYTE |-----|BYTE |-----|BYTE |-----|BYTE |-----|
1   |Y|X|0|0|1|M|R|L| 2   |0|1|0|1|1|0|1|E| 3   | | | | | | | | 4   | | | | | | | |
    |-----|      |-----|      |-----|      |-----|

```

FSP will send out enable/disable packet when FSP receive PS/2 enable/disable command. Host will receive the packet which Middle, Right, Left button will be set. The packet only use byte 0 and byte 1 as a pattern of original packet. Ignore the other bytes of the packet.

```

Byte 1: Bit7 => 0, Y overflow
        Bit6 => 0, X overflow
        Bit5 => 0, Y sign bit
        Bit4 => 0, X sign bit
        Bit3 => 1
        Bit2 => 1, Middle Button
        Bit1 => 1, Right Button
        Bit0 => 1, Left Button
Byte 2: Bit7~1 => (0101101b)
        Bit0 => 1 = Enable
           0 = Disable
Byte 3: Don't care
Byte 4: Don't care (MOUSE ID 3, 4)
Byte 5~8: Don't care (Absolute packet)

```

PS/2 Command Set

FSP supports basic PS/2 commanding set and modes, refer to following URL for details about PS/2 commands:

<http://www.computer-engineering.org/ps2mouse/>

Programming Sequence for Determining Packet Parsing Flow

1. Identify FSP by reading device ID(0x00) and version(0x01) register
2. For FSP version < STL3888 Cx, determine number of buttons by reading the 'test mode status' (0x20) register:

```

buttons = reg[0x20] & 0x30

if buttons == 0x30 or buttons == 0x20:
    # two/four buttons
    Refer to 'Finger Sensing Pad PS/2 Mouse Intellimouse'
    section A for packet parsing detail(ignore byte 4, bit ~ 7)
elif buttons == 0x10:
    # 6 buttons
    Refer to 'Finger Sensing Pad PS/2 Mouse Intellimouse'
    section B for packet parsing detail
elif buttons == 0x00:
    # 6 buttons
    Refer to 'Finger Sensing Pad PS/2 Mouse Intellimouse'
    section A for packet parsing detail

```

3. For FSP version >= STL3888 Cx:

Refer to 'Finger Sensing Pad PS/2 Mouse Intellimouse' section A for packet parsing detail (ignore byte 4, bit ~ 7)

Programming Sequence for Register Reading/Writing

Register inversion requirement:

Following values needed to be inverted(the '~' operator in C) before being sent to FSP:

0xe8, 0xe9, 0xee, 0xf2, 0xf3 and 0xff.

Register swapping requirement:

Following values needed to have their higher 4 bits and lower 4 bits being swapped before being sent to FSP:

10, 20, 40, 60, 80, 100 and 200.

Register reading sequence:

1. send 0xf3 PS/2 command to FSP;
2. send 0x66 PS/2 command to FSP;
3. send 0x88 PS/2 command to FSP;
4. send 0xf3 PS/2 command to FSP;
5. if the register address being to read is not required to be inverted(refer to the 'Register inversion requirement' section), goto step 6
 - a. send 0x68 PS/2 command to FSP;
 - b. send the inverted register address to FSP and goto step 8;
6. if the register address being to read is not required to be swapped(refer to the 'Register swapping requirement' section), goto step 7
 - a. send 0xcc PS/2 command to FSP;
 - b. send the swapped register address to FSP and goto step 8;
7. send 0x66 PS/2 command to FSP;
 - a. send the original register address to FSP and goto step 8;
8. send 0xe9(status request) PS/2 command to FSP;
9. the 4th byte of the response read from FSP should be the requested register value(?? indicates don't care byte):

```
host: 0xe9
3888: 0xfa (??) (??) (val)
```

- Note that since the Cx release, the hardware will return 1's complement of the register value at the 3rd byte of status request result:

```
host: 0xe9
3888: 0xfa (??) (~val) (val)
```

Register writing sequence:

1. send 0xf3 PS/2 command to FSP;
2. if the register address being to write is not required to be inverted(refer to the 'Register inversion requirement' section), goto step 3
 - a. send 0x74 PS/2 command to FSP;
 - b. send the inverted register address to FSP and goto step 5;
3. if the register address being to write is not required to be swapped(refer to the 'Register swapping requirement' section), goto step 4
 - a. send 0x77 PS/2 command to FSP;
 - b. send the swapped register address to FSP and goto step 5;
4. send 0x55 PS/2 command to FSP;
 - a. send the register address to FSP and goto step 5;
5. send 0xf3 PS/2 command to FSP;
6. if the register value being to write is not required to be inverted(refer to the 'Register inversion requirement' section), goto step 7
 - a. send 0x47 PS/2 command to FSP;
 - b. send the inverted register value to FSP and goto step 9;
7. if the register value being to write is not required to be swapped(refer to the 'Register swapping requirement' section), goto step 8

- a. send 0x44 PS/2 command to FSP;
 - b. send the swapped register value to FSP and goto step 9;
8. send 0x33 PS/2 command to FSP;
 - a. send the register value to FSP;
9. the register writing sequence is completed.
 - Since the Cx release, the hardware will return 1's complement of the register value at the 3rd byte of status request result. Host can optionally send another 0xe9 (status request) PS/2 command to FSP at the end of register writing to verify that the register writing operation is successful (?? indicates don't care byte):

```
host: 0xe9
3888: 0xfa (??) (~val) (val)
```

Programming Sequence for Page Register Reading/Writing

In order to overcome the limitation of maximum number of registers supported, the hardware separates register into different groups called 'pages.' Each page is able to include up to 255 registers.

The default page after power up is 0x82; therefore, if one has to get access to register 0x8301, one has to use following sequence to switch to page 0x83, then start reading/writing from/to offset 0x01 by using the register read/write sequence described in previous section.

Page register reading sequence:

1. send 0xf3 PS/2 command to FSP;
2. send 0x66 PS/2 command to FSP;
3. send 0x88 PS/2 command to FSP;
4. send 0xf3 PS/2 command to FSP;
5. send 0x83 PS/2 command to FSP;
6. send 0x88 PS/2 command to FSP;
7. send 0xe9(status request) PS/2 command to FSP;
8. the response read from FSP should be the requested page value.

Page register writing sequence:

1. send 0xf3 PS/2 command to FSP;
2. send 0x38 PS/2 command to FSP;
3. send 0x88 PS/2 command to FSP;
4. send 0xf3 PS/2 command to FSP;
5. if the page address being written is not required to be inverted(refer to the 'Register inversion requirement' section), goto step 6
 - a. send 0x47 PS/2 command to FSP;
 - b. send the inverted page address to FSP and goto step 9;
6. if the page address being written is not required to be swapped(refer to the 'Register swapping requirement' section), goto step 7
 - a. send 0x44 PS/2 command to FSP;
 - b. send the swapped page address to FSP and goto step 9;
7. send 0x33 PS/2 command to FSP;
8. send the page address to FSP;
9. the page register writing sequence is completed.

Gesture ID

Unlike other devices which sends multiple fingers' coordinates to host, FSP processes multiple fingers' coordinates internally and convert them into a 8 bits integer, namely 'Gesture ID.' Following is a list of supported gesture IDs:

ID	Description
0x86	2 finger straight up
0x82	2 finger straight down
0x80	2 finger straight right
0x84	2 finger straight left

ID	Description
0x8f	2 finger zoom in
0x8b	2 finger zoom out
0xc0	2 finger curve, counter clockwise
0xc4	2 finger curve, clockwise
0x2e	3 finger straight up
0x2a	3 finger straight down
0x28	3 finger straight right
0x2c	3 finger straight left
0x38	palm

Register Listing

Registers are represented in 16 bits values. The higher 8 bits represent the page address and the lower 8 bits represent the relative offset within that particular page. Refer to the 'Programming Sequence for Page Register Reading/Writing' section for instructions on how to change current page address:

offset	width	default	r/w	name
0x8200	bit7~bit0	0x01	RO	device ID
0x8201	bit7~bit0		RW	version ID 0xc1: STL3888 Ax 0xd0 ~ 0xd2: STL3888 Bx 0xe0 ~ 0xe1: STL3888 Cx 0xe2 ~ 0xe3: STL3888 Dx
0x8202	bit7~bit0	0x01	RO	vendor ID
0x8203	bit7~bit0	0x01	RO	product ID
0x8204	bit3~bit0	0x01	RW	revision ID
0x820b	bit3	1	RO	test mode status 1 0: rotate 180 degree 1: no rotation *only supported by H/W prior to Cx
0x820f	bit2	0	RW	register file page control 1: rotate 180 degree 0: no rotation *supported since Cx
	bit0	0	RW	1 to enable page 1 register files *only supported by H/W prior to Cx
0x8210			RW	system control 1
	bit0	1	RW	Reserved, must be 1
	bit1	0	RW	Reserved, must be 0
	bit4	0	RW	Reserved, must be 0
	bit5	1	RW	register clock gating enable 0: read only, 1: read/write enable (Note that following registers does not require clock gating being enabled prior to write: 05 06 07 08 09 0c 0f 10 11 12 16 17 18 23 2e 40 41 42 43. In addition to that, this bit must be 1 when gesture mode is enabled)
0x8220	bit5~bit4		RO	test mode status number of buttons 11 => 2, lbtn/rbtn 10 => 4, lbtn/rbtn/scru/scrd 01 => 6, lbtn/rbtn/scru/scrd/scrl/scrr 00 => 6, lbtn/rbtn/scru/scrd/fbtn/bbtn *only supported by H/W prior to Cx
0x8231	bit7	0	RW	on-pad command detection on-pad command left button down tag enable 0: disable, 1: enable *only supported by H/W prior to Cx
0x8234	bit4~bit0	0x05	RW	on-pad command control 5 XLO in 0s/4/1, so 03h = 0010.1b = 2.5 (Note that position unit is in 0.5 scanline) *only supported by H/W prior to Cx

bit7	0	RW	on-pad tap zone enable 0: disable, 1: enable *only supported by H/W prior to Cx
0x8235		RW	on-pad command control 6
bit4~bit0	0x1d	RW	XHI in 0s/4/1, so 19h = 1100.1b = 12.5 (Note that position unit is in 0.5 scanline) *only supported by H/W prior to Cx
0x8236		RW	on-pad command control 7
bit4~bit0	0x04	RW	YLO in 0s/4/1, so 03h = 0010.1b = 2.5 (Note that position unit is in 0.5 scanline) *only supported by H/W prior to Cx
0x8237		RW	on-pad command control 8
bit4~bit0	0x13	RW	YHI in 0s/4/1, so 11h = 1000.1b = 8.5 (Note that position unit is in 0.5 scanline) *only supported by H/W prior to Cx
0x8240		RW	system control 5
bit1	0	RW	FSP Intellimouse mode enable 0: disable, 1: enable *only supported by H/W prior to Cx
bit2	0	RW	movement + abs. coordinate mode enable 0: disable, 1: enable (Note that this function has the functionality of bit 1 even when bit 1 is not set. However, the format is different from that of bit 1. In addition, when bit 1 and bit 2 are set at the same time, bit 2 will override bit 1.) *only supported by H/W prior to Cx
bit3	0	RW	abs. coordinate only mode enable 0: disable, 1: enable (Note that this function has the functionality of bit 1 even when bit 1 is not set. However, the format is different from that of bit 1. In addition, when bit 1, bit 2 and bit 3 are set at the same time, bit 3 will override bit 1 and 2.) *only supported by H/W prior to Cx
bit5	0	RW	auto switch enable 0: disable, 1: enable *only supported by H/W prior to Cx
bit6	0	RW	G0 abs. + notify packet format enable 0: disable, 1: enable (Note that the absolute/relative coordinate output still depends on bit 2 and 3. That is, if any of those bit is 1, host will receive absolute coordinates; otherwise, host only receives packets with relative coordinate.) *only supported by H/W prior to Cx
bit7	0	RW	EN_PS2_F2: PS/2 gesture mode 2nd finger packet enable 0: disable, 1: enable *only supported by H/W prior to Cx
0x8243		RW	on-pad control
bit0	0	RW	on-pad control enable 0: disable, 1: enable (Note that if this bit is cleared, bit 3/5 will be ineffective) *only supported by H/W prior to Cx
bit3	0	RW	on-pad fix vertical scrolling enable 0: disable, 1: enable *only supported by H/W prior to Cx
bit5	0	RW	on-pad fix horizontal scrolling enable 0: disable, 1: enable *only supported by H/W prior to Cx
0x8290		RW	software control register 1
bit0	0	RW	absolute coordination mode 0: disable, 1: enable *supported since Cx
bit1	0	RW	gesture ID output 0: disable, 1: enable *supported since Cx
bit2	0	RW	two fingers' coordinates output

				0: disable, 1: enable *supported since Cx
bit3	0	RW		finger up one packet output 0: disable, 1: enable *supported since Cx
bit4	0	RW		absolute coordination continuous mode 0: disable, 1: enable *supported since Cx
bit6~bit5	00	RW		gesture group selection 00: basic 01: suite 10: suite pro 11: advanced *supported since Cx
bit7	0	RW		Bx packet output compatible mode 0: disable, 1: enable *supported since Cx *supported since Cx
0x833d		RW		on-pad command control 1
bit7	1	RW		on-pad command detection enable 0: disable, 1: enable *supported since Cx
0x833e		RW		on-pad command detection
bit7	0	RW		on-pad command left button down tag enable. Works only in H/W based PS/2 data packet mode. 0: disable, 1: enable *supported since Cx