Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

Mouse Sensor Kit (#28560)

The Parallax Mouse Sensor is a module in kit form which, when assembled, provides the tracking functions of an optical mouse. The two-wire serial interface is directly compatible with the Parallax BASIC Stamp® 2 family, the Parallax Propeller, and other microcontrollers.

Features

- Compact module, including illumination, optics, and custom laser-cut base
- "Close-to-the-metal" register-based serial interface for maximum flexibility
- Holes for mounting to other equipment
- Compatible with any BS2-family BASIC Stamp®, the SX, and the Parallax Propeller
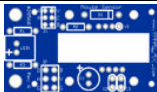- Accommodation for single or dual three-wire (servo-type) interface cables
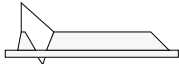
Key Specifications
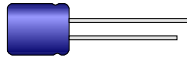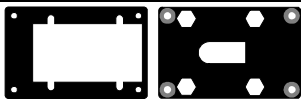
- Power requirements: 5 VDC at 35 mA
- Communication: Two-wire serial (clock and data)
- Logic compatible with 3.3V (using external resistor) and 5V microcontrollers
- Dimensions: 1.80" (45.7mm) L x 1.00" (25.4 mm) W x 0.65" (16.5 mm) H

Application Ideas

- Measuring X and Y displacement on a flat surface
- Detecting vibration in two dimensions over a flat surface

What Comes with the Kit

Part	Parallax No.	Description	Illustration (not to scale)	Quan.
	300-28560	Printed circuit board		1
U1	604-28560	MCS-12086 mouse sensor chip		1
LED1	350-00031	Red T1¾ LED (lens may be red or clear)		1
	350-00032	Right-angle LED holder		1
	721-28560	Clear plastic lens/light guide		1
D1	501-00008	1N5817 Schottky diode		1
R1, R2	150-01021	1K 1/8 W resistor (brown, black, red)		2
R3	150-01022	100Ω 1/8 W resistor (brown, black, brown)		1

Part	Parallax No.	Description	Illustration (not to scale)	Quan.
C1	201-01070	100 µF aluminum electrolytic capacitor (color may vary)		1
C2, C3	201-01063	0.1 µF ceramic capacitors (marked "104")		2
J1	450-00104	2 x 3 header (2mm)		1
	452-00057	2mm shunts		2
J2	450-00105	2x3 header (0.1")		1
	721-00011	Two-piece black Delrin rectangular base set		1
	700-00089	Black plastic snap rivets		4
	710-00028	#2 x ¼" Phillips pan head machine screws		4
	711-00004	#2 hex nuts		4
	713-00018	White nylon spacers		4
Additional Parts for Interfacing				
	805-00002	14" servo extension cable		2
	451-00303	3-pin header		2
	500-00008	2N3904 NPN transistor (marked 2N3904)		1
	500-00003	2N3906 PNP transistor (marked 2N3906)		1
	150-02200	2.2K ¼ W resistors (red, red, red)		3
	150-04710	470Ω ¼ W resistor (yellow, violet, brown)		1

What You Need to Provide

For Assembly

- *Fine-tipped* soldering iron and *small diameter* (1/32") rosin-core solder (lead-free or leaded)
- Wire clippers
- Needle-nosed pliers
- 99% isopropyl alcohol and an old (clean) toothbrush
- Miniature (#0) Phillips screwdriver
- Pointed tweezers
- Eye protection

For Operation

- BASIC Stamp, Propeller, SX, etc.
- Carrier board (e.g. Board of Education, Propeller Demo Board, Propeller Proto Board, etc.)
- One or two servo extension cables with 3-pin headers (e.g. Parallax #805-00011)

Assembly Instructions

Assemble the Circuit Board

1. The printed circuit board is marked on top with the part numbers from the "Part" column in the table above. Here is an illustration of the unpopulated circuit board:
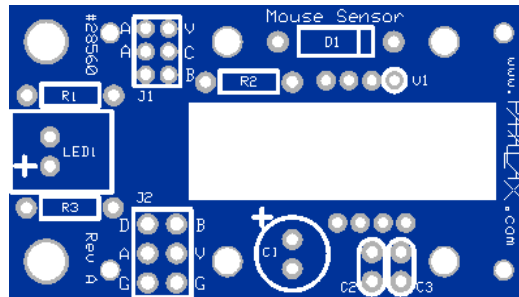
2. Install and solder all the parts, *except* U1 and LED1, as shown below, paying close attention to the polarity of D1 (stripe goes to the right) and of C1 (positive lead – the longer one – goes to the top). Take your time and make sure that all parts are firmly seated against the board. For J1 and J2, solder the diagonal corner pins first and check the seating. If a header is crooked or not all the way in, you have a chance to reheat the joints and make adjustments. Once these headers are seated correctly, you can solder the remaining pins. Here's what the board will look like:
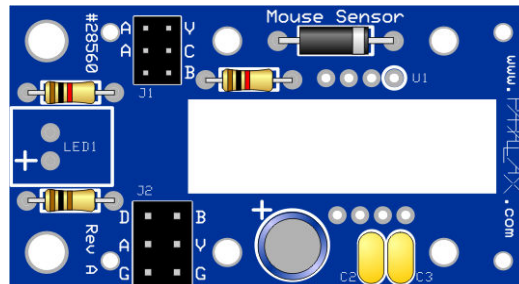
3. Take the LED and insert it into the right-angle holder, being sure to position the longer, positive lead as shown:
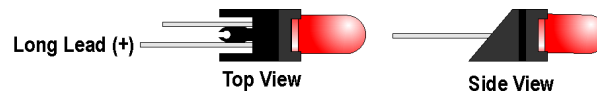
4. With the needle-nosed pliers, grab the leads and pull the LED against the holder until it's firmly seated. Then, still pulling with the pliers, bend the leads sharply downward:
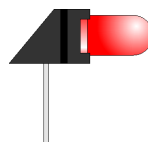
NOTE: The LED is shown in the drawings with a red lens. The LED provided may be red or clear.

5. Now you can solder the LED assembly to the board. Here's what the board will look like:
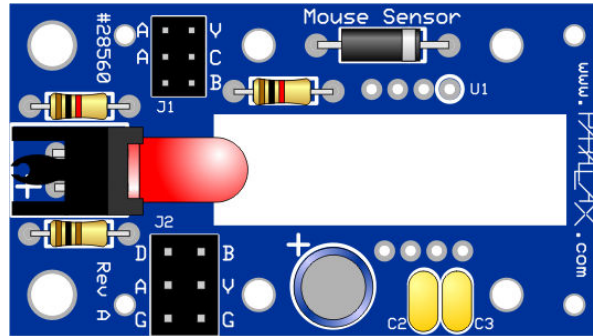
6. At this point, the board is completely assembled, *except for U1, the sensor chip*. Make sure to clip all the leads that protrude through the board close to, but not completely flush with, the bottom of the board.

7. Use the isopropyl alcohol and the toothbrush to clean any residual flux from the board. This is the last chance you'll get to clean it, so do a thorough job.

Assemble the Base

NOTE: The mechanical assembly involves some very tiny parts that can easily become lost if dropped. Work in a location such that, if you do drop something, you will be able to find it again. Working on top of a folded towel can help to avoid parts bouncing and getting lost.

1. There are two Delrin base pieces, a top piece and a bottom piece. The bottom of the bottom piece may or may not have four engraved recesses in it:

Engraved Recess

Top Piece   **Bottom Piece (Bottom View)**
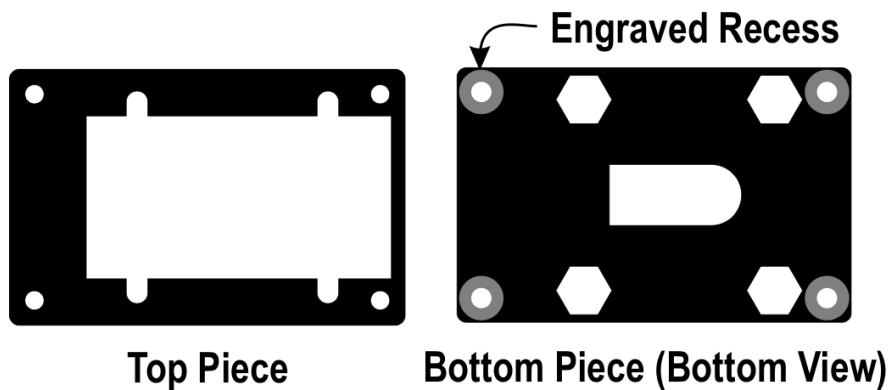
2. Peel off any protective film that may be in place over these pieces (one side), and poke out any "hanging chads" that may remain in some of the holes.

3. Stack the top piece over the bottom piece, maintaining the orientations shown above, so that you can still see the engraved recesses, if there are any. The hexagonal holes in the bottom piece will align with the slotted cutouts in the top piece. Insert the black plastic snap rivets into the corner holes, so that the heads rest in the engraved recesses:

TOP PIECE
BOTTOM PIECE

4. Firmly press the heads of the snap rivets into place so they rest against the recesses created for them. Their prongs will then spread to hold the two pieces together:

TOP PIECE
BOTTOM PIECE

5. Place the base assembly rivet-head-side-down on the bench, and drop the clear plastic lens-and-light guide into the rectangular recess created for it:
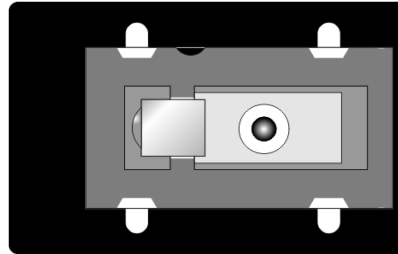
6. (See note below for an alternative to steps 6 and 7.) Place the assembled circuit board over the base assembly, so that the little nibs from the rivets fit into the corner holes of the circuit board. The end of the LED should be very near or just touching the lens assembly:
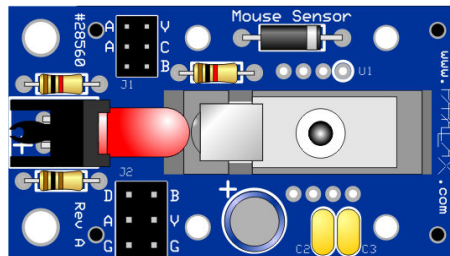
7. Now comes the tedious part. Using the tweezers, slip a nylon spacer between the circuit board and the base assembly, so that it is centered on one of the four small holes. Install a #2 screw from the top, through the circuit board, spacer, and base assembly. Take one of the hex nuts, and insert it into the pocket in the bottom of the base assembly. While holding it in place with your thumb, start the screw into it with a Phillips screwdriver. *Do no tighten.* Repeat for the other three spacers, screws, and nuts. Then tighten all four screws. This is what the assembly will look like, top and bottom:
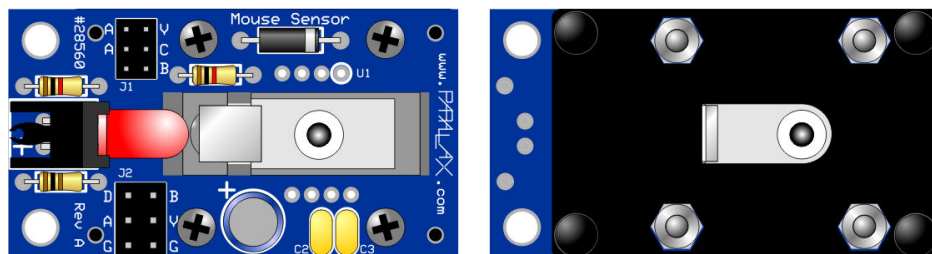
NOTE: As an alternative to steps 6 and 7 above, you can first center the spacers over the holes in the base, then position the circuit board on top of these, and simply drop the screws through the holes. The nuts can then be installed from the bottom, one at a time, and the screws tightened.

Back to Soldering

The last step in the assembly process is to solder in the sensor chip. It was delayed until this point, so that it will mate properly with the lens. Here's what to do:

1. On the bottom of the chip, there is a small circle of clear yellow tape covering a hole. With the tweezers, very carefully remove this tape. *From this point onward, be very careful not to let any foreign substances enter the hole.*

2. Position the chip so that its pin #1 (indicated by a very small and subtle dimple) aligns with the circled pad on the circuit board. Insert the chip into the lead holes:
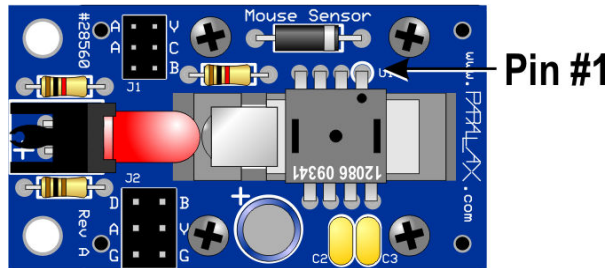
3. Press the chip into the board as far as you can. It will probably touch the lens assembly, which is good.

4. *Making sure the chip is level with the board*, solder one of its corner pins from the top side of the board.

5. While applying finger pressure to the chip, remelt the solder at the corner you've just soldered to make sure the chip is seated as far as it will go. Check again to make sure the chip is still level. It may take several tries, but it's important to get this right.

6. Solder the remaining seven pins from the top of the board.

7. You're finished! Well, almost. You've still got two shunts to install, but we'll get to that in the next section.

Quick Start Circuit (BASIC Stamp)
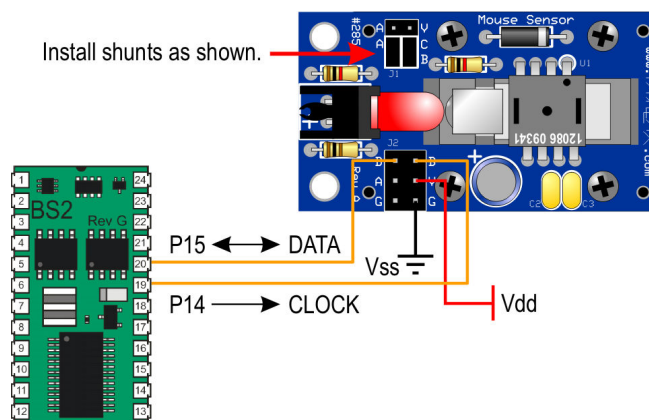
NOTE: Board of Education users can simply connect using two extension cables from the onboard servo headers. *Make sure that servo power is jumpered to Vdd and not Vin.*

Quick Start Circuit (Propeller)
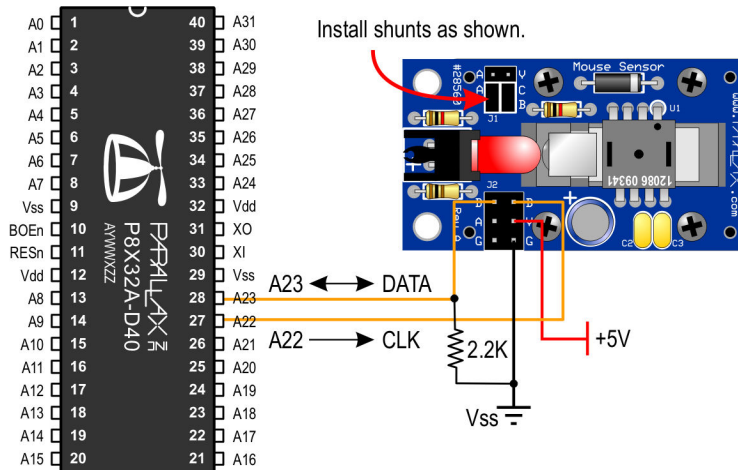
NOTE: *Do not omit the 2.2K resistor*. It is required in order to limit the input voltage to the Propeller.

Connecting and Testing

Once the Mouse Sensor powers up, the red LED should come on. For a further indication that it's working, place the Mouse Sensor on a flat surface and keep it still for a second or two. While keeping an eye on the LED, move it slightly. The LED should brighten just a little while the Mouse Sensor is moving, then dim a little when it stops. This indicates that the Mouse Sensor is detecting motion.

BASIC Stamp 2 Series

To test the Mouse Sensor further with a BASIC Stamp, wire it as shown above. Then upload the program **mouse_monitor.bs2**, shown at the end of this document and downloadable from the Mouse Sensor product page (search "28560" at www.parallax.com). When run, a debug window will pop up, and you should see a display that looks something like this:
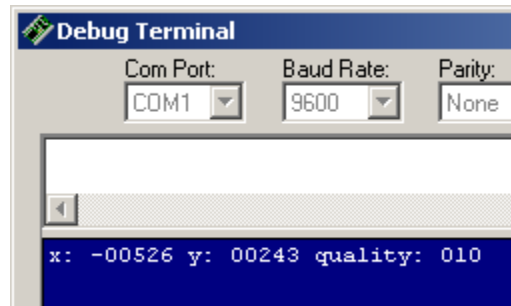
The program accumulates the sensor's X and Y displacement data to show the current position. It also displays a "quality" figure, which indicates how textured the surface is that the sensor is looking at. The higher the number, the better.

You can also monitor the mouse position graphically, using the program **mouse_monitor.exe**, downloadable from the Mouse sensor product page. You will have to make a small change to **mouse_monitor.bs2** to do so. Just comment out the line that reads **#DEFINE USE_DEBUG**:

```
'#DEFINE USE_DEBUG       'Comment this out to use 38400 baud, instead of DEBUG.
```

Then upload the PBASIC program, and start the **.exe**. Once the program finds the Mouse Sensor output, you will see a display that looks like this:
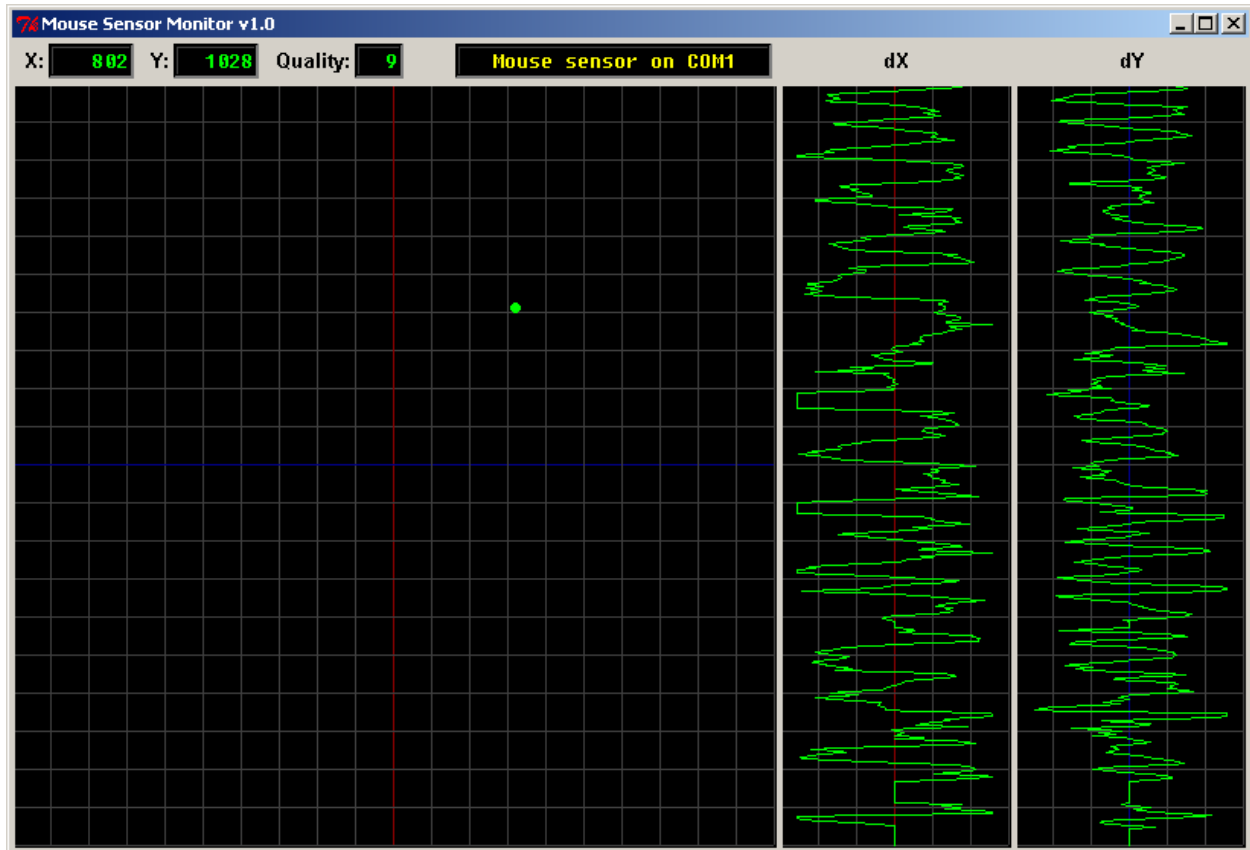
The green dot keeps track of the sensor's current X and Y position. The **dX** and **dY** strip charts track the sensor's instantaneous change in position. If you move the sensor too fast, a message that says "OVERFLOW" will display. The background grid spacing is 250 sensor pixels. The default resolution for the PBASIC program is about 500 d.p.i., which works out to 1/2" of travel per division.

One thing that quickly becomes obvious from using this program is that, while the Mouse Sensor is very good at tracking relative motion, it does accumulate small errors when keeping track of absolute position. To demonstrate this, make a small mark at the sensor's current location, and reset the BASIC Stamp to zero the position. Then move the sensor around on the surface, and return to the starting point. Is the reported position (0, 0)?

Propeller

To test the Mouse Sensor with the Propeller chip, wire it as shown above. Then upload the program **MouseSensorMonitor.spin** shown at the end of this document and downloadable as an archive from the Mouse Sensor product page. It makes use of the **MouseSensor** object, which can also be found in the Propeller Object Exchange (obex.parallax.com). It is designed to be used with the **mouse_monitor.exe** program, as shown above. You will find it to be a lot more responsive with the Propeller than with the BASIC Stamp.

Advanced Three-wire Connection

The Mouse Sensor includes circuitry to harvest its operating power from the clock line. This makes it possible to interface to the Mouse Sensor with a three-wire servo-type extension cable. Because of the Mouse Sensor's current requirements, however, some additional circuitry is required at the host end, as described below. First, however, you will need to change the shunt configuration on the mouse sensor to the following:
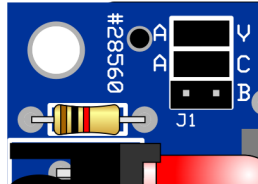
This connects the **A** input on the interface header to both the sensor chip's clock line and to the power-harvesting circuitry.

BASIC Stamp 2 Family

Here's how to connect the BASIC Stamp for three-wire (servo cable) operation, using the additional parts that come with the kit:
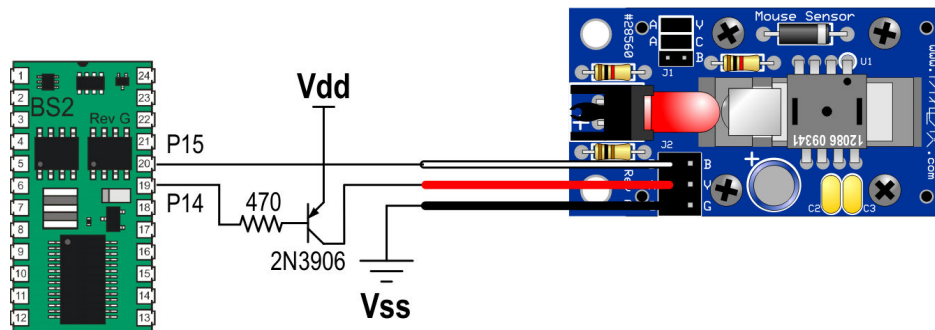
Using this circuit entails a change in the program, since the clock is now inverted from what it was. This is easily accomplished by commenting out the line that says **#DEFINE NEG_CLK**:

```
'#DEFINE NEG_CLK          'Comment this out to use a the multiplexed clock and Vdd line.
```

You will notice that the program gives a much snappier performance with this setting, too. That's because the sensor chip requires a negative-going clock. To provide such a clock, the PBASIC program must resort to bit-banging the clock and data lines, which can be quite slow. But, with the above circuitry, it can communicate with a positive-going clock, and that can be done with the much faster **SHIFTIN** and **SHIFTOUT** instructions.

Propeller

The three-wire Propeller circuitry is a little more complicated than that of the BASIC Stamp. This is because the Propeller outputs do not rise high enough to turn off a PNP transistor powered from +5V. Therefore, an additional transistor is required to do the job. Here's the circuit:
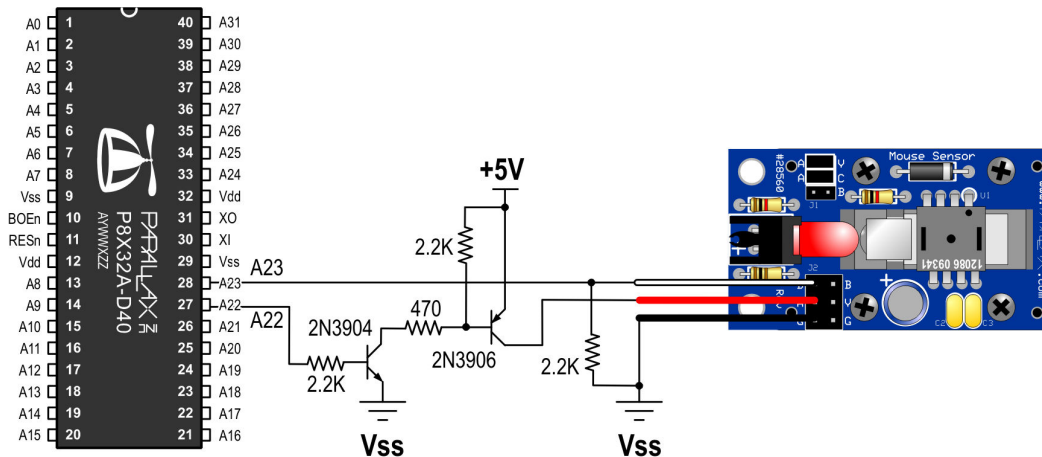
The Spin program requires no changes to accommodate this new circuit. The reason is that the clock signal is inverted twice.

Resources and Downloads

You may download free example and demo programs from the Mouse Sensor product page (search "28560" at www.parallax.com). and from the Propeller Object Exchange (obex.parallax.com).

Device Information

Theory of Operation

The mouse sensor chip is actually a tiny camera that is continuously snapping pictures, comparing each with the one before to detect movement. It works best on surfaces that have a visible texture. Texture provides features that it can recognize from one snapshot to the next.

Communication with the chip takes place by reading and writing its internal registers. The most important of these are listed below:

Address	Type	Description	Range
$00	Read/Write	Operation Mode bits: 7: Set to 1 to reset chip; 0 (default) to operate. 6: Set to 1 for power down; 0 (default) to run.	
$02	Read-only	DX: Amount of change in X direction since last poll.	-128 ($80) to 127 ($7F)
$03	Read-only	DY: Amount of change in Y direction since last poll.	-128 ($80) to 127 ($7F)
$04	Read-only	Image quality: the higher the better.	0 ($00) to 255 ($FF)
$16	Read-only	Status bits: 7: Set to 1 if motion occurred since last poll. 4: Set to 1 if DY register overflowed since last poll. 3: Set to 1 if DX register overflowed since last poll. 0: Set to 1 for 500 dpi; 0 for 1000 dpi.	
$1B	Read/Write	Configuration bits: 7: Set to 1 for 500 dpi; 0 (default) for 1000 dpi.	

A typical flowchart for operating the chip is as follows:
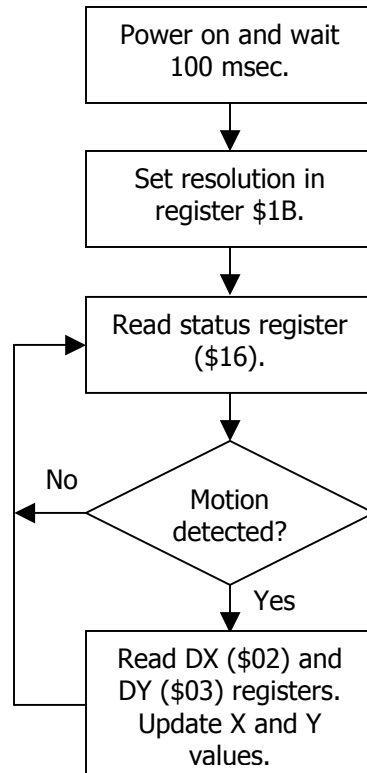
Communication with the chip takes place via two-wire (clock and data) serial I/O. The clock line is normally high and pulses low to transfer data bits in or out of the chip.

Reading Data from the Sensor Registers

Data reads take place by setting the data line to an output, then sending the register address as eight bits, most significant bit first, with bit 7 cleared to zero. Then the data line is switched to an input, and eight bits of data are clocked out of the chip, most significant bit first:
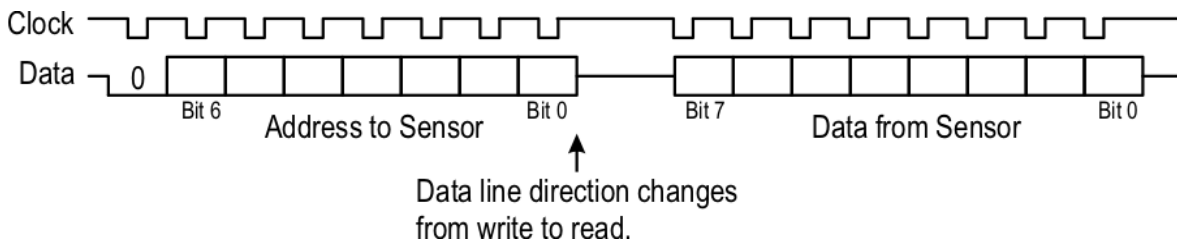
Writing Data to the Sensor Registers

Data writes take place by setting the data line to an output, then sending the register address as eight bits, most significant bit first, with bit 7 set to 1. Then eight data bits are clocked into the chip, most significant bit first.
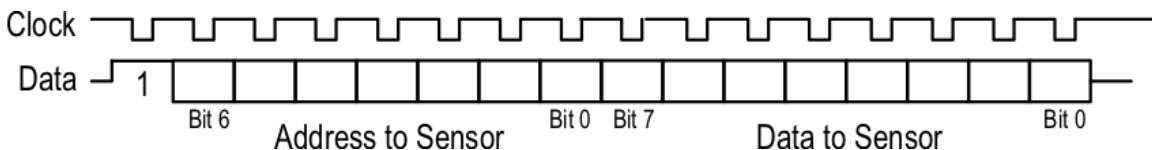
Module Specifications

Symbol	Quantity	Conditions	Min.	Typ.	Max.	Units
V_{dd}	Supply Voltage		4.1	5.0	5.5	V
I_{dd}	Supply Current			35		mA
V_{IL}	Input low voltage	Estimate only: value TBD			0.7	V
V_{IH}	Input high voltage	Estimate only: value TBD	2.0			V
V_{OL}	Output low voltage			0.0		V
V_{OH}	Output high voltage	V_{dd} = 5 V, no load			5.0	V
		V_{dd} = 5 V, 1.5 mA load			3.5	V
F_{CLK}	Clock frequency				2	MHz

Main Header Pin Definitions

Pin	Name	Function
D	Data I/O	Serial Data Input and Output
A	Clk A	Serial Clock Input (Option A), and Optional Power (+5V)
G (2 pins)	Gnd	Ground (Vss = 0 V)
V	Vdd	Power input, if separate from Clk A (+5V)
B	Clk B	Serial Clock Input (Option B)

Options Header Pin (Shunt) Configurations

Pin Pair	Configuration
B — C	Connect Clk B to sensor chip's clock input.
A — C	Connect Clk A to sensor chip's clock input.
A — V	Connect Clk A to power harvesting circuitry, with a 1K pull-down resistor to Vss.
A — [unmarked]	No connection. Used to park an unused shunt.
B — [unmarked]	No connection. Used to park an unused shunt.

Connection Diagrams

Here are the Mouse Sensor's available connection configurations. For details on the "Single Cable" connection, consult the "Advanced Three-wire Connection" section above. When power is provided on the clock line, any reasonably fast high-side driver circuit capable of sourcing 5 volts and at least 35 mA should work. It is not necessary for such a driver to have current sinking capability: a stiff pull-down is provided on the Mouse Sensor module itself.
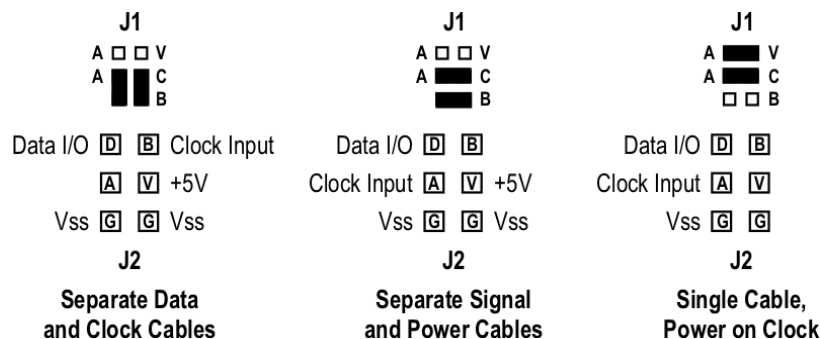
Mouse Sensor Dimensions

0.125"

0.75"

1.00"

1.80"

0.125"

Mounting holes: 0.125" dia.

0.65"

1.55"

0.25"

Mouse Sensor Schematic

+5V

D!
1N5817

R2
1K

U1

1

Vreg
Vdd
Vss
/LED

SDIO
SCLK

MCS12086

C2
0.1 µF

C1
47 µF

C3
0.1 µF

V   C B

J1

A

R1
1K

+5V

LED1

R3
100R

B   V G

J2

D   A G
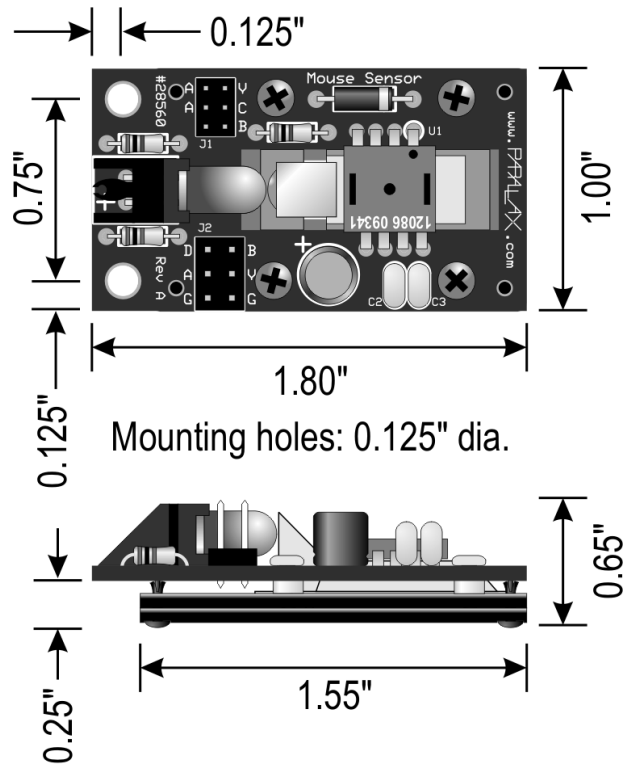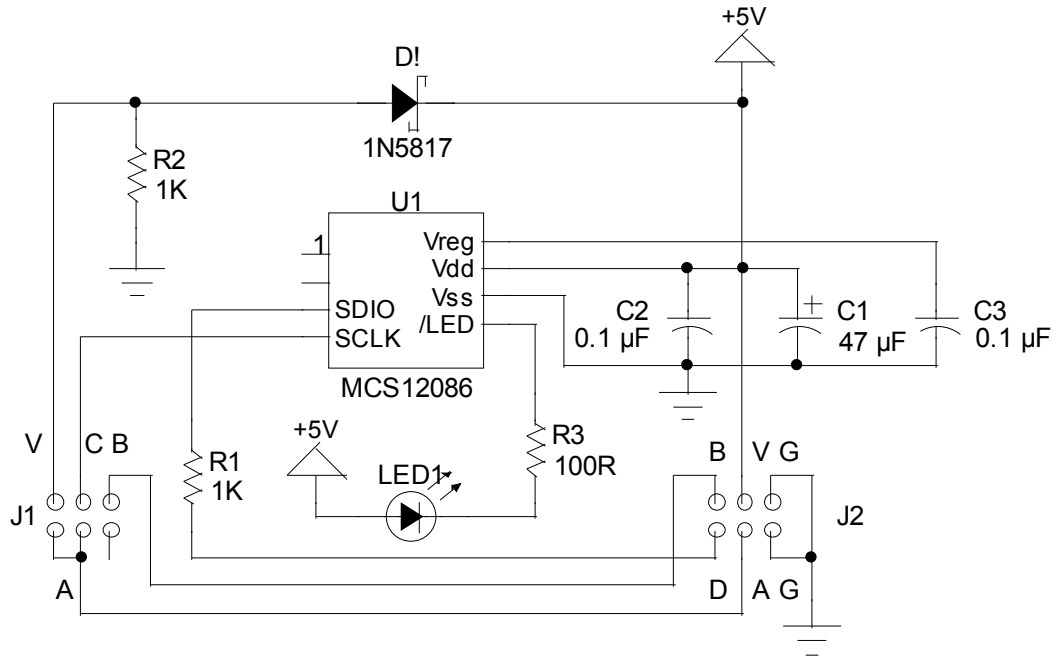
Source Code

BASIC Stamp 2 Program

This .bs2 program is available for download from the Mouse Sensor product page; search "28560" at www.parallax.com.

```
' ========================================================================
'
'   File...... mouse_monitor.bs2
'   Purpose... Monitors data coming from Mouse Sensor (#28560).
'   Author.... Parallax
'   E-mail.... support@parallax.com
'   Started... 24 Feb 2010
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' ========================================================================

' -----[ Program Description ]--------------------------------------------

' This program monitors the queries the Parallax Mouse Sensor Module
' and outputs the data to the serial port.

' -----[ Configuration Constants ]----------------------------------------

#DEFINE USE_500DPI      'Comment this out to use 1000 dpi, instead of 500 dpi.
#DEFINE USE_DEBUG       'Comment this out to use 38400 baud, instead of DEBUG.
#DEFINE NEG_CLK         'Comment this out to use a the multiplexed clock and Vdd line.
#DEFINE DO_XYQ_ONLY     'Comment this out to dump all the registers, not just X,Y,Q.

' -----[ I/O Definitions ]------------------------------------------------

sclk  PIN 14            'Serial clock pin to mouse sensor.
sdio  PIN 15            'Serial data I/O pin to mosue sensor.

' -----[ Constants ]------------------------------------------------------

'Set the baud rate to 9600 if using DEBUG; else set it to 38400.

#IF (USING_DEBUG) #THEN
  #SELECT $STAMP
    #CASE BS2, BS2E, BS2PE : baud CON 84
    #CASE BS2SX, BS2P : baud CON 240
    #CASE BS2PX : baud CON 396
  #ENDSELECT
#ELSE
  #SELECT $stamp
    #CASE BS2, BS2E, BS2PE : baud CON 6
    #CASE BS2SX, BS2P : baud CON 45
    #CASE BS2PX : baud CON 121
  #ENDSELECT
#ENDIF

'Mouse sensor register addresses.

DY    CON $02           'Data register for current change in Y location.
DX    CON $03           'Data register for current change in X location.
QLTY  CON $04           'Data register for current image quality value.
STAT  CON $16           'Status register.
```

```
CONF   CON $1B          'Configuration register.

LORES CON $80          'Value to write to CONF for 500 dpi resolution.
CHNG  CON $80          'Bitmask for STAT to see if position changed.
OFLOW CON $18          'Bitmask for STAT to detect X/Y overflow.
NEG   CON $80          'Sign bit for DX and DY.


' -----[ Variables ]----------------------------------------------------

addr  VAR Byte         'Address value for mouse sensor register.
i     VAR Byte         'General counter.
dat   VAR Byte         'Data value to/from mouse sensor register.
sd    VAR Byte         'Scratch register.
q     VAR Byte         'Quality value from mouse sensor.
ovfl  VAR Byte         'Overflow counter.

x     VAR Word         'Current cummulative X position.
y     VAR Word         'Current cummulative Y position.

' -----[ Program ]------------------------------------------------------

'Initialize clock, depending on polarity.

#IF (NEG_CLK) #THEN
  HIGH sclk
#ELSE
  LOW sclk
#ENDIF

PAUSE 100              'Wait for mouse sensor to come out of reset.

#IF (USE_500DPI) #THEN
  addr = CONF          'Change resolution to 500 dpi.
  dat = LORES
  GOSUB WriteAddr
#ENDIF

#IF (USE_DEBUG) #THEN
  DEBUG CLS            'Clear the screen if using DEBUG window.
#ENDIF

'Main program loop.

DO
  #IF (DO_XYQ_ONLY) #THEN
    GOSUB DumpXYQ      'Use this to monitor X, Y, and Quality only.
  #ELSE
    GOSUB DumpALL      'Use this to dump all registers.
  #ENDIF
LOOP

' -----[ Subroutines ]--------------------------------------------------

' DumpXYQ outputs X, Y, and Quality data to the programming port, for use with
' either DEBUG or an external program.

DumpXYQ:

  addr = STAT
  GOSUB ReadAddr
  #IF (USE_DEBUG) #THEN
    IF (dat & CHNG = 0) THEN
```

```
        DEBUG HOME, "x: ", SDEC5 x, " y: ", SDEC5 y, " quality: ", DEC3 q
        RETURN
      ELSEIF (dat & OFLOW) THEN
        ovfl = 10
      ENDIF
    #ELSE
      IF (dat & CHNG = 0) THEN
        SEROUT 16, baud, ["x", SDEC x, " y", SDEC y, " q", DEC q, CR]
        RETURN
      ELSEIF (dat & OFLOW) THEN
        SEROUT 16, baud, ["x", SDEC x, " y", SDEC y, " q999", CR]
      ENDIF
    #ENDIF
    addr = DX
    GOSUB ReadAddr
    x = x + dat
    IF (dat & NEG) THEN x = x + $ff00
    addr = DY
    GOSUB ReadAddr
    y = y + dat
    IF (dat & NEG) THEN y = y + $ff00
    addr = QLTY
    GOSUB ReadAddr
    q = dat
    #IF (USE_DEBUG) #THEN
      DEBUG HOME, "x: ", SDEC5 x, " y: ", SDEC5 y, " quality: ", DEC3 q
      IF (ovfl) THEN
        ovfl = ovfl - 1
        DEBUG " OVERFLOW"
      ENDIF
      DEBUG CLREOL
    #ELSE
      SEROUT 16, baud, ["x", SDEC x, " y", SDEC y, " q", DEC q, CR]
    #ENDIF
    RETURN

' DumpAll outputs the contents of all the sensor chip's registers.

DumpAll:

  SEROUT 16, baud, [HOME]
  FOR addr = 0 TO $7f
    GOSUB ReadAddr
    IF (addr & 15 = 0) THEN SEROUT 16, baud, [HEX2 addr, ": "]
    SEROUT 16, baud, [HEX2 dat]
    IF (addr & 15 = 15) THEN
      SEROUT 16, baud, [CR]
    ELSEIF (addr & 7 = 7) THEN
      SEROUT 16, baud, ["   "]
    ELSE
      SEROUT 16, baud, [" "]
    ENDIF
  NEXT
  SEROUT 16, baud, [CR]
  RETURN

' ReadAddr reads a sensor chip register.
'   Inputs:  addr = address ($00 - $7F) to read.
'   Outputs: dat  = contents of the addressed register.

ReadAddr:

  #IF (NEG_CLK) #THEN
```

```
      sd = addr
      GOSUB WriteNeg
      GOSUB ReadNeg
  #ELSE
      SHIFTOUT sdio, sclk, MSBFIRST, [addr\8]
      INPUT sdio
      SHIFTIN sdio, sclk, MSBPOST, [dat\8]
  #ENDIF
    RETURN

WriteAddr:

' WriteAddr writes data to a sensor chip register.
'   Inputs:  addr = address ($00 - $7F) to write.
'            dat  = data to write to the addressed register.

  #IF (NEG_CLK) #THEN
      sd = addr | $80
      GOSUB WriteNeg
      sd = dat
      GOSUB WriteNeg
  #ELSE
      SHIFTOUT sdio, sclk, MSBFIRST, [addr|$80\8, dat\8]
  #ENDIF
    RETURN

WriteNeg:

' WriteNeg simulates the SHIFTOUT instruction, but with a
' negative-going (inverted) clock.
'   Inputs: sd = data to write, MSB first.

  OUTPUT sdio
  FOR i = 0 TO 7
    sdio = sd.BIT7
    PULSOUT sclk, 25
    sd = sd << 1
  NEXT
  INPUT sdio
  RETURN

ReadNeg:

' ReadNeg simulates the SHIFTIN instruction, but with a
' negative-going (inverted) clock.
'   Outputs: dat = data read from serial bus, MSB first.

  FOR i = 0 TO 7
    dat = dat << 1
    PULSOUT sclk, 25
    dat.BIT0 = sdio
  NEXT
  RETURN
```

Propeller Application

Here's the Top Level **MouseSensorMonitor.spin** listing. The referenced objects are included with the archive downloadable from the Mouse Sensor page (search "28560" at www.parallax.com) or from the Propeller Object Exchange (obex.parallax.com).

```
CON

  _clkmode      = xtal1 + pll16x
  _xinfreq      = 5_000_000

  SCK_PIN      = 22          'Change these pin specs as necessary.
  SDA_PIN      = 23

OBJ

  ms    : "MouseSensor"
  sio   : "FullduplexSerial"

PUB start | x, y, q, status

  sio.start(31, 30, 0, 38400)
  ms.start(SCK_PIN, SDA_PIN)
  q~
  repeat
    if ((status := ms.read(ms#STAT)) & ms#CHNG)
      if (status & ms#OFLOW)
        report(x, y, 999)
      report(x += ms.getdx, y += ms.getdy, q := ms.read(ms#QLTY))
    else
      report(x, y, q)

PUB report (x, y, q)

  sio.tx("x")
  sio.dec(x)
  sio.str(string(" y"))
  sio.dec(y)
  sio.str(string(" q"))
  sio.dec(q)
  sio.tx(13)
```