Cody's Robot Optical Motion Sensor #1 (CROMS-1)

This webpage describes an optical motion sensor that I developed for my robotics experiments. It is based on an Agilent/Avago ADNS-2610 Optical Mouse Sensor that was taken from an optical mouse. The sensor was designed to work with a special lens, light guide, and LED to view the surface that the mouse sets upon. This arrangement is not really suitable for my robotics applications (though optical mice have been used successfully on robots as "odometer trailers"). I wanted a compact sensor that could be positioned above the floor so it would not rub against it, yet still be able to optically detect the motion of the robot. Consequently, I developed a printed circuit assembly and an optical assembly that would satisfy these goals.

Background

In order to perform dead reckoning navigation on a robot, some sort of sensor must be installed to relate motion of the robot to a measurable signal. I originally looked at using mechanical encoders taken from a old IBM PS/2 mouse and attaching them to the axels of the drive motors. As the axels rotated, the robot could gain an indication of its rate of movement, hence the distance traveled. Some of the robotics articles that I read pointed out that the use of mechanical encoders for distance measurement were succeptible to error due to slippage of the wheel or track against the surface they were running on.

I happened upon the application note Interface to Optical Mouse Sensor on the Kronos Robotics website. It discusses the use of a sensor from an optical mouse for use as a position-sensing coprocessor on a robot. The idea is that the optical mouse sensor would be pointed at the surface that the robot was running on (A new lens system would be created to allow the sensor to be focused on the surface.). As the sensor is moved, the distance changes would appear as mouse counts in the "x" and "y" directions. With proper calibration, these counts would equate to measures of distance moved by the robot.

Not having access to the GE optical mouse described in the Kronos Robotics application note, I found that the BTC M850 3D Optical Mouse has the same sensor: the Agilent/Avago ADNS-2610. After hacking the BTC M850 mouse to directly access the sensor, I was able to gain experience obtaining motion data (as well as other data) from it. While this demonstrated the feasibility of using the mouse sensor, it did not resolve the issue of creating a new lens system to properly focus the sensor.

I found another very useful source of infomation, related especially to lens system development, in the article "Insect-Inspired Optical-Flow Navigation Sensors" published in the October, 2005 issue of NASA Tech Briefs. The article discusses the feasibility of using a mouse sensor to measure the movement of a Mars probe as it flys a few meters above the terrain. A lot of information is provided to develop a new lens system based on the specifications of the standard mouse lens system. The full article can be obtained from the NASA Tech Briefs website in the Technical Support Packages area, under the Computers/ Electronics (Electronic Components and Systems) category. You'll have to register to access the document, but registration is free. Using my experience with the ADNS-2610 and using the NASA document for additional inspiration, I developed a new printed circuit board assembly for the ADNS-2610 and an optical assembly to hold it. The steps for constructing Cody's Robot Optical Motion Sensor #1 (CROMS-1) are described below.

Bringing the Pieces Together

The components list, see Table 1, consists mainly of relatively common items. You may already have them available in your junk box. The exceptions are the optical mouse (due to the optical sensor) and the "Virtual Lenses" device, obtained from BG Micro. Other optical mice will have the ADNS-2610, so if a BTC M850 3D Optical Mouse cannot be found, another optical mouse that uses the same sensor will be acceptable. The "Virtual Lenses" device, though, is a pretty unique item. Fortunately, at the time of this writing, BG Micro has several hundred of these devices in stock!

The items in the component list marked by * can be obtained from other sources besides the BTC M850 Optical Mouse. This may be a necessary recourse, as the leads on some of these compoents can be rather short. Note that any components used must fit within the space provided within the optical assembly, which is rather tight. In addition to the items specified in the component list, solder, two-part epoxy, and double-sided adhesive tape, are needed to assemble the CROMS-1. Tools that were used in the assembly process are a soldering iron, a hobby knife, a rotary tool with cutting disks and drill bits, a file, and scissors. Materials to fabricate a printed circuit board are also needed.

Table 1: Components List

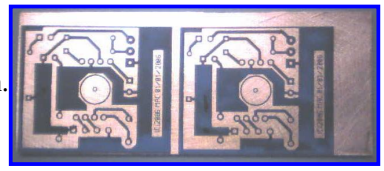
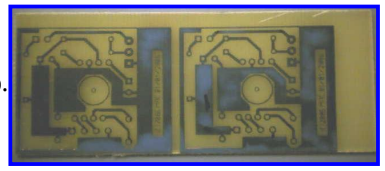
Designator	Description	Quantity	Source
U1	ADNS-2610 Optical Mouse Sensor	1	BTC M850
LED1	HLMP-ED80 LED lamp	1	BTC M850
Q1	2N3906 PNP bipolar transistor	1	BTC M850*
C1	10 µF, 10V electrolytic capacitor	1	BTC M850*
C2	0.1 µF ceramic capacitor	1	BTC M850*
C3	2.2 µF, 10V electrolytic capacitor	1	BTC M850*
R1	32 Ω, 1/8 W resistor	1	BTC M850*
R2	100 KΩ, 1/8 W resistor	1	BTC M850*
R3	1 KΩ, 1/8 W resistor	1	BTC M850*
XTAL1	24 MHz resonator	1	BTC M850
CONN1	Four-conductor ribbon cable	1	
CONN2	9V battery snap with two-pin connector	1	BG Micro BAT1064
n/a	1/16" single-sided PC board material	1½"x1½"	
n/a	Virutal Lenses	1	BG Micro LEN1015
n/a	1¾" plastic hex standoffs	1	BG Micro ACS1178
n/a	2.5 millimeter "Fun Foam" sheet	1"x5"	
n/a	Flexi-straw	1	
n/a	2 millimeter-thick cardboard	½"x½"	
n/a	Self-tapping screws	4	
n/a	Mounting brackets	2	Builder's discression

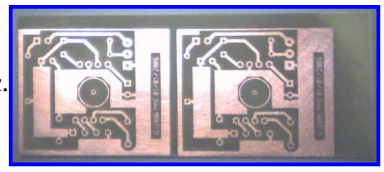
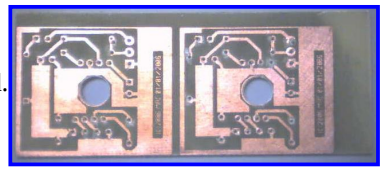
Electrical Assembly

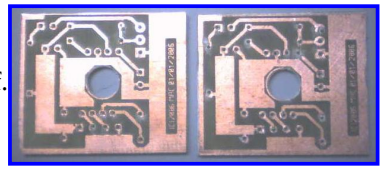
A schematic was developed for CROMS-1 using the open-source gschem tool, which is part of the gEDA Project. Prior to generating the schematic, though, a device symbol for the ADNS-2610 had to be created. The instructions for using tragesym made it very easy to do this. The schematic for CROMS-1 is quite simple. The main component in the schematic is the optical sensor, with resistors, capacitors, resonator, and transistor providing voltage conditioning, timing, and LED driver functions. The power and control signals are provided to CROMS-1 via CONN1. Pin 1 is +5 Volts, pin 2 is the serial clock (SCLK), pin 3 is the serial data input/output (SDIO), and pin 4 is ground. The LED is attached to CONN2 so that it receives power to illuminate the imaged surface. Pin 1 is connected to the anode of the LED. Pin 2 is connected cathode of the LED.

Once the schematic was verified, the artwork for the printed circuit board was created. The gEDA Project came to the rescue again, with a tutorial for using gsch2pcb. It describes how to use the utility to convert gschem schematics to a components footprint data set and a netlist usable by the PCB interactive printed circuit board editor. The gsch2pcb tutorial also describes how to create new device footprints. This was necessary for the ADNS-2610 with its staggered DIP pinout. Once the printed circuit layout was completed, it was replicated so that several CROMS-1 circuit boards could be created during the PCB fabrication process. While the PCB program generated a number of printed circuit layout patterns, I only used the printed circuit back (circuit) layout and the silk screen layout to fabricate the CROMS-1 printed circuit boards.

The printed circuit board was fabricated using the method developed by Thomas P. Gootee; Easy Printed Circuit Board Fabrication Using Laser Printer Toner Transfer. This process is inexpensive and very easy to do as long as you have access to a laser printer or a photocopier and the right glossy printer paper. The Staples brand photo paper described by Gootee is now called "photo basic gloss". Its item number is still 471861, but the UPC barcode for the 30-sheet package is now 7 18103 02856 1.

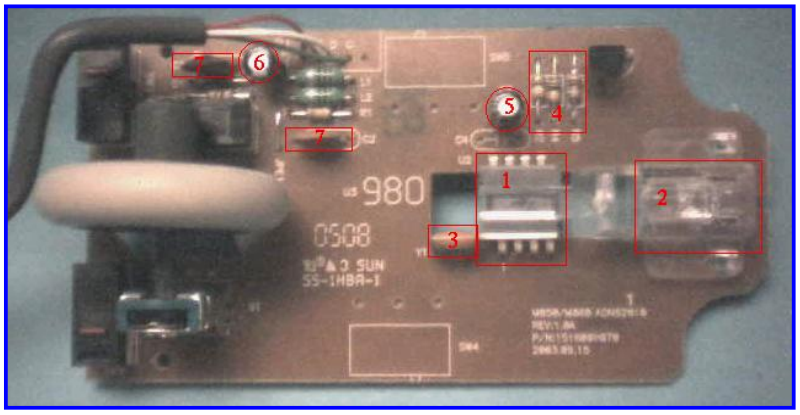
a.  b. 

c.  d. 

e.  f. 

Using Gootee's instructions, I fabricated two CROMS-1 printed circuit boards. The images to the right show the steps in the process after the toner was ironed onto the single-sided PC board material: *a.* mask applied, *b.* etching, *c.* mask removed, *d.* holes drilled, *e.* component placement guide applied, and *f.* finished PCBs. The most difficult task was drilling the holes. It is recommended that this be done with a drill press. A hand-held rotary tool can be used for this task, but the results can be problematic.

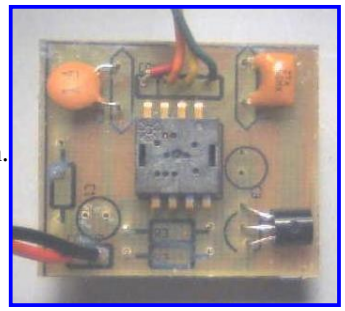
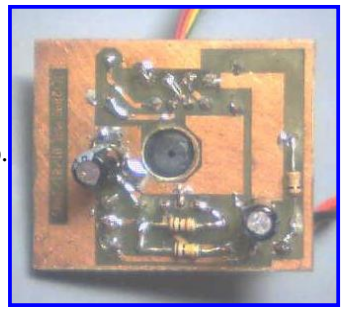
Now that the PC board is fabricated, the parts need to be harvested from the mouse. The locations of the components on the BTC M850 mouse are indicated in the photo:



1. ADNS-2610 mouse sensor
2. HLMP-ED80 LED (unsolder the LED leads, then remove the clip and LED together from the PC board, and then remove the LED from the clip)
3. 24 MHz resonator
4. 32 Ω resistor (orange, red, and black bands)
   1000 Ω resistor (brown, black, and red bands)
   100 KΩ resistor (brown, black, and yellow bands)
5. 2.2 µF, 10V electrolytic capacitor
6. 10 µF, 10V electrolytic capacitor
7. 0.1 µF ceramic capacitor (choose capcitor with the longest leads)

Instructions for disassembling the BTC mouse can be found on the BTC M850 Mouse Hack webpage and will not be repeated here. The parts are easily unsoldered from the mouse's printed circuit board. The board does not have plated-trough holes, so the solder is easily cleared with a soldering iron and a solder sucker tool. It is advised that the aperture (hole) on the ADNS-2610 sensor be covered prior to unsoldering to prevent solder splash or other foreign matter from obstructing or damaging the optical sensor.

Photos *a.* and *b.* show the component and circuit sides of the populated CROMS-1 electrical assembly. Contrary to standard practice, some components were soldered to the circuit side of the board. This was due to the shortness of the leads of those parts harvested from the mouse. Note that if the capacitors are soldered on the component side of the board, the leads must be long enough to allow these capacitors to be laid over on their sides, like with the other components. The four conductor ribbon cable was soldered to the attachment point for CONN1. Left to right; the red wire connects to 5 Volts, the orange wire connects to SCLK, the yellow wire connects to SDIO, and the green wire connects to ground. The 9-volt battery snap was cut from the wire with the two-pin connector. The pair of wires leading to the two-pin connector are shortened to about four inches in length. The red wire is soldered to the pin 1 on CONN2. Its circuit trace goes to R1. The black wire is soldered to pin 2 on CONN2. Its circuit trace goes to the emitter of transistor Q1.

a.  b. 

Optical Assembly

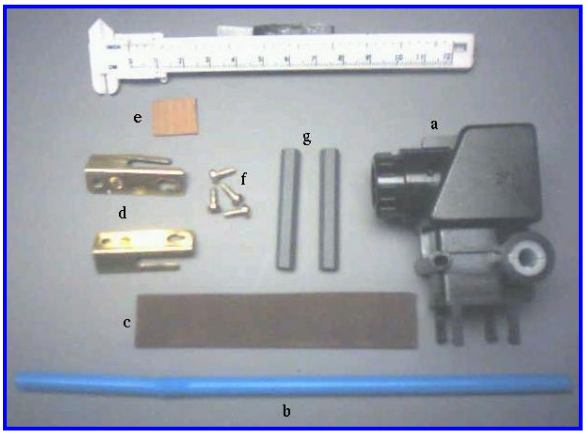
The optical assembly designed for CROMS-1 serves two purposes. First, it provides a housing for the printed circuit assembly described above. Second, it provides a new lens assembly that enables the sensor to be placed some distance above the surface that it is imaging. Unlike an optical mouse, CROMS-1 does not make contact with the surface. For robotics applications, this is undesirable as contact with the surface would cause drag during movement. If the imaging surface is not smooth, then it would even be likely that the sensor would snag, causing the robot's movement to be constrained or potential dammage could be inflicted upon the sensor.

The nice thing about the optics in an optical mouse is that all of the hard work has been done. That is, the lens, illuminating LED, and their positioning have already been designed by Agilent/Avago as part of a reference design. Any optical mouse that follows the reference design will work properly. The CROMS-1 optical assembly, of course, does not follow the reference design. Consequently, a lens must be selected and positioned, along with the illuminating LED, to cause the imaged surface to appear on the focal plane of the ADNS-2610 optical sensor.

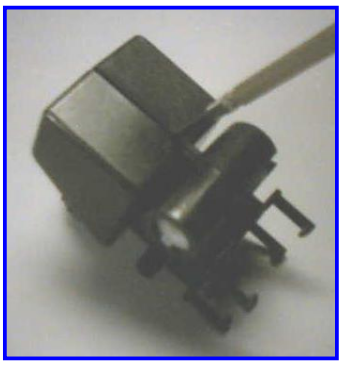
Fortunately, the NASA Tech Briefs article, mentioned above, goes into some detail about deriving the optical characteristics of the optical mouse reference design. All of the necessary information was found in the Agilent/Avago data sheets. Using some fundamental knowledge of thin lens optics and depth of field equations, it is relatively straightforward to take this information and develop the CROMS-1 lens system, given a lens with the desired optical behavior and a known focal length.

I found that several design issues had to be met to obtain a successful design. Some of those issues were obvious to me, but other were not. First, a usable lens with a reasonably short focal length had to be found. Second, an enclosure had to be found that would hold the lens and the electrical assembly in a compact arrangement. Third, which wasn't immediately obvious to me, the brightness of the image had to be controlled so as not to overload the auto-brightness controls of the mouse sensor. I had the great fortune of finding that the BG Micro Virtual Lenses assembly resolved most of these issues. It has a lens with a focal length of 30 millimeters and its enclosure is just large enough to hold the electronics of the optical sensor. The remaining issues had to be addressed in the details of the optical design.

The following pictorial described the steps taken to fabricate the CROMS-1 optical assembly. None of the steps are particularly difficult to perform. Having a rotary tool is the most useful device for making this easy to do.

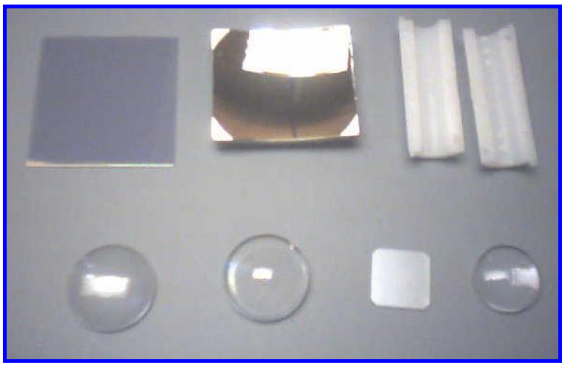


The optical assembly was created from the items illustrated to the right. These items are *a.* Virtual Lenses assembly, *b.* flexi-straw, *c.* small sheet of fun foam, *d.* mounting bracket for mini-blinds, *e.* small square of 2 millimeter cardboard, *f.* self-tapping screws, and *g.* plastic hex standoffs. The plastic caliper at the top of the photo is there to show the size of the items.

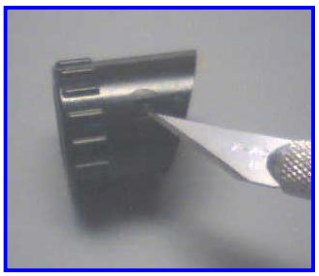


The Virutal Lenses assembly is easily opened by running the blade of a screwdriver or other thin tool between the halves of the case. Start from the middle of the back and working around; separating the halves a little at a time. Eventually, the two halves will separate. Be careful not to let the lens fall out as the halves separate.

Upon opening the case, a treasure trove of lenses and mirrors are found. The Virtual Lenses assembly appears to be some form of right-angle eyepiece for some form of optical system. Could it have come from some medical equipment? Or could it be from some obsolete military hardware? One can only wonder. The good folks at BG Micro only say that it was part of a virtual reality screen.

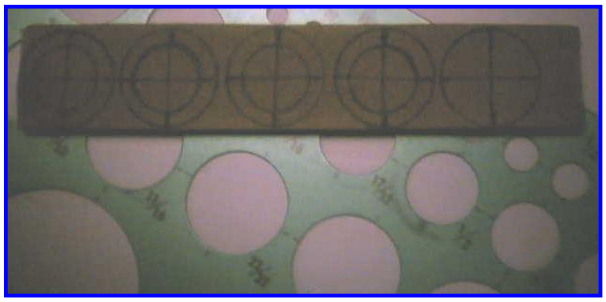


The optical items within the Virutal Lenses assembly consist of two plano-convex lenses, a concavo-convex lens, a partially-silvered beam-splitting mirror, a concave mirror, and a small, square graticule. The white, plastic insert splits into two pieces once it is removed from the case. It may be part of a focusing mechanism, though that is just a guess. Be careful handling the lenses, as they scratch easily. The lens that will be used is the smallest lens on the lower, righthand side of the photo.



When the halves of the Virtual Lenses assembly are separated, the eyepiece is freed. This eyepiece provided some form of fine focusing ability, as it could rotate withing the Virtual Lenses assembly. This eyepiece will serve as the lens barrel. It already contains some lenses, which need to be removed. Using a hobby knife, cut away the lens retaining tabs on both sides of the lens barrel. The lenses will slip out easily revealing a thick double-convex lens and a concavo-convex lens.

In order to place the lens at the proper position in the lens barrel, a spacer must be constructed. I used some "fun foam" that had been discarded from another project. The idea is to insert a spacer that is about 10 millimeters in length between the bottom of the inside of the barrel to where the lens will be placed. It turns out that four rings of fun foam provide about the right spacing. The outside diameter of the rings is 23/32" and the inside diameter is 1/2". The fifth ring will be used as an aperture stop.



The photo to the right show the spacers after they have been cut out. Don't worry if the holes are not perfectly centered. The positioning is not all that critical, as will be shown. The ring on the far right is the aperture stop. It has a hole 1/4" (6.35 millimeters) in diameter in the center. I used a single-hole punch to create the hole. As you can see, this hole is not perfectly centered either. This will also not be a problem.

The four photos to the right show the steps in the lens barrel assembly process: *a.* The components gathered together prior to assembly. The spacers are are stacked to show their combined height (about 10 millimeters). *b.* Using epoxy, glue the spacers into the lense barrel. Make sure that the spacers are sitting level one on top of the other. Before the epoxy sets, move the spacers around inside the barrel center the inner tube formed by the spacers. Now place a thin bead of epoxy on the topmost spacer and carefully set the lens on top of the spacer's opening. Be careful not to get epoxy on the inner area of the lens. Otherwise, the epoxy will distort the focus of the lens. While the epoxy is setting, take a short strip of double-sided adhesive tape and punch a hole 1/4" (6.35 millimeters) in diameter into the tape. Then place the tape onto the aperture ring, making sure that the holes in the tape and the aperture ring are aligned. *c.* After the epoxy has set, place the aperture ring onto the lens. The tape will secure the aperture ring to the lens. Make sure that the aperture hole is centered on the lens. *d.* As can be seen, the aperture ring restricts the aperture angle of the light reaching the optical sensor. The misshapened spacer rings don't have that much effect on the light reaching the sensor.

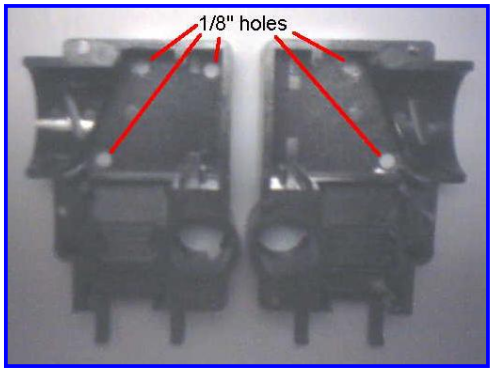
a. 

b. 

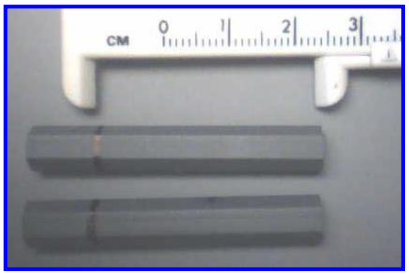
c. 

d. 

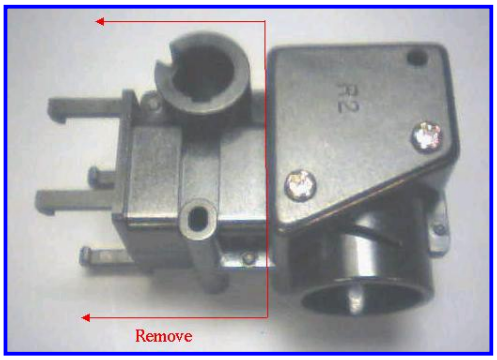
While the epoxy on the lens barrel is drying, work can start on the body of the optical assembly. The first step is to drill some 1/8" holes into the sides of the body. The photo shows where they should be placed. The holes closest to where the lens barrel sets are for screws that will hold the body halves together. The hole at the back of the right half of the enclosure will allow wires for the illuminating LED to exit the enclosure.



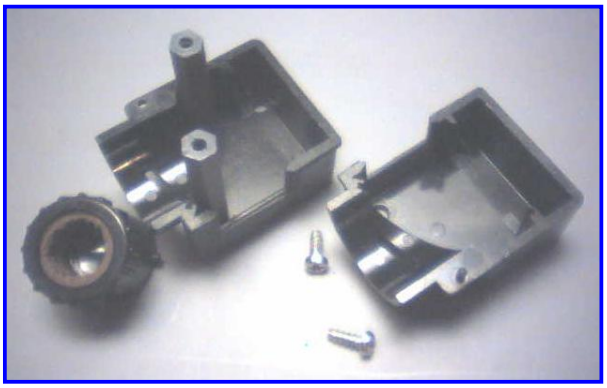
The next step is to cut the hex standoffs to the proper length. The standoffs need to be shortened to 34 millimeters. A rotary tool with a cutoff wheel will make short work of this task. Prior to mounting the standoffs into the optical enclosure, drive the self-tapping screws into the ends of the standoffs. This will make it easier to drive the screws into the standoffs during the actual assembly.



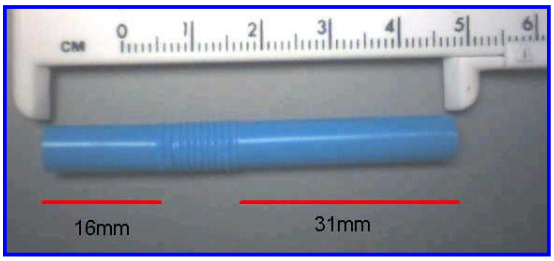
Assemble the standoffs and the halves of the enclosure together to test the fit. At this point it is necessary to remove the unneeded plastic of the enclosure. The photo on the left show what needs to be removed. Disassemble the enclosure and use a rotary tool to cut off the plastic. If desired, the opening that is left can be blocked with two small sheets of plastic, one for each side of the enclosure. The photo on the right shows the enclosure after reassembly.



This photo shows what the inside of the enclosure should look like with the standoffs installed. With the epoxy on the lens barrel dry and the aperture ring installed. The optical assembly can be reassembled for the next step: adding the LED wire conduit.

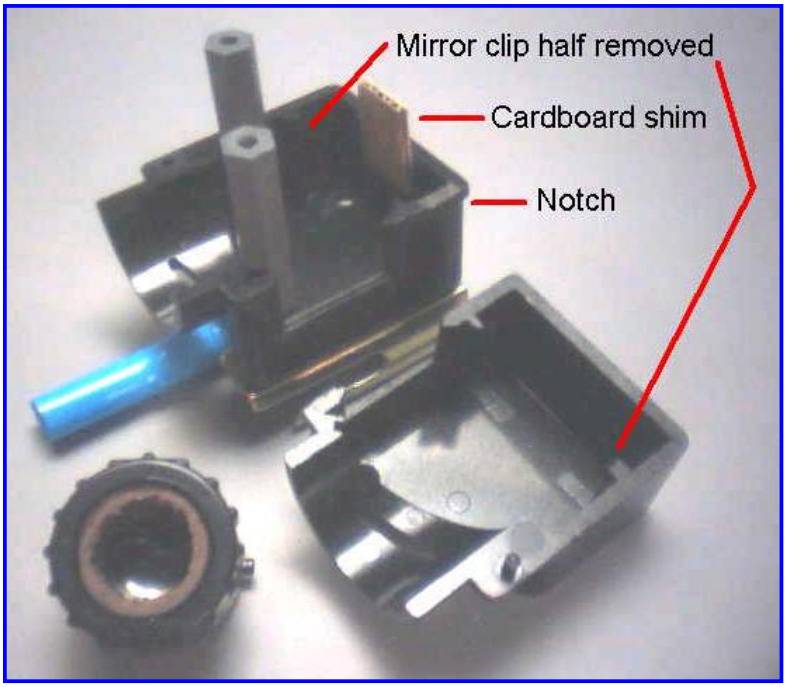


The LED conduit consists of a flexi-straw cut to the right length and epoxied to the right half of the optical enclosure. One end of the straw needs to be cut down to a length of 16 millimeters from the one end of the straw's "accordion". The other end of the straw needs to be cut down to a length of 31 millimeters from the other end of the "accordion". The total length of the straw should be 58 millimeters. These measurements should be adjusted if the length of the "accordion" is different from what is shown here.



The LED conduit is epoxied to the right side of the optical enclosure. It should be positioned so that it runs parallel to the centerline of the lens barrel. The top end of the conduit should be in line with the 1/8" hole that allows the LED wires to come out of the enclosure. Most likely, the straw is made out of polyethylene. Consequently, epoxy won't stick very well to the tube. To resolve this problem, use enough epoxy to totally cover the bonded area of the straw. The epoxy will harden, forming an enclosing "bracket" around the tube. Note that one of the mounting brackets is attached to the optical enclosure. This was done because the epoxy tends to run a bit. The excess epoxy helps to secures the mounting bracket to the optical enclosure.

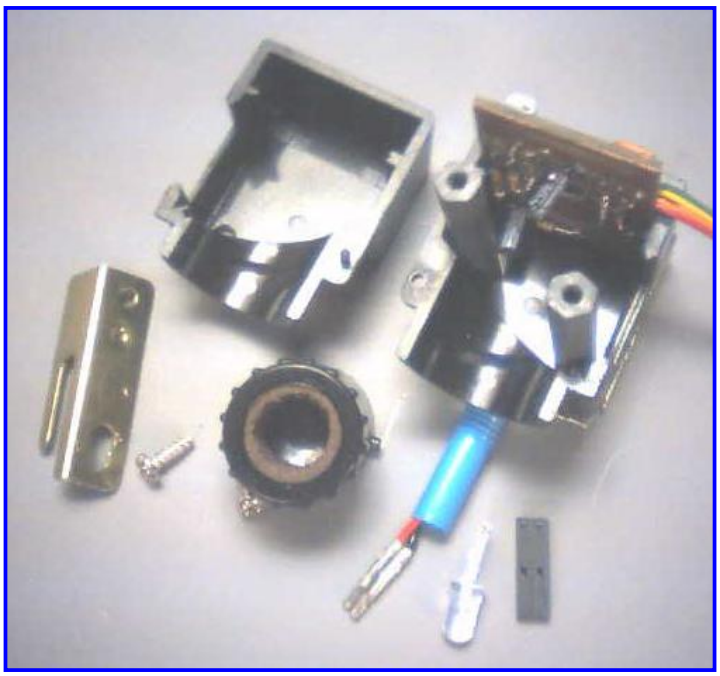
After the epoxy has set, disassembled the enclosure again. It is necessary to trim out part of a clip that held the beam-splitting mirror. **Do not totally remove the clip!** Only remove the portion the curls around to hold the mirror. Leave the portion that extends from the wall of the enclosure. This serves as part of the support structure for the electrical assembly. See the photo for details and compare it to some of the prevous photos of the interior of the enclosure. In addition, a small notch should be cut into the right half of the enclosure near the large opening in the enclosure. This notch is needed to provide some clearance for the four-conductor ribbon cable that comes out of the electrical assembly. The notch should not be cut too deeply, as the compression of the two halves of the enclosure provide a strain relief to the electrical connections of the ribbon cable.



Finally, attach a small square of 2-millimeter thick cardboard to the back of the right half of the enclosure. Double-sided adhesive tape works well for this task. The cardbord duplicates the function of the clip used inside the reference mouse design. It keeps the electrical assembly pressed flat against the support structure of the enclosure. The CROMS-1 is now ready for its final assembly!

Final Assembly

The final assembly consists of inserting the electrical assembly into half of the optical assembly. Prior to this insertion, the shell of the two-wire connector must be removed temporarily. This is to allow the wires to pass through the hole in the enclosure and through the LED conduit. The shell of the connector is removed by using the fine point of a hobby knife to push in two small tabs on the crimp connectors inside the shell. These tabs are visible through two small holes on the side of the shell. Once the crimp connectors are removed, bend the tabs back out so they'll lock back inside the shell when they are reinserted.

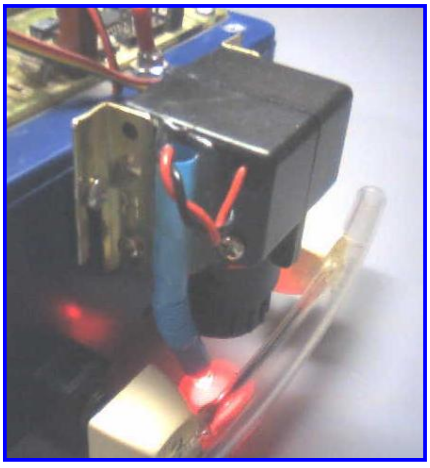


The short side of the printed circuit board with the LED wires should be inserted into the right half of the optical enclosure. The LED wires need to exit though the hole in the enclosure. The printed circuit board should then fit between two tabs at the back of the enclosure and the support structure. The top of the ADNS-2610 sensor should fit under the carboard square. The electrical assembly should fit snuggly and not shift around, though it should be easy to remove. It may be necessary to slightly trim the printed circuit board using a file to ensure a proper fit. *It would be a good idea to do the trimming of the printed circuit board prior to populating it with components.*

The lens barrel needs to be put into its proper location,

followed by the other half of the optical assembly. The assembly is then secured with the screws. The halves of the enclosure should fit closely together with little or no gap between them. The lens barrel should be able to rotate easily with a twist. It should not so loose that the tube rotates freely. The other mounting bracket should be attached at this time if one is used.

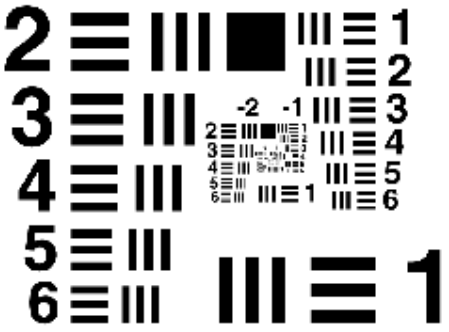
The LED wires should now be slipped down through the LED conduit to extend out the other end near the lens barrel. Place the crimp connectors back into the connector shell, making sure that they click in place and cannot be pulled back out. Finally, plug the LED into the connector. The flat side of the LED (cathode) should plug into the connector side with the black wire. The connector is then pushed back up into the LED conduit so that only the LED extends out of the conduit. The wire should be pulled up at the other end to form loop. The loop should prevent the connector from dropping back out of the conduit. The "accordion" of the LED conduit can now be adjusted to shine the LED onto the spot observed by CROMS-1.



The photo to the right shows the finished sensor mounted to my robot. I built two of these sensors. One is mounted to the front of the robot and the other is on the rear. The vertical positioning of each sensor is such that the surface underneath the robot is in focus (more or less).

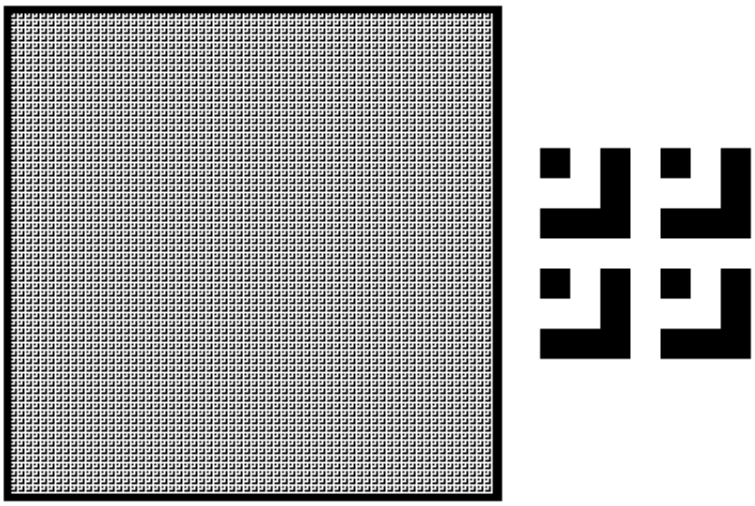
Testing

Once the CROMS-1 was attached to my robot, I needed to do some testing to see how well it performed. I decided to use two different test patterns. The first test pattern is the classic USAF Lens Test Chart. I used the Group 3 patterns, as these are what were used in the NASA Tech Brief article mentioned above. The illustration at the right is not to scale, but is only representative of the appearance of the USAF test chart. You can download an actual USAF test chart, in several graphic formats, from here.



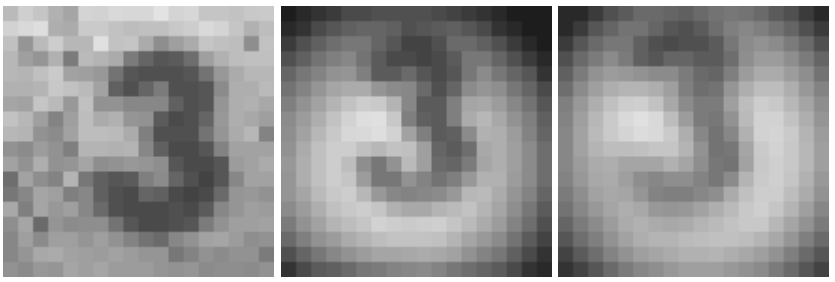
The nice thing about the USAF lens test pattern is that it is a long accepted test standard. One problem with it is that it is difficult to find the Group 3 pattern with my sensor. The sensor's field of view is only a bit more than one millimeter square. It takes a bit of careful nudging to place a small portion the Group 3 pattern within the field of view. Another problem is that my copy lens test chart was created

with a laser printer, so it does not have the same high-quality of a photographic film or a metal stencil.

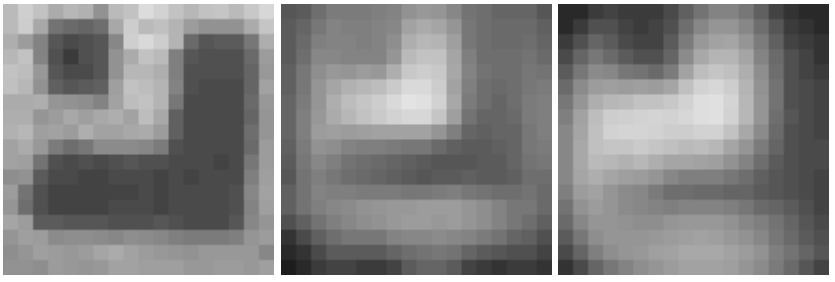


The second pattern I used is one that I designed myself. Its purpose is to provide an indication of the sensor orientation relative to the pattern. The left graphic is the actual pattern. The right graphic is the pattern magnified sixty-four times. The pattern forms a repeating set of points and line segments at right angles, providing an unambiguous indication of orientation. Consequently, it is fairly easy to align the field of view to see one of the patterns.

I used the same robot to test the CROMS-1 sensor as was used to test the BTC Mouse Hack. It made sense test the new sensor with a derivative of the Forth code that I used then. The code I used to test the functions of CROMS-1 is listed here. The peculiarities of the Forth code syntax are described on the BTC Mouse Hack web page and won't be repeated here.



The left photo shows the image seen by the optical sensor of the BTC Mouse Hack test device when positioned over the number "3" on the Group 3 USAF Lens Test Chart. The number is clearly seen. The center and right photos are images of the same number "3" as observed by the two CROMS-1 sensors that I contructed for my robot. As can be seen, the focus is not quite there yet. It is fairly close, considering that the depth of field is only about two millimeters. There is also some vignetting, probably due to a combination of the aperture ring and uneven illumination by the LED. In the reference mouse design, the light coming from the LED is focused somewhat by its own lens and prism.



Using my designed test pattern gave similar results. The left photo shows the image seen by the optical sensor of the BTC Mouse Hack test device. The center and right photos are images of my test patten taken by the CROMS-1 sensors. Work will be done to continue with improving the focus of the sensor. As it stands now, CROMS-1 can still be used to detect motion, even with the focus and vignetting problems. It will be useful for robot navigation applications.

Last updated Friday, March 17, 2006

BTC Optical Mouse Hack

A while back, I happened upon the application note "Interface to Optical Mouse Sensor" on the Kronos Robotics website. It discusses the application of a sensor from an optical mouse for use as a position sensing coprocessor on a robot. The application note did not go beyond the proof-of-concept stage. It did not describe how to use all of the functions of the Agilent mouse sensor, which are available through a direct interface to the mouse sensor. It turns out that there are many functions available in the Agilent mouse sensor that cannot be accessed through the PS2 interface.

I didn't have easy access to the GE mouse described in the Kronos Robotics application note. I had to find another optical mouse that employed an Agilent optical mouse sensor. I also wanted to go beyond the proof-of-concept stage and acutally use the mouse sensor in a robotics application. Before developing that application, though, I first had to gain familarity with interfacing directly to the Agilent optical mouse sensor and accessing its functions from within my Forth environment. This webpage describes the hack I did to a BTC optical mouse to enable direct control of the Agilent (now Avago, see below) ADNS-2610 Optical Mouse Sensor and the Forth code written to control it

Inside the BTC M850 3D Optical Mouse

BTC (Behavior Tech Computer) is a Taiwanese manufacturer of low-cost computer peripherals and consumer electronics devices. They have sold a wide range of mice, including standard ball mice, optical mice, and (apparently) laser optical mice. The M850 is a three-button and scroll wheel optical mouse. The packaging, illustrated to the right, calls the mouse as a "3D Optical Mouse", though it really is justs a typical optical mouse. Perhaps, the "3D" capabilities are derivied from the KeyMaestro software on a floppy disk that (sometimes) comes with the mouse. I found this mouse at Fry's Electronics and they usually have plenty in stock (2005 - 2006 time frame). While the Fry's price sticker shows a $6.99 price tag, it is

often on sale for $4.99 (one per customer please!). Note the "Agilent enabled" label and the M850 model number at the top of the box front, as these are important to ensuring that you get the correct BTC mouse. Note that Agilent Technologies has spun off its optical mice components to Avago Technologies, so if BTC continues to use the same optical mouse sensors in the future, the packaging may change.

The M850 mouse has a rounded, ergonomic body. The right and left mouse buttons are formed by a single, wrap-around plastic element that is flush to the body. The mouse wheel extends out of the center-front of the mouse body. The wheel is smooth, but easily rotated and pressed. I've seen two mouse color styles: dark grey body and cord with a white mouse wheel and black body and cord with a black mouse wheel. The plastic element for the mouse buttons is always silver in color. The M850 box literature mentions the availability of mice with either a PS2 connector, a USB connector, or a PS2 connector with a USB adaptor. I've only seen the version with the PS2 connector sold at Fry's Electronics.

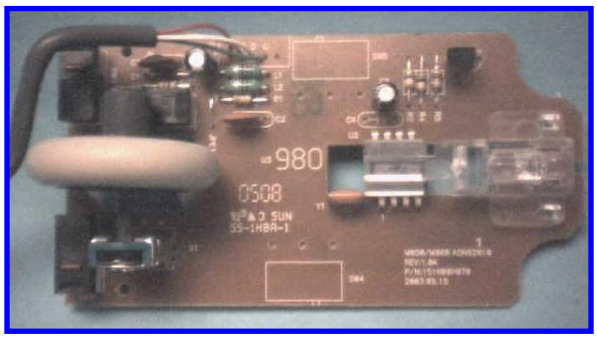
The bottom of the mouse is unremarkable. Just a label describing the product and its compliance with various governing organization. Access to the guts of the mouse is accomplished by first removing the two metal screws located on the the right and left of the bottom rear of the mouse. The mouse button element must also be released by unclipping it by pressing the hook on the far back end of the bottom of the mouse. The base of the mouse is then removed by sliding it back and away from the body shell of the mouse. Be careful that the mouse button element does to re-clip

itself back to the base of the mouse.

Once the mouse body shell is removed the guts of the mouse are clearly visible and accessible. There is not much to look at. The printed circuit board holding the mouse electronics is easily remove by unwrapping the mouse cord from the built-in strain relief in the base of the mouse and lifting the board out.

With the printed circuit board removed, you can see the Agilent (Avago) HDNS-2100 Solid-State Optical Mouse Lens. The lens actually consists of the focusing lens for the mouse sensor and a light prism that directs the light from the illuminating LED onto the viewing surface.

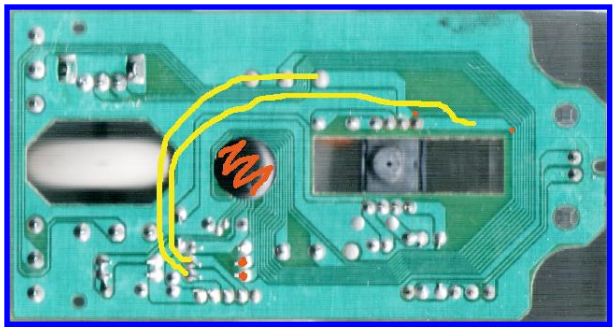
The printed circuit board is a simple, single-sided affair. The microswitches of the mouse buttons and the rotary encoder of the scroll feature are to the front of the board. The electronics for the optical mouse sensor are to the rear. The sensor itself figures prominently in the center. The clear plastic clip is the Agilent (Avago) HDNS-2200 Solid Statte Optical Mouse LED Assembly Clip. It holds the LED at the correct position to shine into the light prism of the HDNS-2200. The part of the clip over the mouse sensor acts as a plastic spring that keeps the clip and lens assembly aligned. Note that this printed circuit board supports two BTC products: the M850 and the M860 (4D Optical Mouse). There are silk screen markings and solder pads for two more microswitches.



The circuit side of the printed circuit board is the typical maze of narrow traces. One unique feature is the that the mouse controller processor chip is attached using chip on board (COB) technology. The chip is hidden under the round black blob of epoxy at the center of the board. The other (obvious) unique feature is the bottom of the optical mouse sensor. The small hole at the center of the round extension of the sensor is the optical aperture. The light from the illuminating LED reflects off the target surface and is focused by the lens onto the sensor array through the optical aperture.

Modding the Mouse

With the mouse completely disassembled, it is now time to modify the wiring to allow direct control of the optical mouse sensor. This is actually very simple to do. The idea is to cut out the "middle man" (the mouse controller) and wire the mouse cable's data and clock lines directly to the optical mouse sensor. Refer to the circuit illustration to the right.

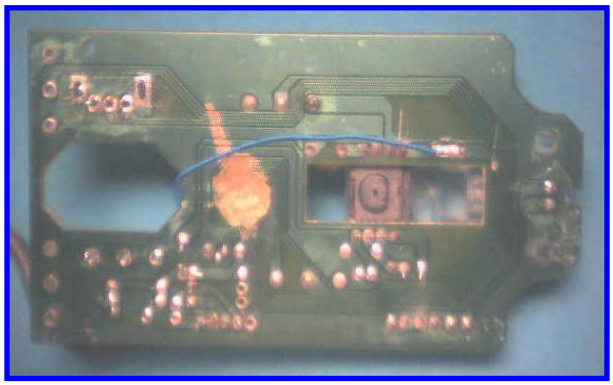


The red dots near the sensor indicate where traces are cut to decouple the data and clock signals from the mouse controller. The red dots below the mouse controller indicate the leads of two inductors (labeled L1 and L2 on the component side of the board) that need to be unsoldered from the circuit. Note that the other pins of the two inductors are also disconected from the traces that lead to the cable's data line (white wire) and clock line (green wire). These lead holes need to be cleared for the insertion of new wires.

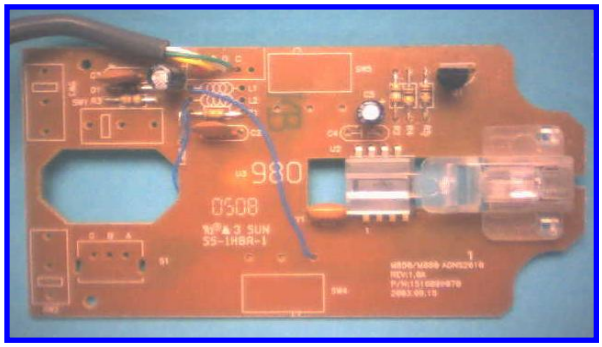
The yellow lines indicate the new wiring between the mouse sensor's data and clock lines and the mouse cable's data and clock lines. The wire attachment

point for the sensor's data signal is an unused lead hole that must be cleared first. The wire attachment point for the sensor's clock signal is made by scraping away some of the solder mask off the indicated trace and lap soldering the wire to it.

The mouse controller is also removed by cutting away the epoxy blob and removing the chip. This is most easily done by grinding the epoxy off using a rotary tool. Before performing the modifications, it is strongly advised that you cover the optical aperture of the mouse sensor to prevent foreign objects (dust, solder, etc) from getting onto the sensor array or blocking the aperture. I covered the aperature with a paper dot, created by a hole punch, followed by tape to hold the paper dot in place.



The two pictures to the right show the finished modifications. I was almost too agressive with the rotary tool while removing the mouse controller chip. I succeeded in not only removing the epoxy and the chip, I almost removed the underlying printed circuit board as well. Fortunately, I didn't need to use the surrounding circuit traces. The unused microswitches and rotary encoder were harvested from the mouse for use in other robotics projects. The wiring was done using 30-gauge wire used for wire-wrapping.



In retrospect, I probably didn't need to remove inductors L1 and L2, though doing so made it easier to attach the wires. The inductors are probably used to condition the signals traveling on the mouse cable. The mouse cable was originally about five feet in length. I shortened it to a bit less than three feet in length. Losing the inductors didn't appear to have much of an impact for my testing. YMMV.

I fashioned a new connector at the end of the

shortened mouse cable to plug into my robot. I won't show it here, but what you'll need to do is provide 5 volts power (red wire), ground (yellow wire), and the mouse data and clock signals. If you want to use the original PS2 connector for this purpose, that is fine. You may want to keep inductors L1 and L2 in place if you retain the original cable length. Remember that the data and clock signals on the PS2 connector are now connected directly to the mouse sensor. After this mod, the signals on the PS2 connector are no longer compliant to the PS2 mouse standard.

The mouse, once reassembled, is now ready for experimentation!

Optical Mouse Sensor Control

My testing of the mouse sensor was performed using a Harris RTXEB single board computer, which is based on the Harris RTX2001A Real-Time Express microcontroller. Consequently, the test code presented below is written in Forth. I believe that the code is sufficiently commented that it could be easily rewritten in another programming language.

The mouse sensor is accessed through a parallel port placed at address 0x1A on the G-bus of the RTX2001A microcontroller. Only a few bits of the port are actually used because of the synchronous serial data interface of the ADNS-2610 optical mouse sensor. Listed below are the definitions of the bits of the G-bus data (GD) used by the synchronous serial data interface:

GD0 (write) - Serial Clock (\SCK)
: The inverted serial clock used to shift data into and out of the optical mouse sensor.

GD0 (read) - Serial Data Input (SDIO)
: Input data read from the optical mouse sensor.

GD1 - Data Direction Control
: Flow control for data shifted to or from the optical mouse sensor (0 input, 1 output).

GD7 - Serial Data Output (SDIO)
: Output data written to the optical mouse sensor.

The Forth code is written using the built-in RTXEB Forth interpreter, EBFORTH. While this interpreter is mostly compliant with the Forth 83 standard, some of the instructions are specific to the RTXEB. These are described as follows:

`G! ( n g -- )`
: Writes a value n on the stack into a specified G-bus port address g and removes the value from the stack.

`G@ ( g -- n )`
: Read a value from a specified G-Bus port address g and pushes it onto the stack.

`H ( -- addr)`
: The address of a variable containing the next available dictionary location for code.

`H-FENCE ( -- )`
: The address of a variable containing the beginning address of the user dictionary (the default is 0x4400).

`HEX ( -- )`
: Sets the numeric input-output conversoin base to sixteen

`R ( -- )`
: The address of a variable containing the next available memory location for variables.

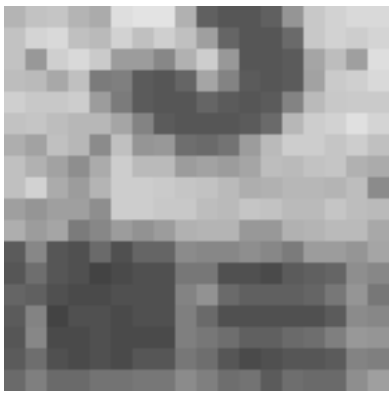
`R-TOP ( -- )`
: The address of a variable containing the maximum memory address for variables (the default is 0x43FE).

The Forth code listing defines the Forth words used to implement the synchronous serial data interface that sends and receives data from the ADNS-2610. The serial clock on bit GD0 is inverted because the mouse sensorconsider an idle clock to be held at the high state. For my implementation of the interface, it was easier to invert the signal to assume a low level for the clock idle state. Data that is received from the mouse sensor is delivered most-significant bit first. As sequential bits of data are read in on bit GD0, they are left-shifted to move them into bits of greater significance. Data that is sent to the mouse sensor are also delivered most-significant bit first. Therefore, the data output is aligned to bit GD7. As sequential bit are written to the mouse sensor, the next least-significant bit is left-shifted so it is ready to be written to bit GD7 as well. The data direction control on bit GD1 controls the direction of data written to or read from the mouse sensor. The synchronous serial data bus is bidirectional.

A number of Forth words were written that encapsulate the operations needed to exercise

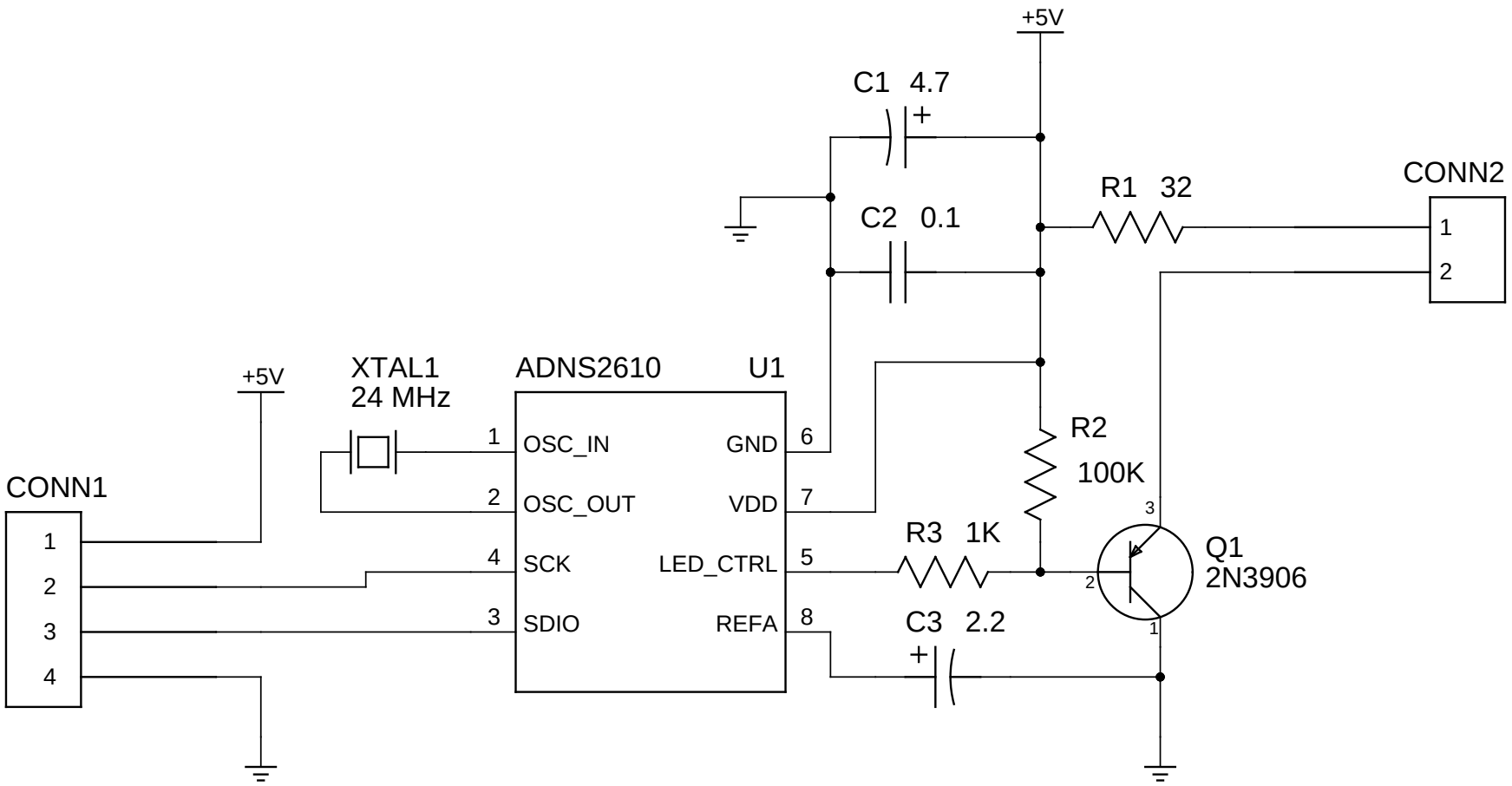
the various registers of the ADNS-2610. The words mirror the register functions described in the ADNS-2610 documentation. With the words the mouse sensor's operational state can be controlled and monitored, the motion counts can be read, the image surface can be evaluated, and the individual pixels of the image can be retrieved.

The animated GIF image to the right shows a sequence of three images taken with the mouse sensor sitting on top of the "Group 3" lines of a USAF Lens Test Chart. Please forgive the quality of the test chart, as it was created with a laser printer. It is not a high-quality photographic film or a metal stencil.

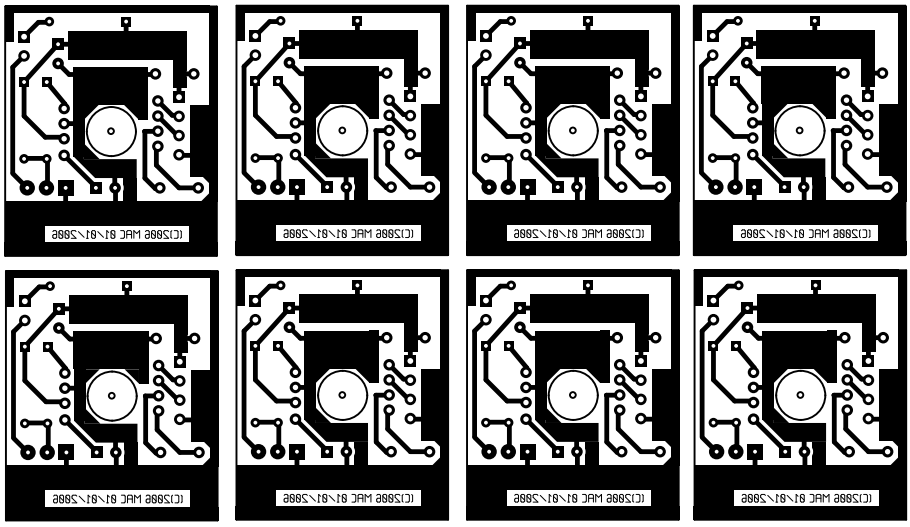


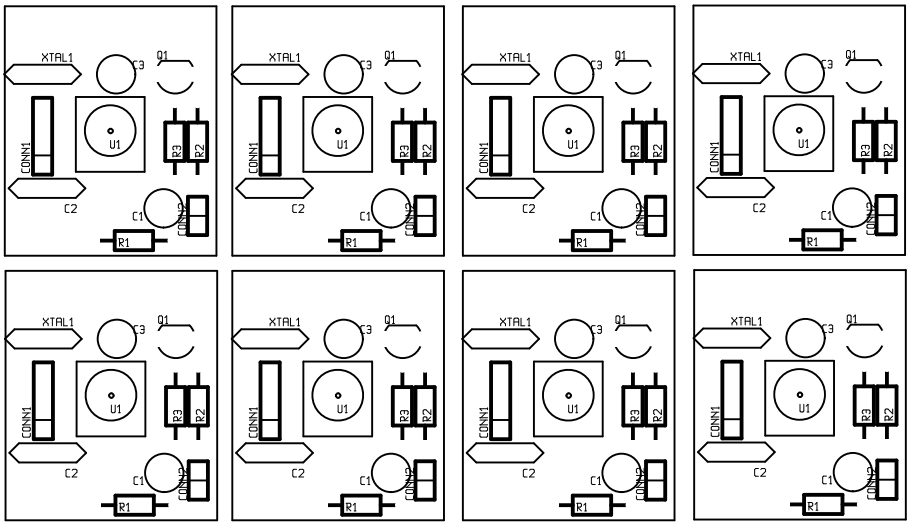
Now that the optical mouse chip can be accessed and controlled directly, it can be used to experiment with to develop new applications for robotics.

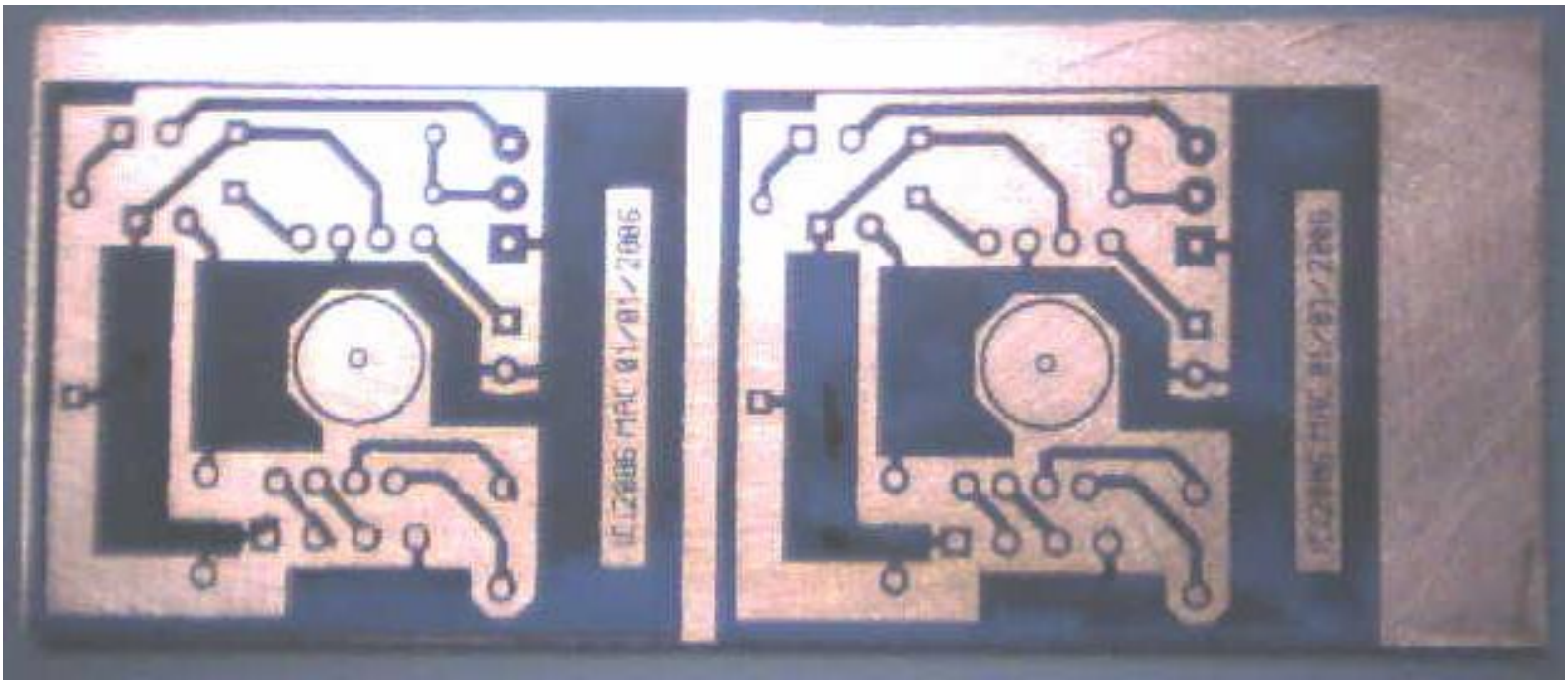
Last updated Friday, March 17, 2006

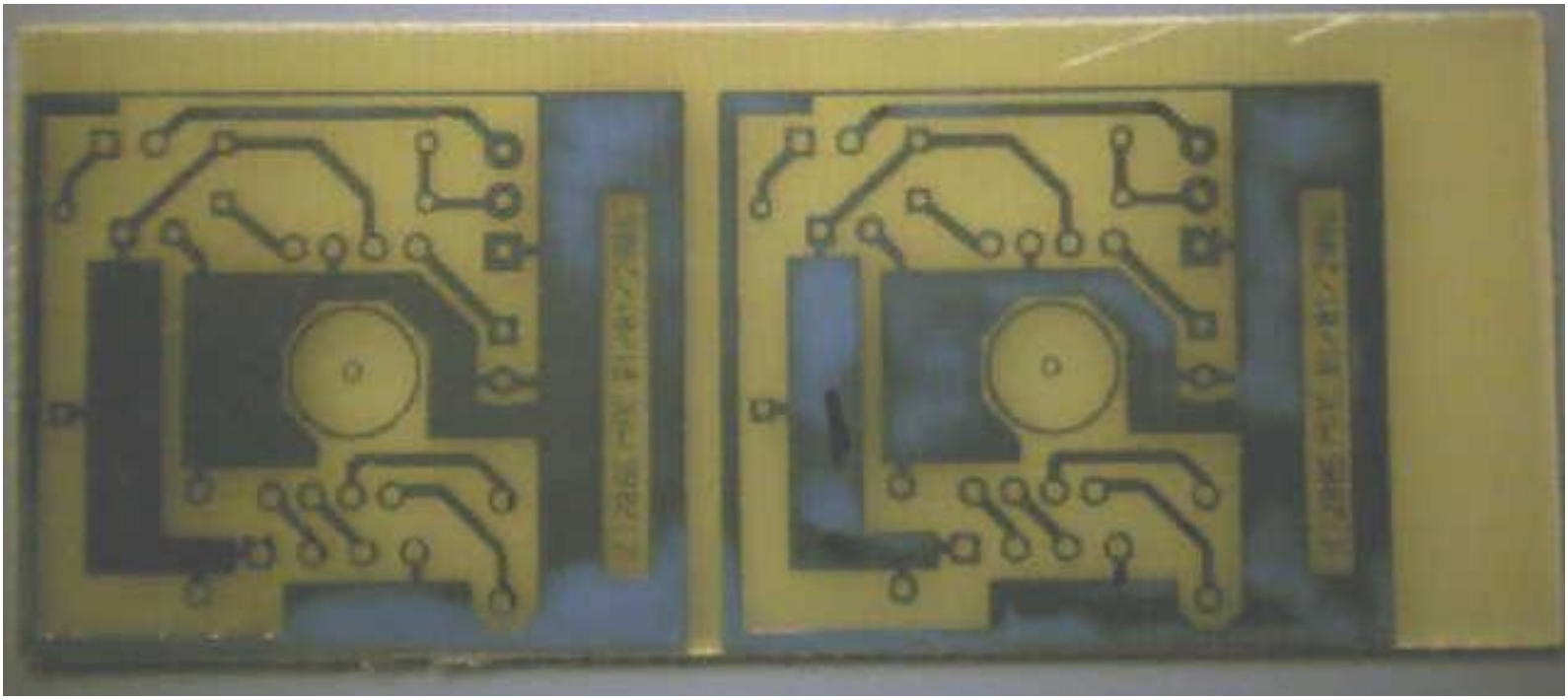


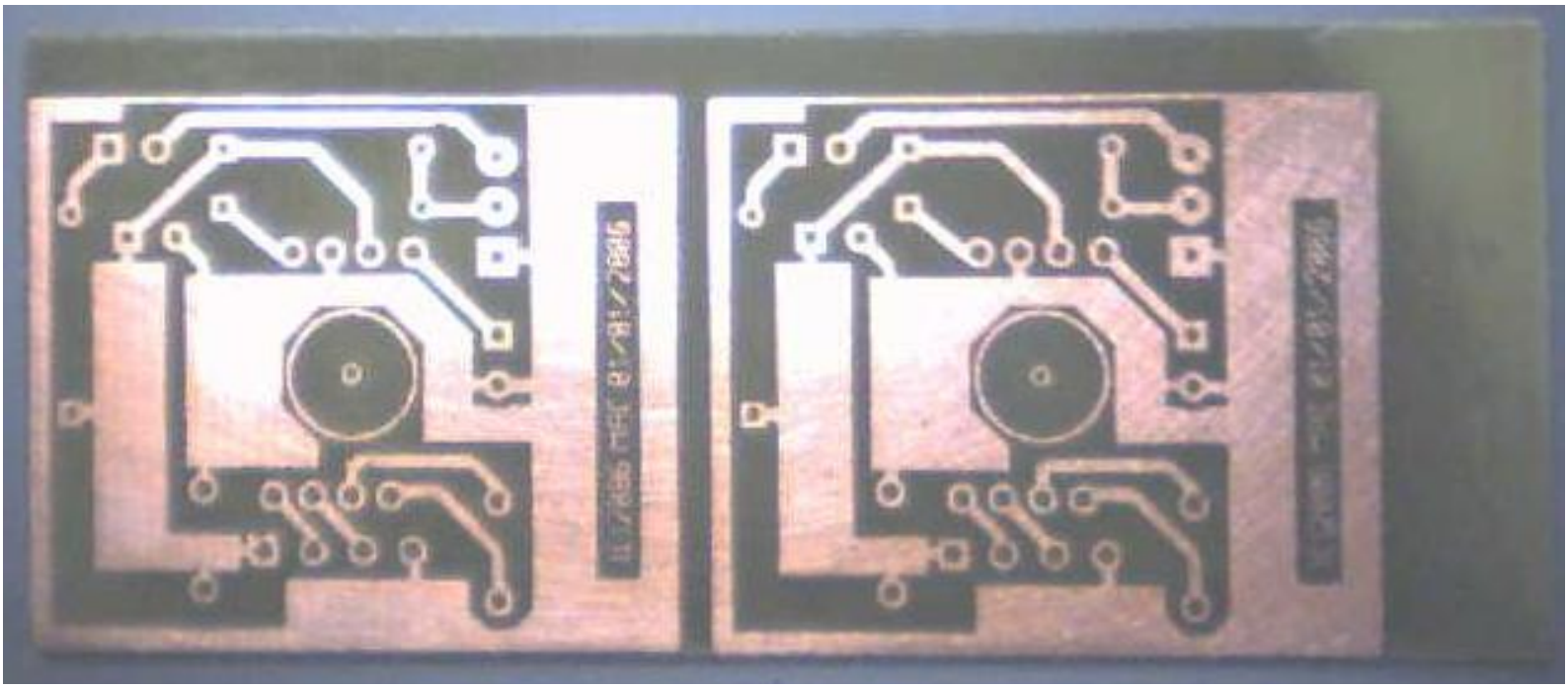
Optical Encoder #1 TITLE	
FILE: oencoder1.sch PAGE 1 OF 1	REVISION: DRAWN BY: Mac A. Cody

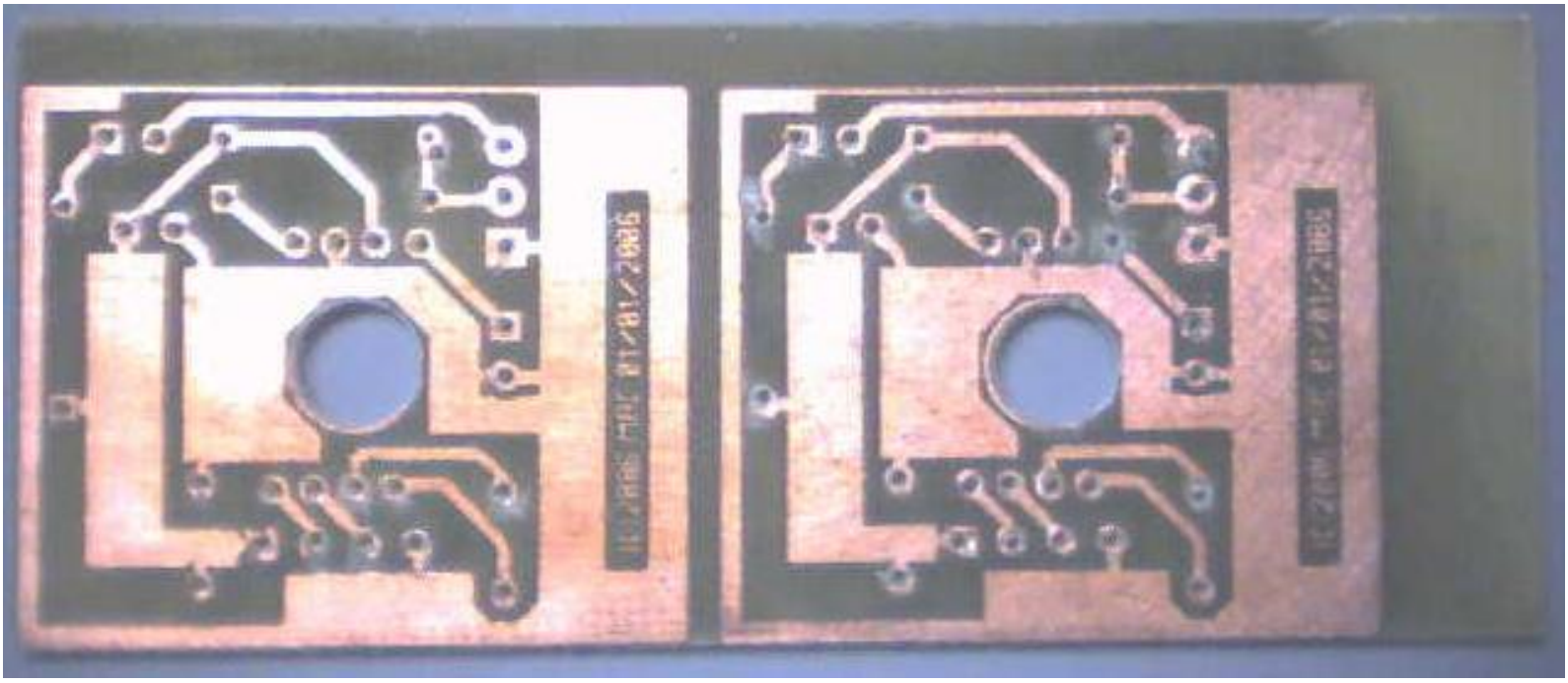

(C)2006 MAC 01/01/2006

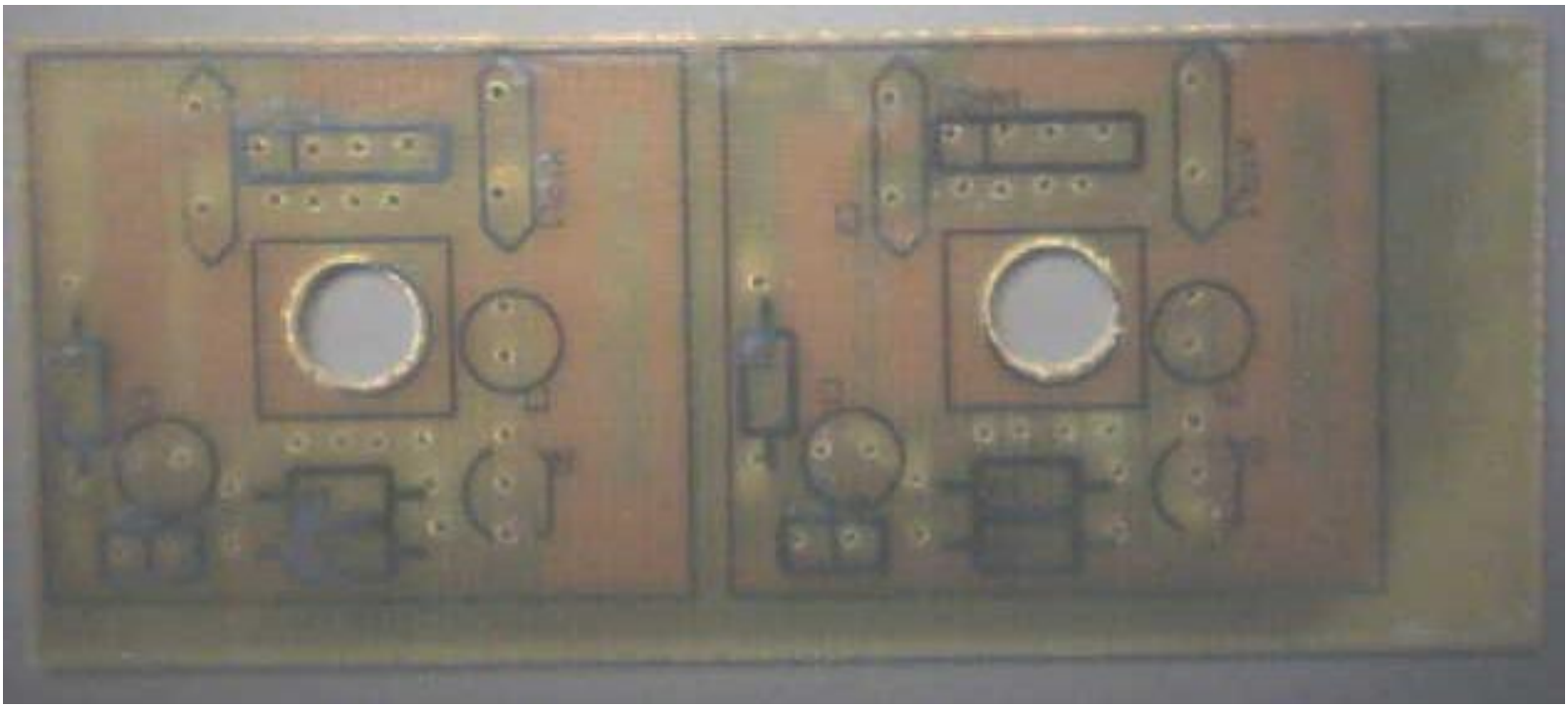

XTAL1
C3
Q1
CONN1
U1
R3
R2
C2
C1
CONN2
R1

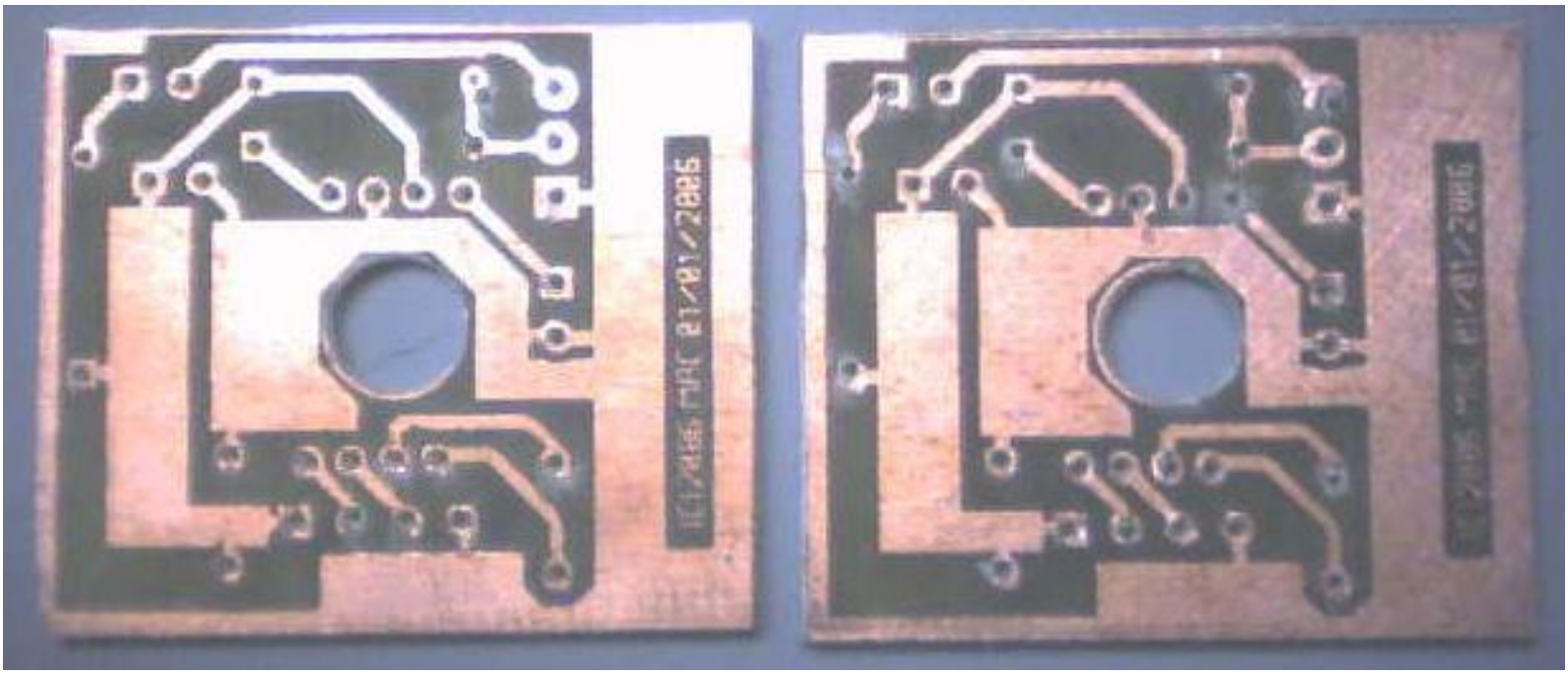


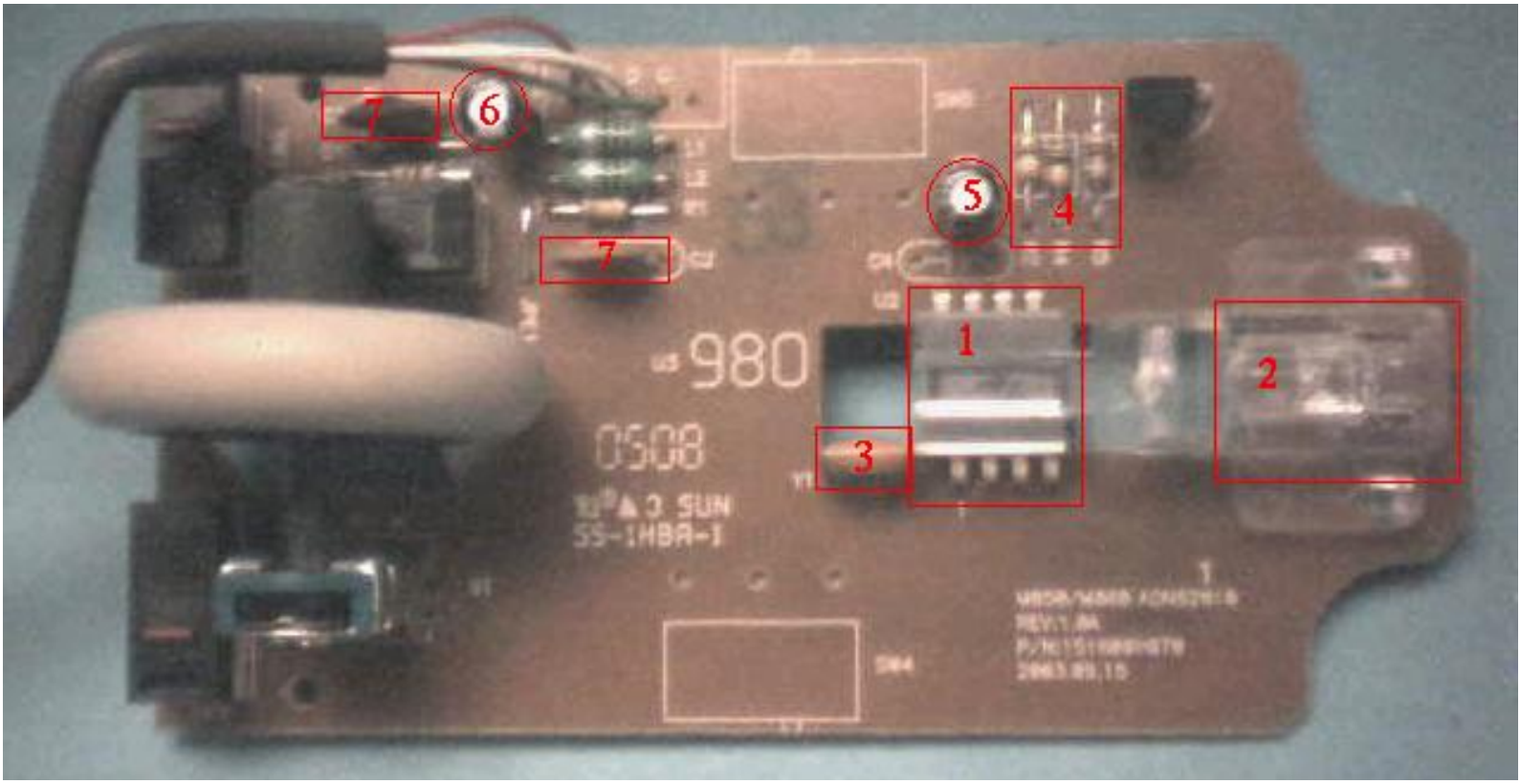


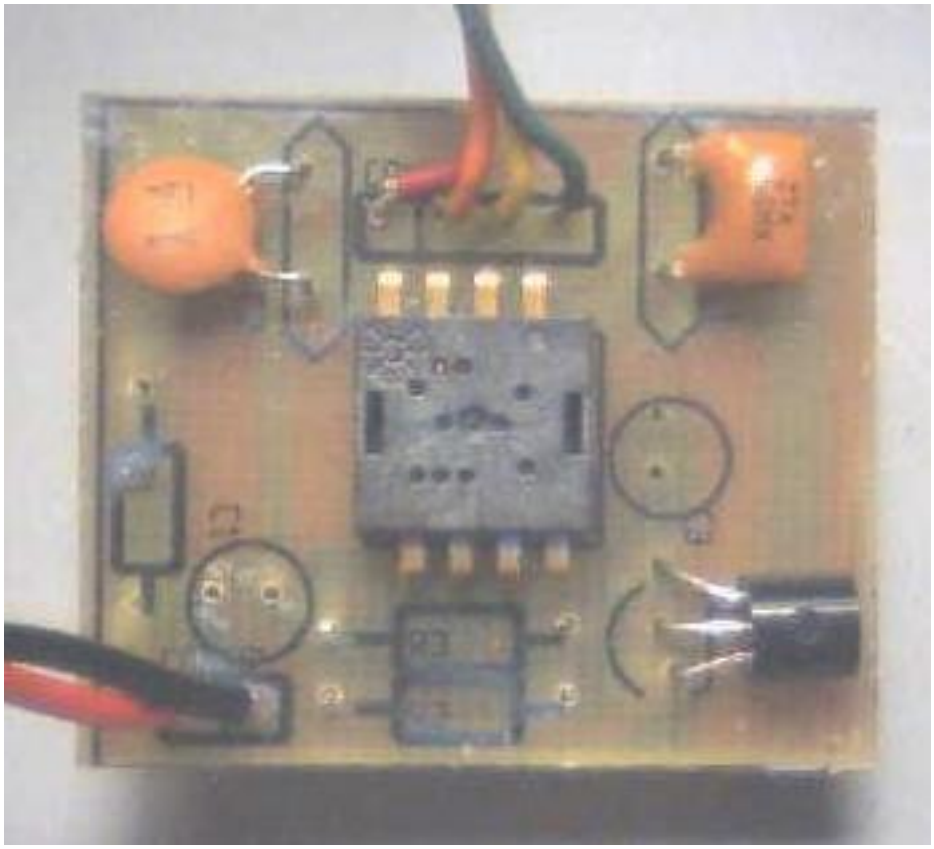


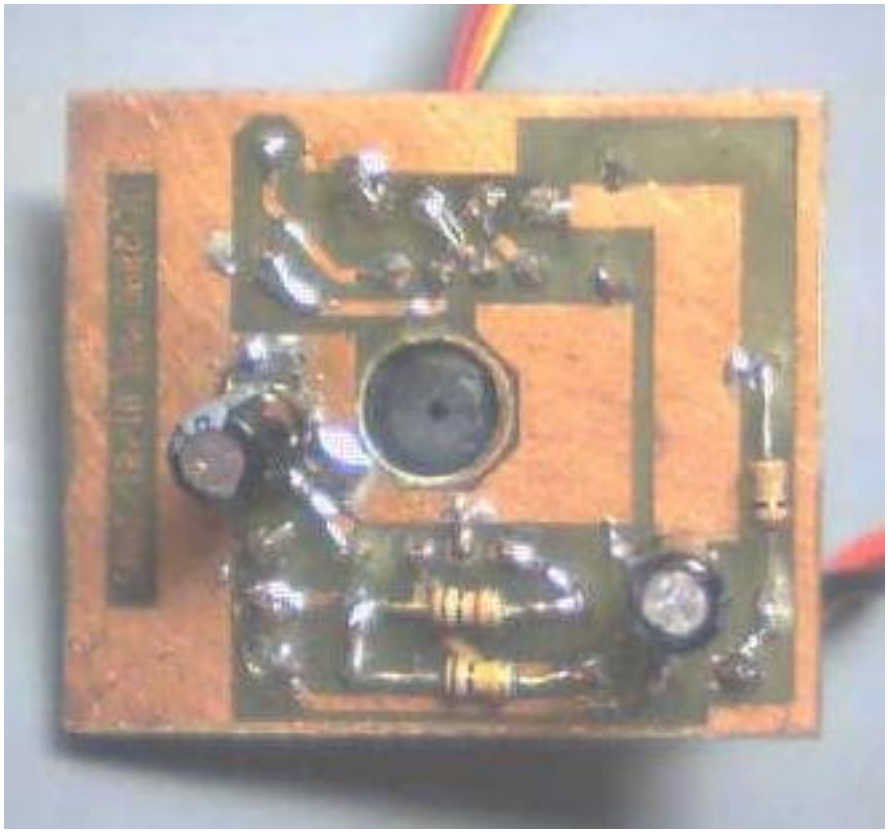







7
6
5
4
7
1
2
3
980
0608
SS-1HBA-1





Optical Mouse Parameters	Minimum	Typical	Maximum	Units
Imager Resolution for Standard Mouse		400		counts per inch
		15.75		counts per mm
Detectors along either sensor axis		18		detectors
Imager Coverage for Standard Mouse		1.14		mm
Detector pitch		63.5		micrometers
Mouse motion rate	0		12	inches/sec
	0		304.8	mm/sec
Frame rate		1500		fps
Maximum motion between successive frames			0.2032	mm/frame
Maximum counts per frame			3.2	counts/frame
Distance from object surface to lens reference plane	2.3	2.4	2.5	
Standard Mouse Object to Image Distance	8.74	8.82	8.91	
Standard Mouse lens magnification	0.85	1	1.15	n/a
Standard Mouse lens focal length	2.18	2.21	2.23	mm
Standard Mouse lens Numerical Aperture (NA)	0.1	0.13	0.16	n/a
Distance from mouse sensor lid to object surface		7.45		mm
Distance from mouse sensor lid to entrance aperture		1.42		mm
Distance from mouse sensor entrance aperture to object surface		6.03		mm
Distance from mouse sensor entrance aperture to image		2.79		mm
Mouse sensor entrance aperture diameter		0.8		mm
Mouse sensor clear aperture diameter	0.44	0.57	0.71	mm
Mouse sensor optical f-number	5	3.85	3.13	n/a
Lens focal length		15		mm
Specified distance from lens to image		30		mm
Required distance from lens to object		30		mm
Magnification		-1		n/a
Entrance pupil diameter		8.59		mm
Clear aperture diameter		6.35		mm
Optical f-number		2.36		n/a
Numerical aperture (NA)		0.21		n/a
Resolvable lines per mm (based on 1:1 sensor imaging)		8		lpm
Circle of Confusion		0.125		mm
Hyperfocal Distance		762		mm
Near Focus Limit		29.42		mm
Far Focus Limit		30.6		mm

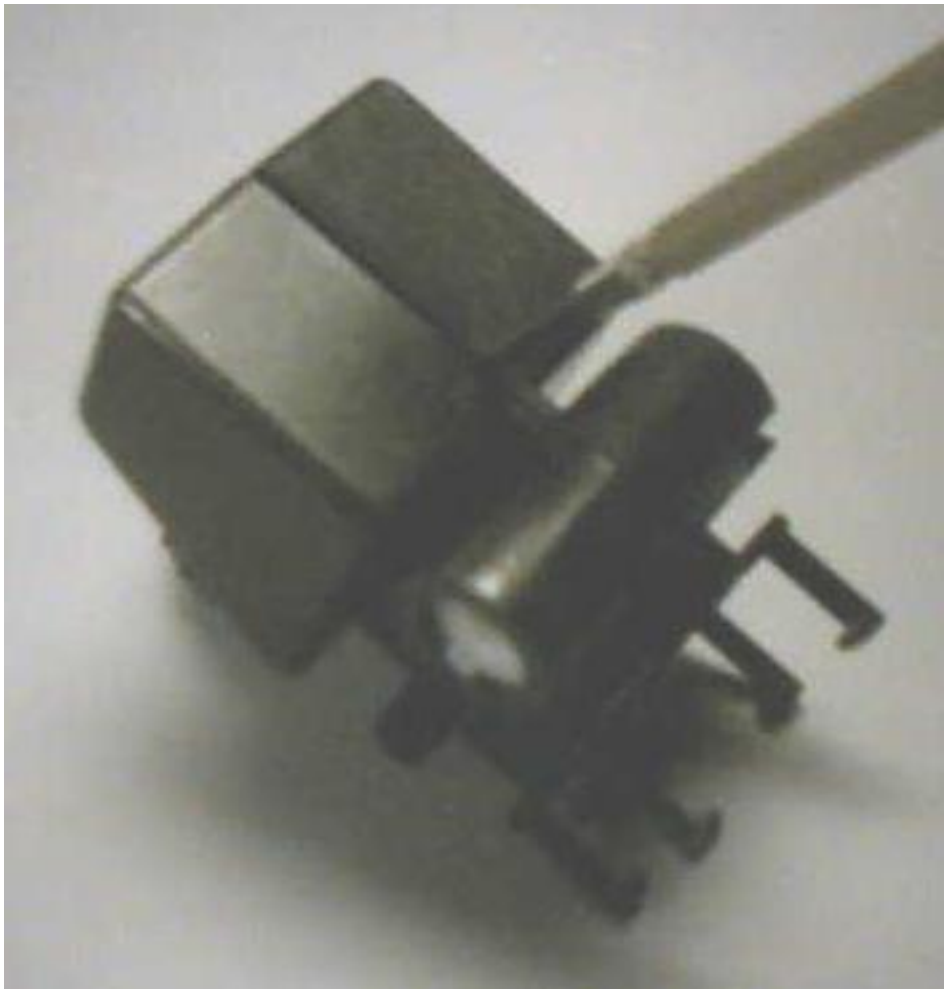
Data Sources:

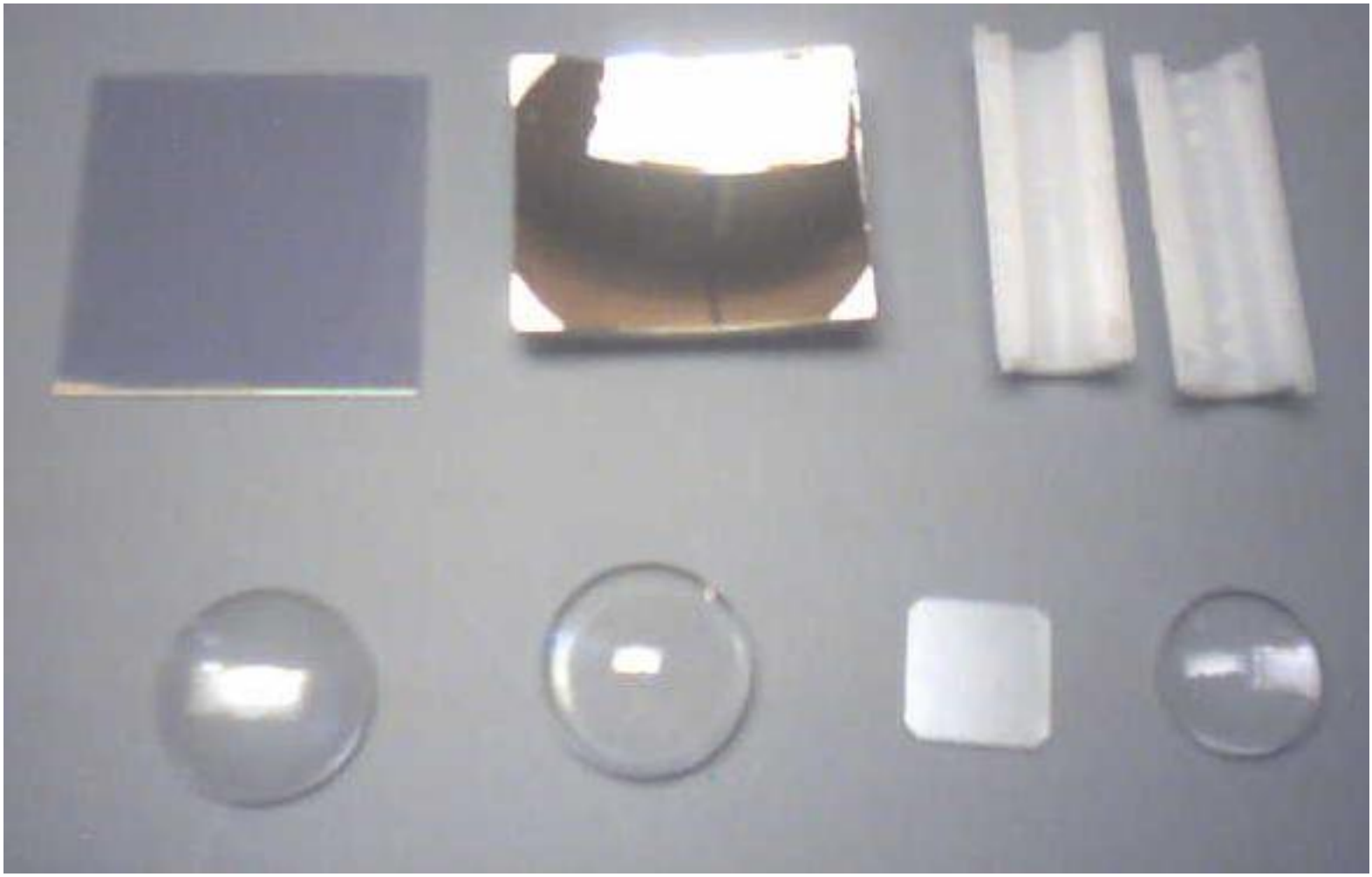
Agilent ADNS-2610 Optical Mouse Sensor

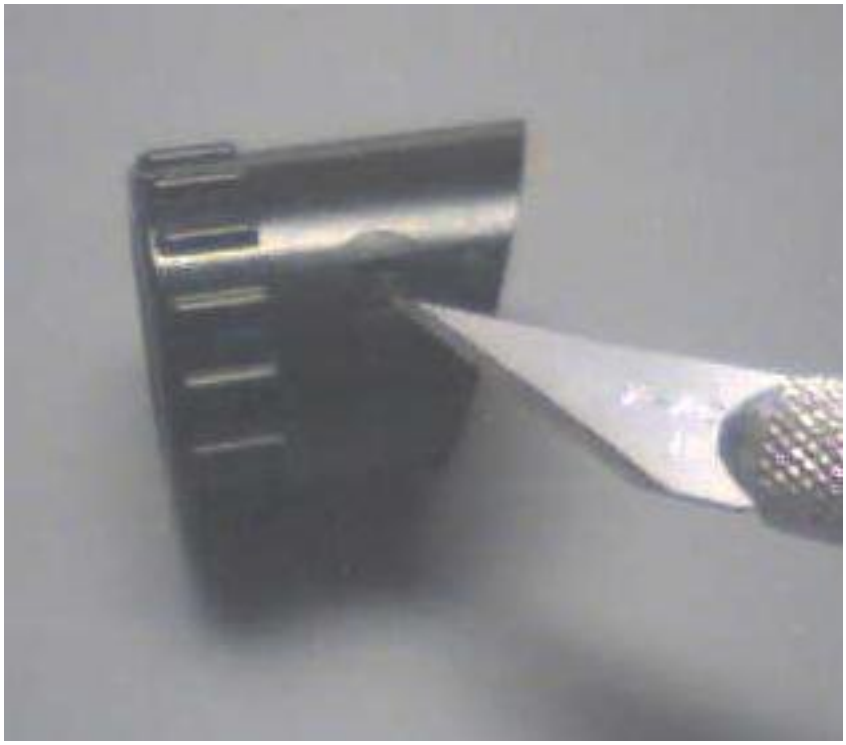
Agilent ADNS-2051 Optical Mouse Sensor (similar parameter)

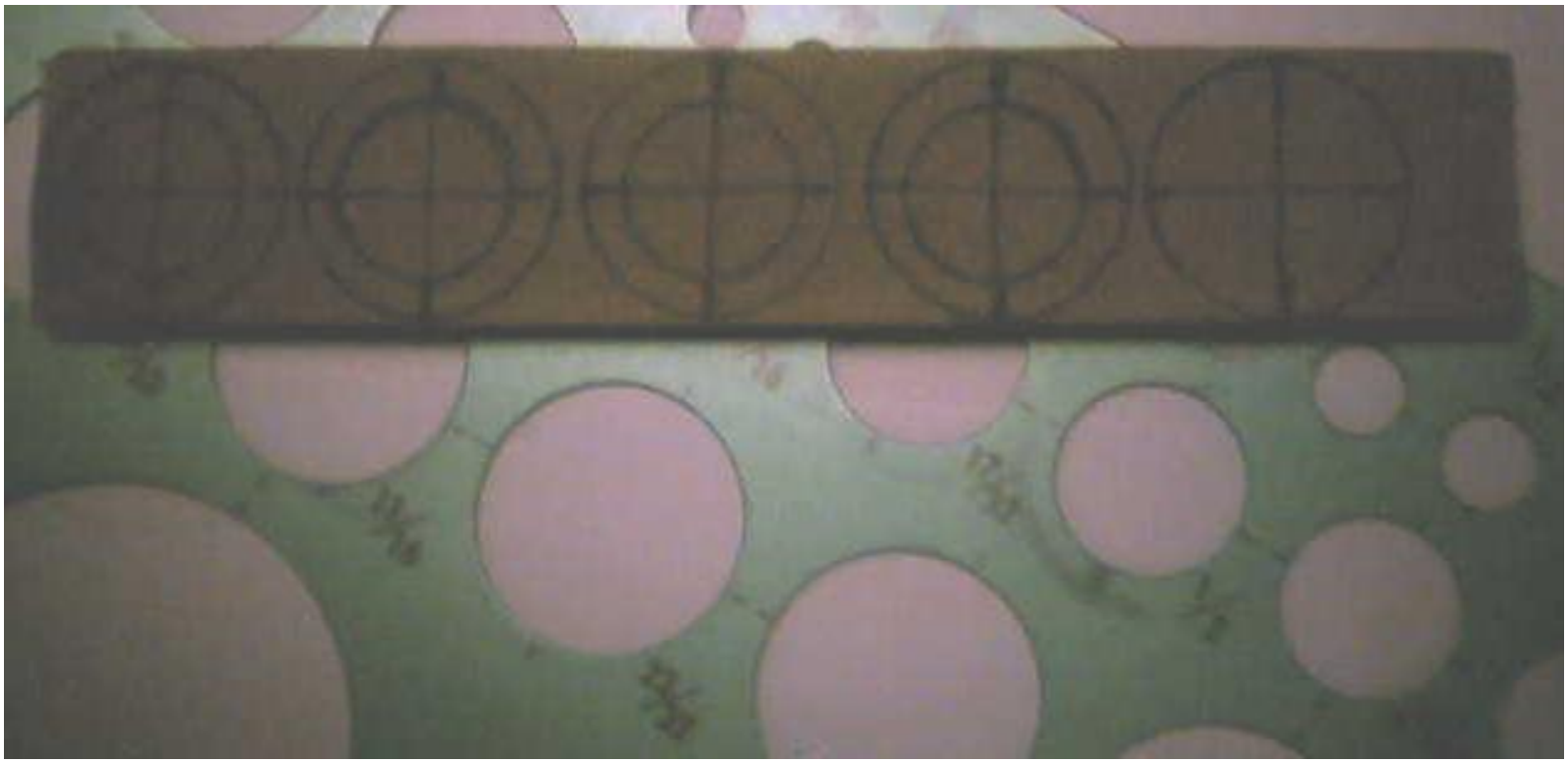
Agilent HDNS-2100 Solid-State Optical Mouse Lens

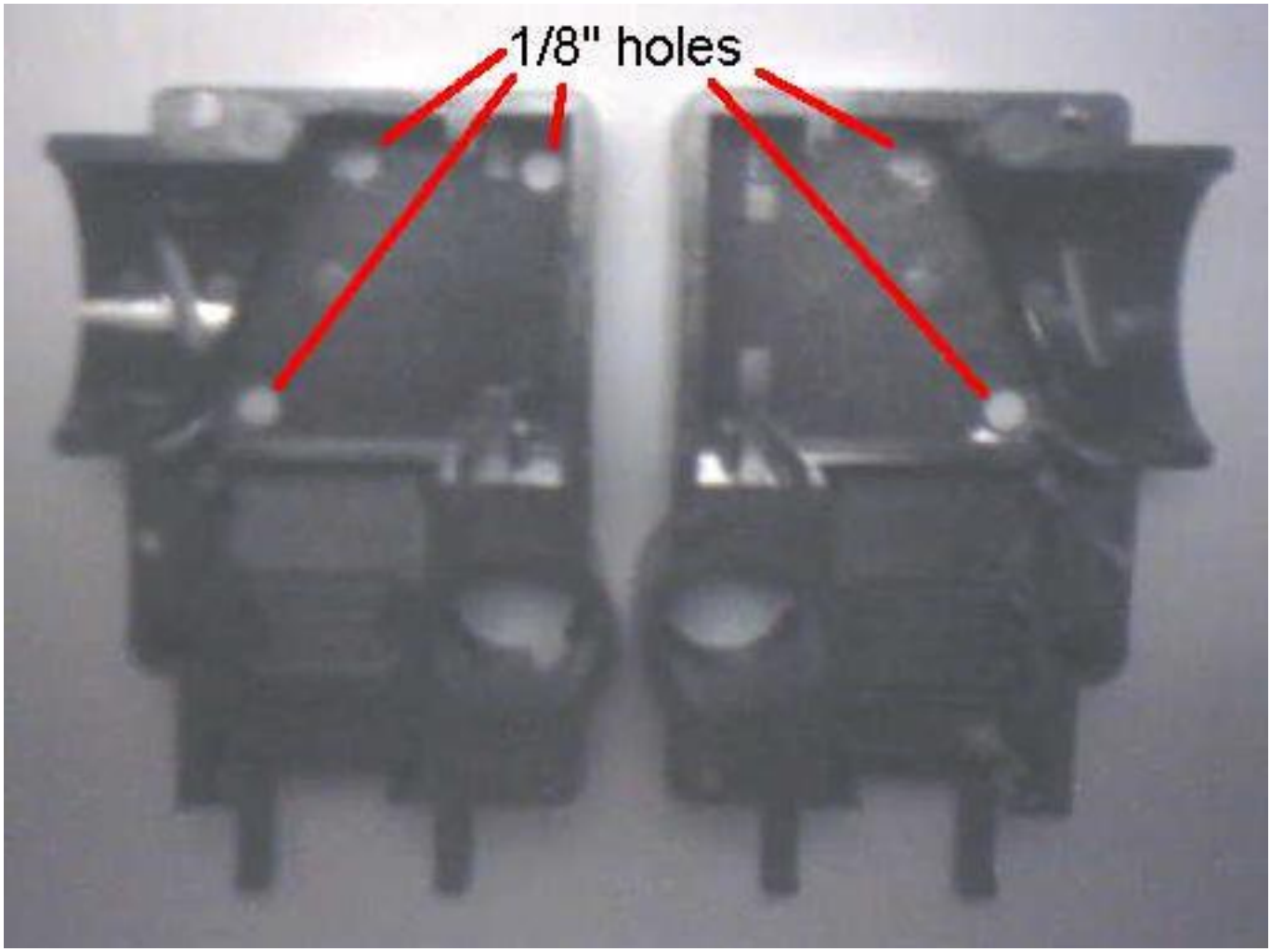

e
a
g
f
d
c
b

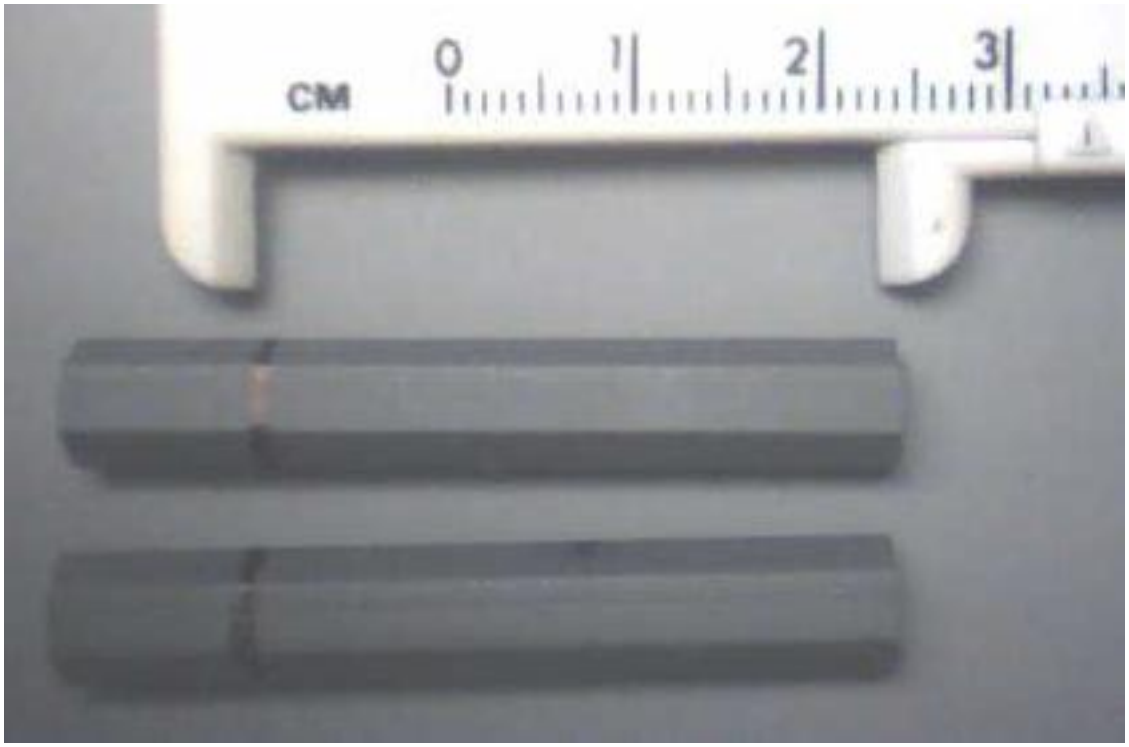


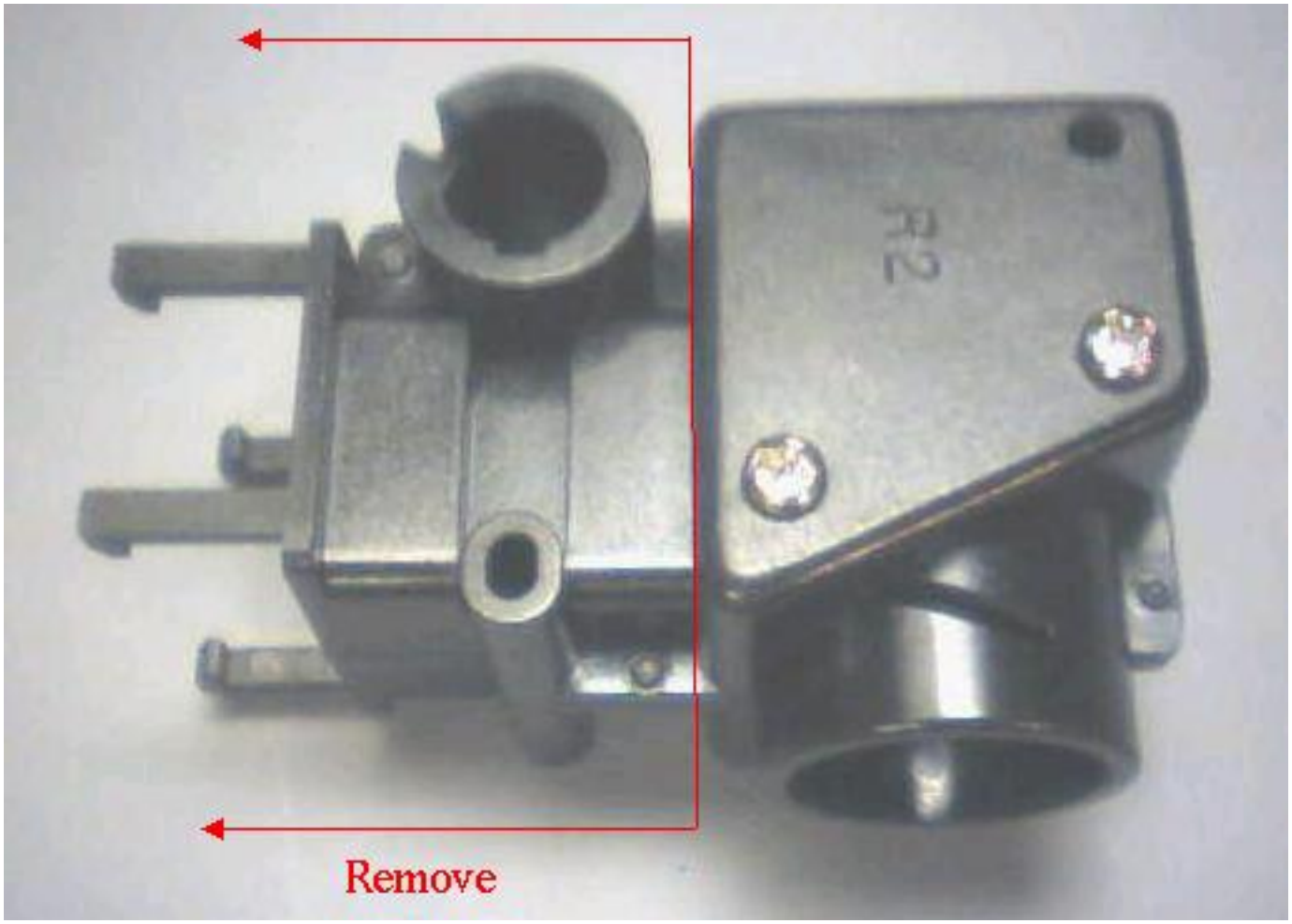


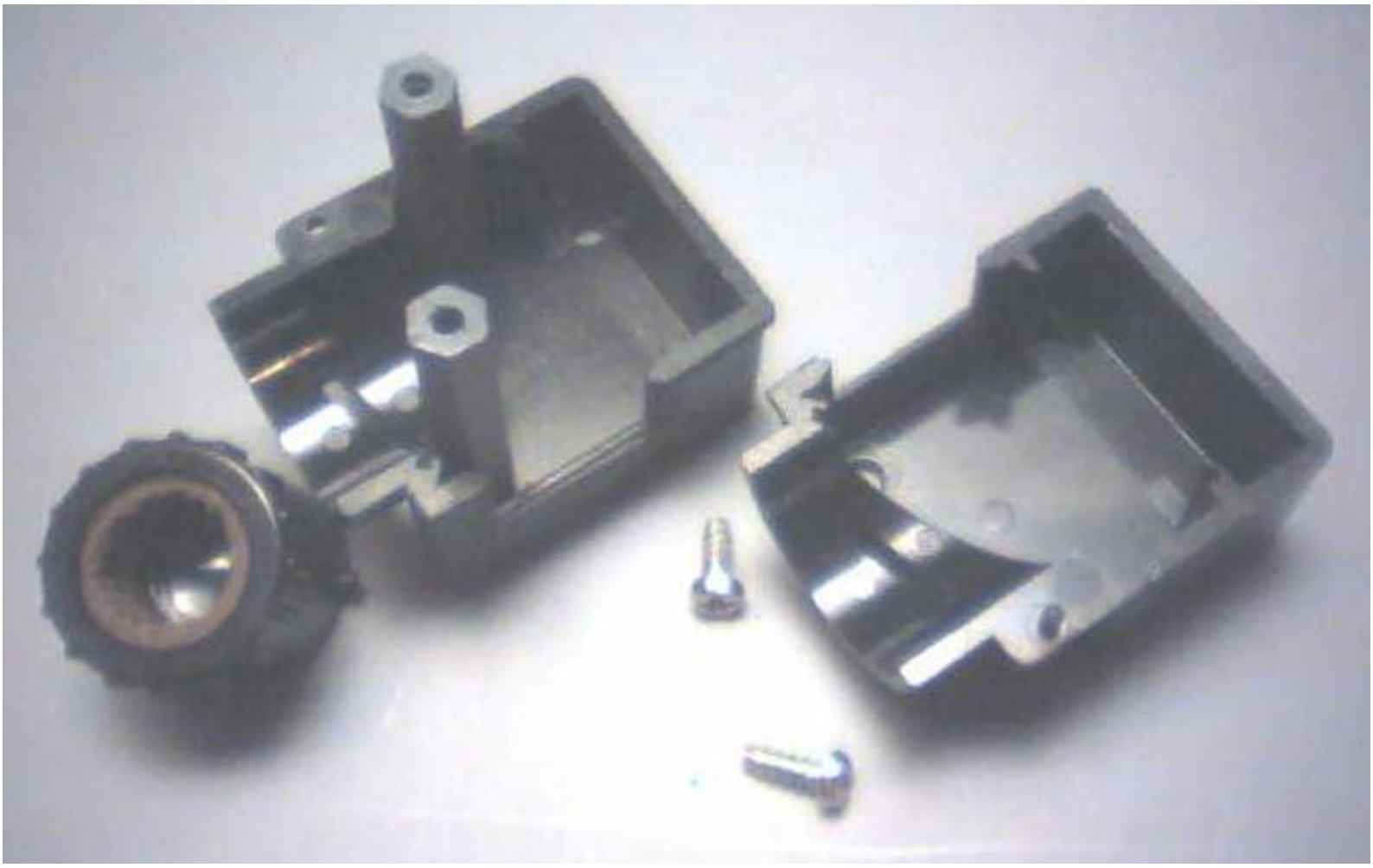


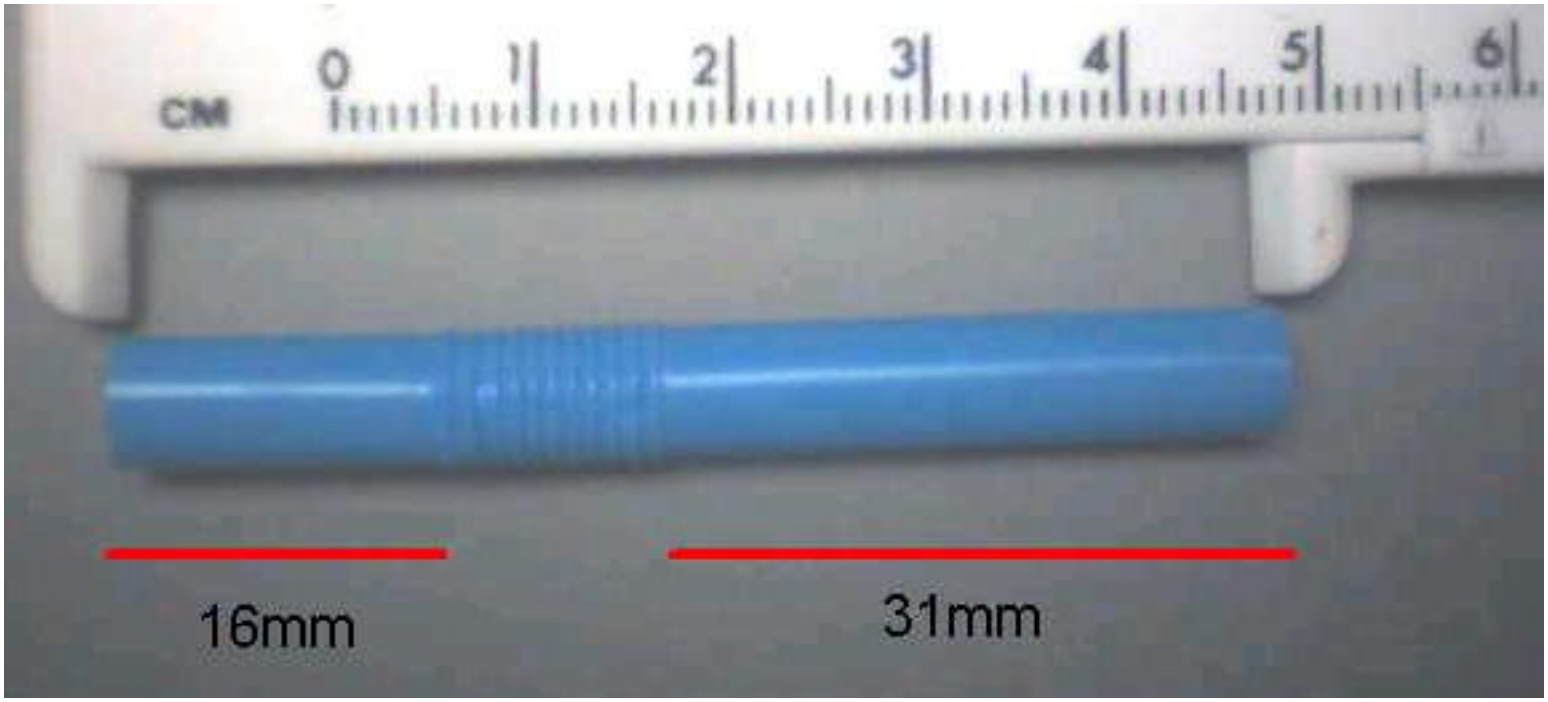


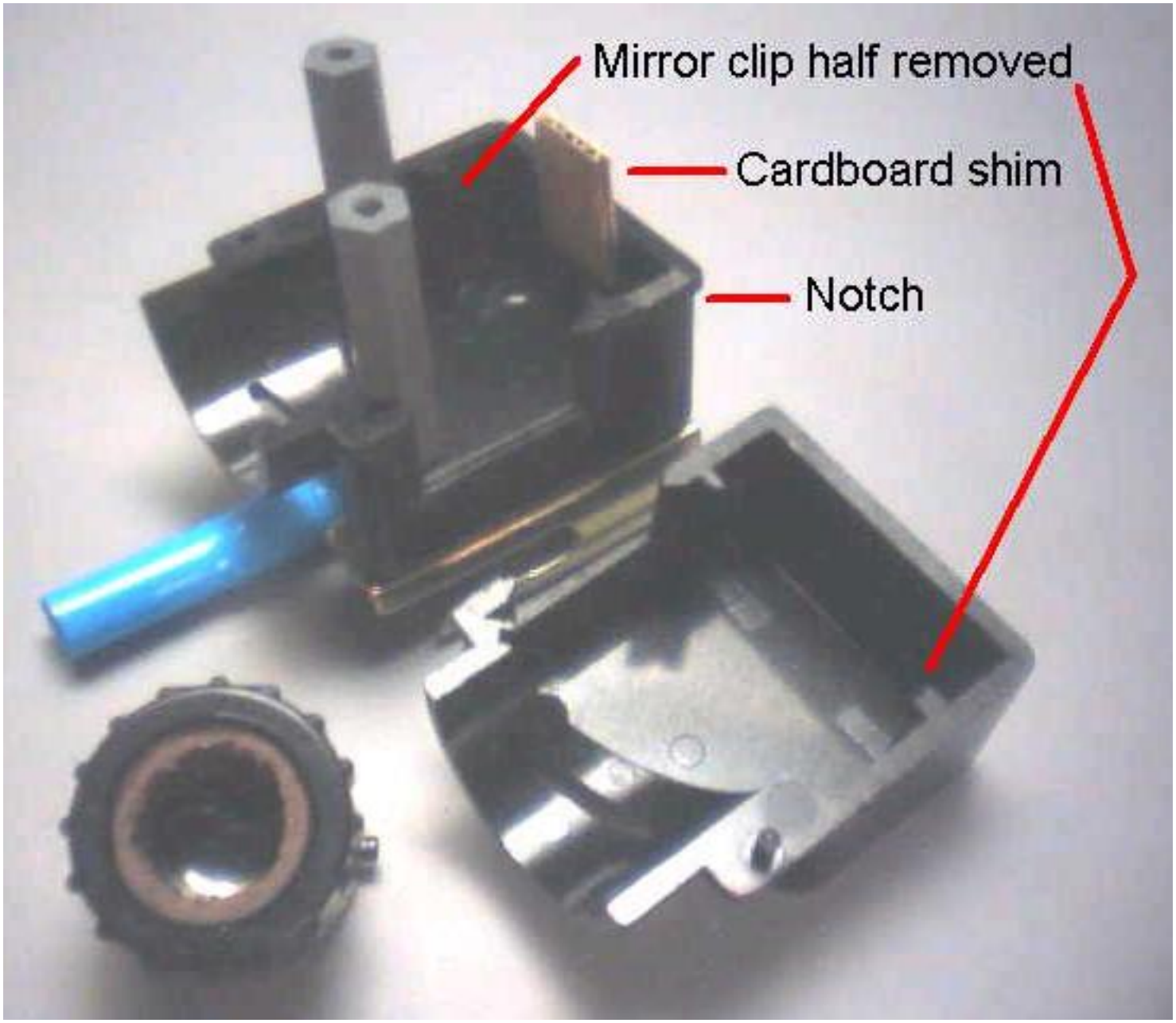

1/8" holes

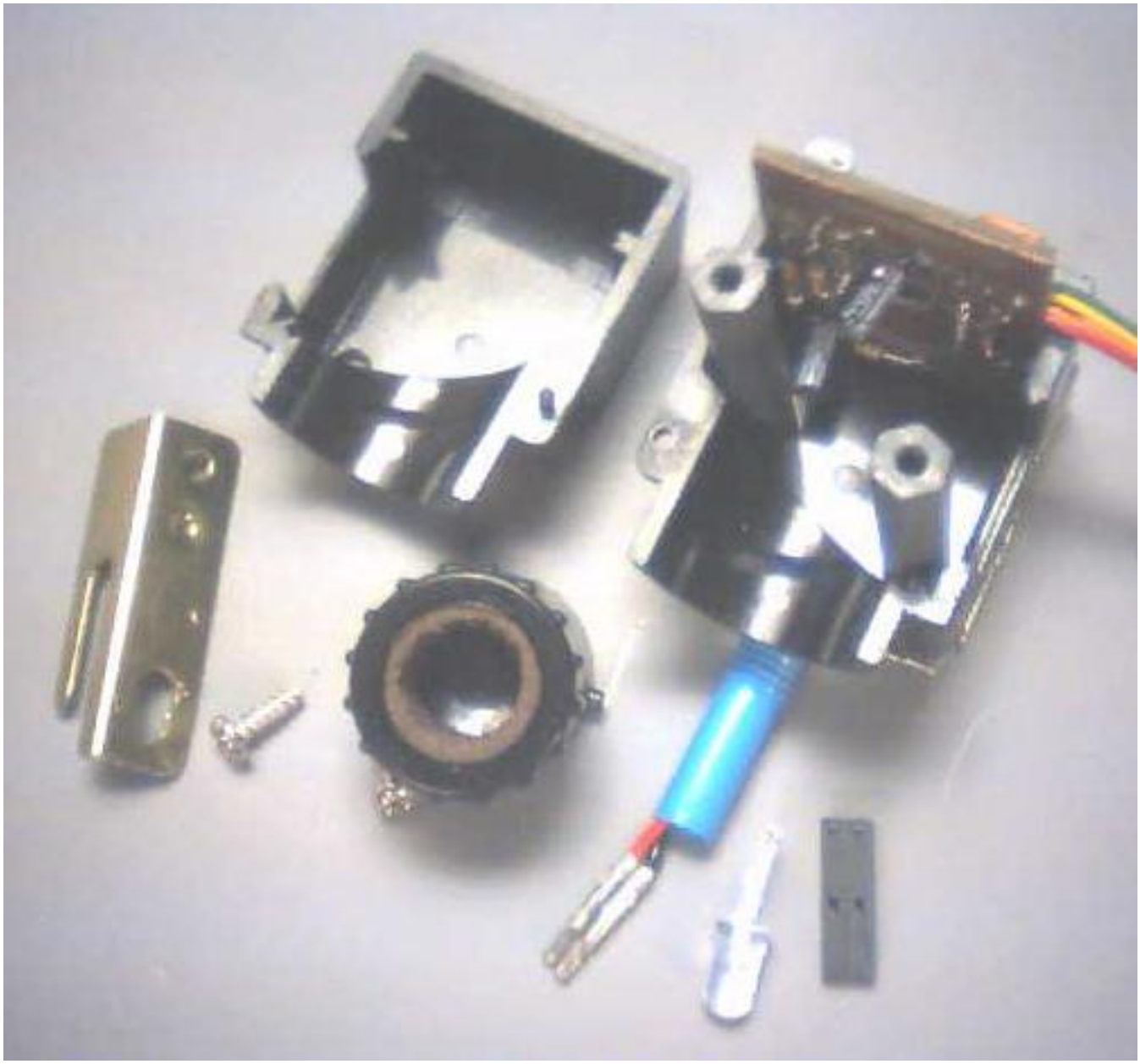

CM
0
1
2
3

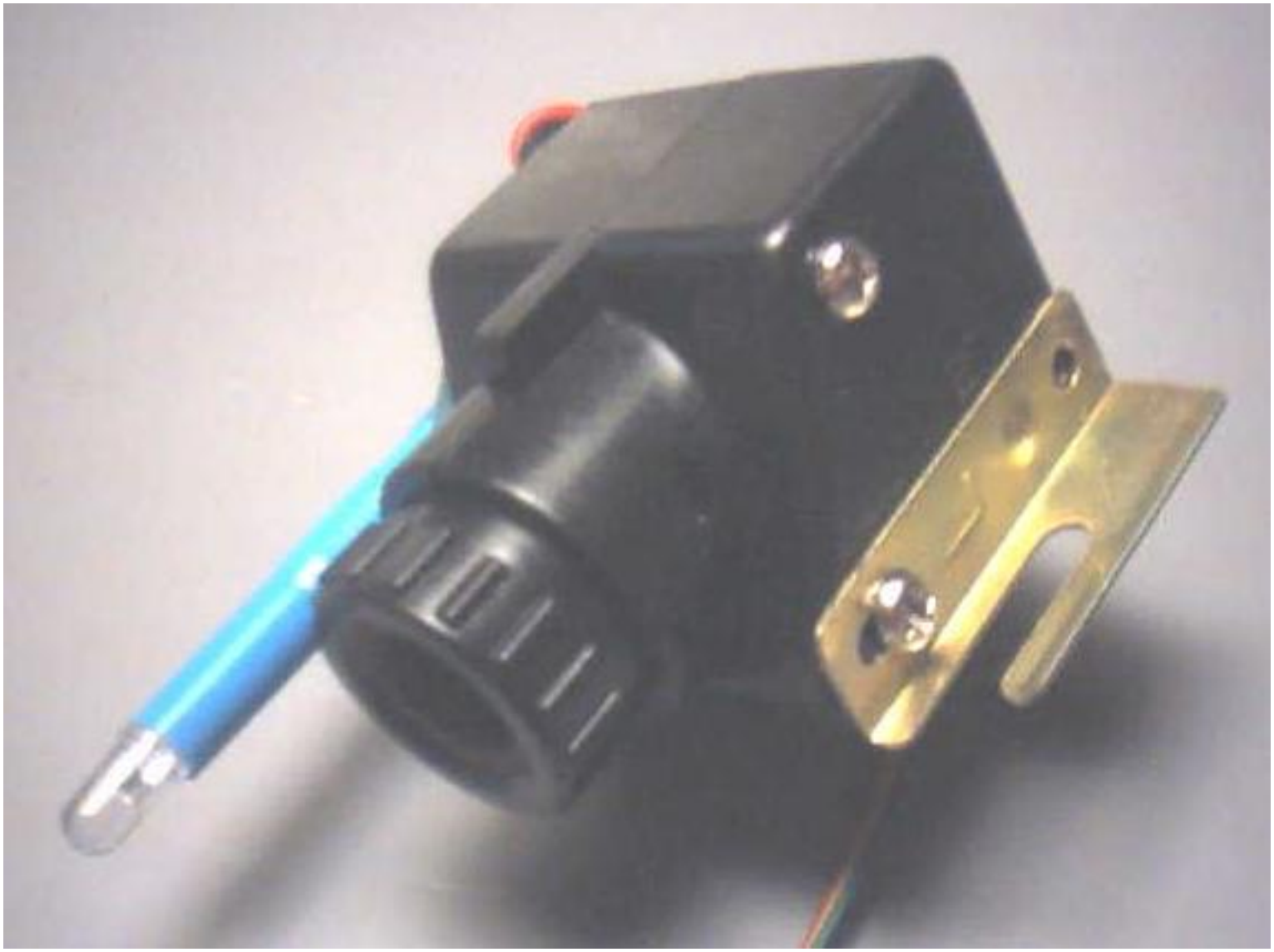

Remove

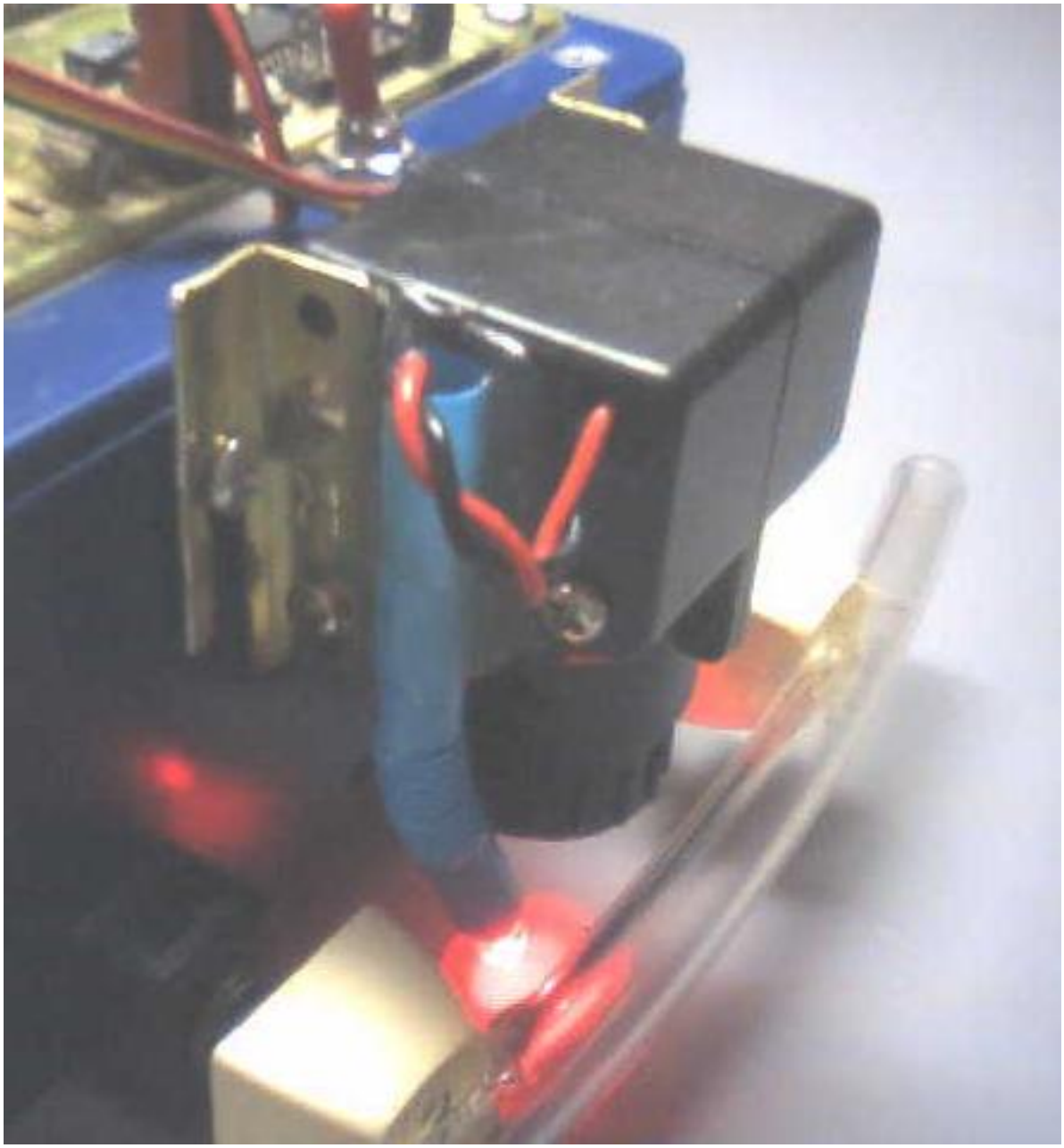



CM
0
1
2
3
4
5
6
16mm
31mm


Mirror clip half removed
Cardboard shim
Notch







```
\  Dual synchronous serial port (DSSP) dual optical encoder (DOE) interface
HEX

46FE R-TOP !
4700 DUP H ! H-FENCE !

8080 CONSTANT DSSP_OUT_BUS_BITS
0101 CONSTANT DSSP_IN_BUS_BITS
0002 CONSTANT DSSP_WRITE_EN
0001 CONSTANT DSSP_SCLK_LOW
\ 8080 CONSTANT DSSP_FRAME_START
0080 CONSTANT DSSP_FRAME_START    \ Used for testing of one optical encoder.
\ 4040 CONSTANT DSSP_DATA_VALID
0040 CONSTANT DSSP_DATA_VALID     \ Used for testing of one optical encoder.

: DSSP-WRITE-BITS ( n -- )
   DSSP_OUT_BUS_BITS AND  \ Isolate the MSB of both bytes.
   DSSP_WRITE_EN OR       \ Maintain the SSP bus drivers on.
   DUP                    \ Make a copy of the data.
   DSSP_SCLK_LOW OR       \ Bring the SSP clock line low.
   1A G!                  \ Output the data.
   0018 FOR NEXT          \ Extend duration of clock level
   1A G!                  \ Output the data, bring the SSP clock line high.
   0018 FOR NEXT          \ Extend duration of clock level
;

: DSSP-WRITE-BYTES ( n -- )
   7 FOR                  \ Enter loop to write out eight bits for each byte.
      DUP                 \ Make a copy of the data.
      DSSP-WRITE-BITS     \ Write out the MSB of both bytes.
      2*                  \ Shift the data left by one bit.
   NEXT                   \ Continue the loop.
   DROP                   \ Discard the trash data.
;

: DSSP-WRITE-REGISTERS ( data addr --  )
   DSSP_WRITE_EN 1A G!    \ Output data to turn on the SSP bus drivers.
   DSSP-WRITE-BYTES       \ Output address bytes onto SSP busses.
   DSSP-WRITE-BYTES       \ Output data bytes onto SSP busses.
   0000 1A G!             \ Output data to turn off the SSP bus drivers.
;
```

```
: DSSP-READ-BITS (  -- n )
   DSSP_SCLK_LOW 1A G!    \ Output data to bring the SSP clock line low.
   0018 FOR NEXT          \ Extend duration of clock level
   DSSP_IN_BUS_BITS       \ Set up mask to isolate the incomming bits.
   0000 1A G!             \ Output data to bring the SSP clock line high.
   0018 FOR NEXT          \ Extend duration of clock level
   1A G@                  \ Input the data.
   AND                    \ Isolate the LSB of both bytes.
;

: DSSP-READ-BYTES (  -- n )
   0                      \ Initially start with an empty bit field.
   7 FOR                  \ Enter loop to read in eight bits for each byte.
      2*                  \ Shift the data left by one bit
      DSSP-READ-BITS      \ Read in bits into LSB bit of both bytes.
      OR                  \ Combine new bits with shifted bits.
   NEXT
;

: DSSP-READ-REGISTERS ( addr -- data )
   DSSP_WRITE_EN 1A G!    \ Output data to turn on the SSP bus drivers.
   DSSP-WRITE-BYTES       \ Output address bytes onto SSP busses.
   NOP                    \ Pause for last bit before bus driver shut-off.
   0000 1A G!             \ Output data to turn off the SSP bus drivers.
   00C7 FOR NEXT          \ Loop 200 times for delay between write and read.
   DSSP-READ-BYTES        \ Input data bytes from the SSP busses.
;

: DOE-RESET ( -- )
   8080 8080
   DSSP-WRITE-REGISTERS
;

: DOE-POWER-DOWN ( -- )
   4040 8080
   DSSP-WRITE-REGISTERS
;

: DOE-KEEP-AWAKE ( -- )
   0101 8080
   DSSP-WRITE-REGISTERS
;
```

```
: DOE-LET-SLEEP ( -- )
   0000 8080
   DSSP-WRITE-REGISTERS
;

: DOE-READ-CONFIG ( -- data )
   0000
   DSSP-READ-REGISTERS
;

: DOE-STATUS ( -- data )
   0101
   DSSP-READ-REGISTERS
;

: DOE-DELTA-Y ( -- data )
   0202
   DSSP-READ-REGISTERS
;

: DOE-DELTA-X ( -- data )
   0303
   DSSP-READ-REGISTERS
;

: DOE-SQUAL ( -- data )
   0404
   DSSP-READ-REGISTERS
;

: DOE-MAX-PIXEL ( -- data )
   0505
   DSSP-READ-REGISTERS
;

: DOE-MIN-PIXEL ( -- data )
   0606
   DSSP-READ-REGISTERS
;

: DOE-PIXEL-SUM ( -- data )
   0707
   DSSP-READ-REGISTERS
```

```
;

: DOE-RESET-PIXEL-DATA ( -- )
   0000 8888
   DSSP-WRITE-REGISTERS
;

: DOE-READ-PIXEL-DATA ( addr -- code)
   DOE-RESET-PIXEL-DATA       \ Reset the pixel hardware.
   0640 FOR NEXT              \ Delay loop of 1600 loops.
   0143 FOR                   \ Loop 324 times to read in pixels
      0808                    \ Address for pixel data.
      DSSP-READ-REGISTERS     \ Read bytes.
      SWAP                    \ Bring address to TOS.
      !+2                     \ Write out values and increment by two bytes.
   NEXT                       \ Loop if all status bits are true.
   DROP                       \ Discard the address.
;

: DOE-SHUTTER-UPPER ( -- data )
   0909
   DSSP-READ-REGISTERS
;

: DOE-SHUTTER-LOWER ( -- data )
   0A0A
   DSSP-READ-REGISTERS
;

: DOE-SHUTTER-RATES ( -- data1 data0 )
   DOE-SHUTTER-UPPER          \ Obtain the upper-byte shutter rate values
   DOE-SHUTTER-LOWER          \ Obtain the lower-byte shutter rate values
   OVER                       \ Duplicate the upper-byte shutter rate data
   OVER                       \ Duplicate the lower-byte shutter rate data
   FF00                       \ Mask for the lower byte of sensor #1
   AND                        \ Isolate the lower byte of sensor #1
   2/ 2/ 2/ 2/ 2/ 2/ 2/ 2/    \ Shift the lower byte to it proper place
   SWAP                       \ Bring upper-byte shutter rate values to TOS
   FF00                       \ Mask for the upper byte of sensor #1
   AND                        \ Isolate the upper byte of sensor #1
   AND                        \ Combine with lower byte for sensor #1 value
   -ROT                       \ Rotate sensor #1 data into the stack
   00FF                       \ Mask for the lower byte of sensor #0
```

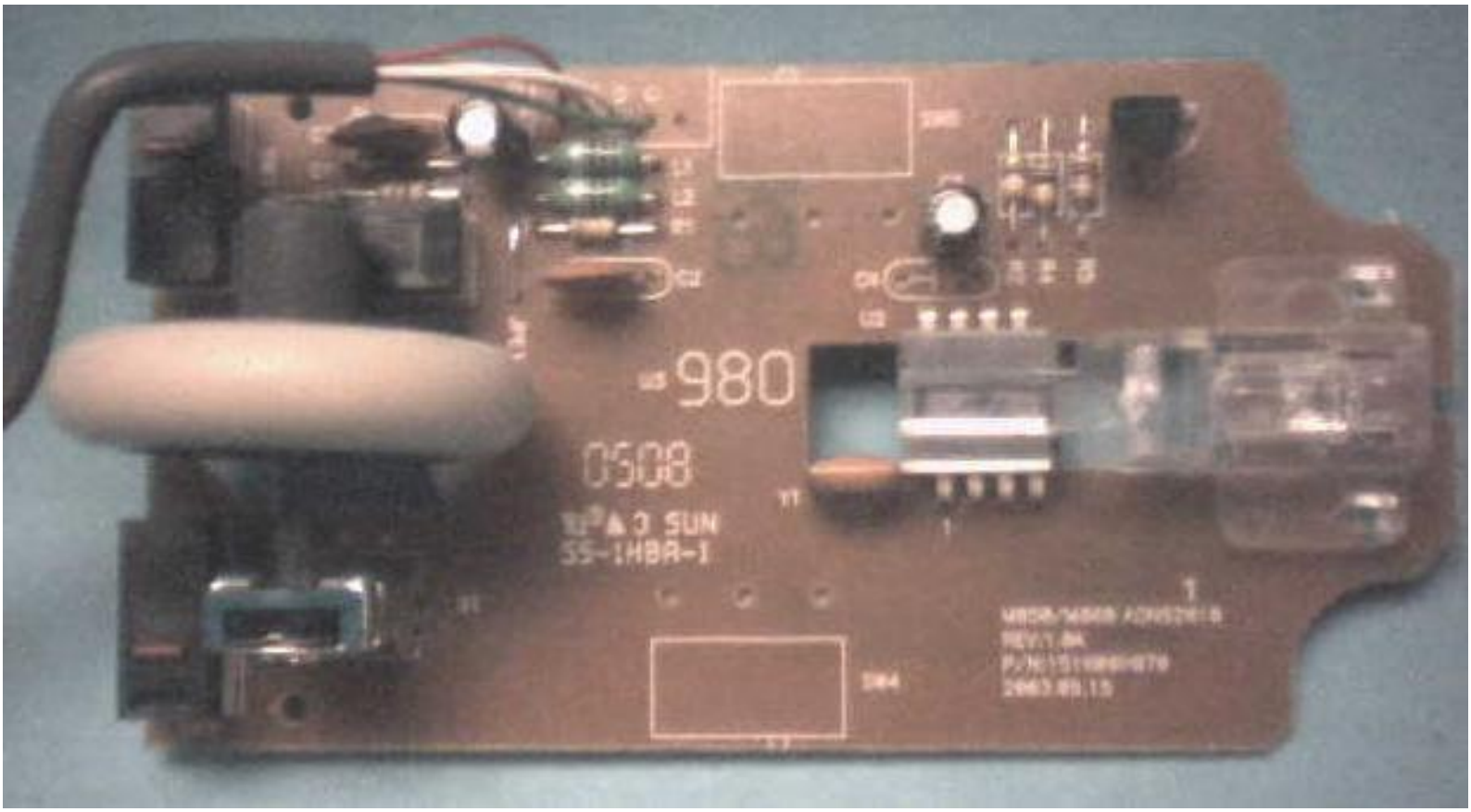
```
   AND                  \ Isolate the lower byte of sensor #0
   SWAP                  \ Bring upper-byte shutter rate values to TOS
   00FF                 \ Mask for the lower byte of sensor #0
   AND                  \ Isolate the upper byte of sensor #0
   2* 2* 2* 2* 2* 2* 2* 2*     \ Shift the upper byte to it proper place
   AND                  \ Combine with lower byte for sensor #0 value
;

: DOE-INV-PRODUCT-ID ( -- data )
   1111
   DSSP-READ-REGISTERS
;

: CRUDE-FOCUS ( n -- )
   0 DO
      DOE-SQUAL
      00FF AND
      2 .H CR
   LOOP
;

: PIXEL-DUMP ( addr -- )
   ." {" CR              \ Output initial open brace for data.
   12 0 DO               \ Eighteen rows in the image.
      ." {"             \ Output open brace for row.
      12 0 DO            \ Eighteen pixels in each row.
         @+2             \ Read data value and increment pointer by two.
         SWAP            \ Bring data value to TOS.
         003F AND         \ Mask to clear unused bits.
         2* 2*           \ Shift left by two to increase dynamic range.
         ."  #"          \ Output color triplet indicator
         DUP 2 .H        \ Output first copy of byte for red color.
         DUP 2 .H        \ Output second copy of byte for green color.
         2 .H ." "       \ Output third copy of byte for blue color.
      LOOP
      ."  }" CR          \ Output close brace for row and newline.
   LOOP
   ." }" CR              \ Output close brace for data and newline.
;

: GET-PIXELS
   R @ DOE-READ-PIXEL-DATA
   R @ PIXEL-DUMP
```

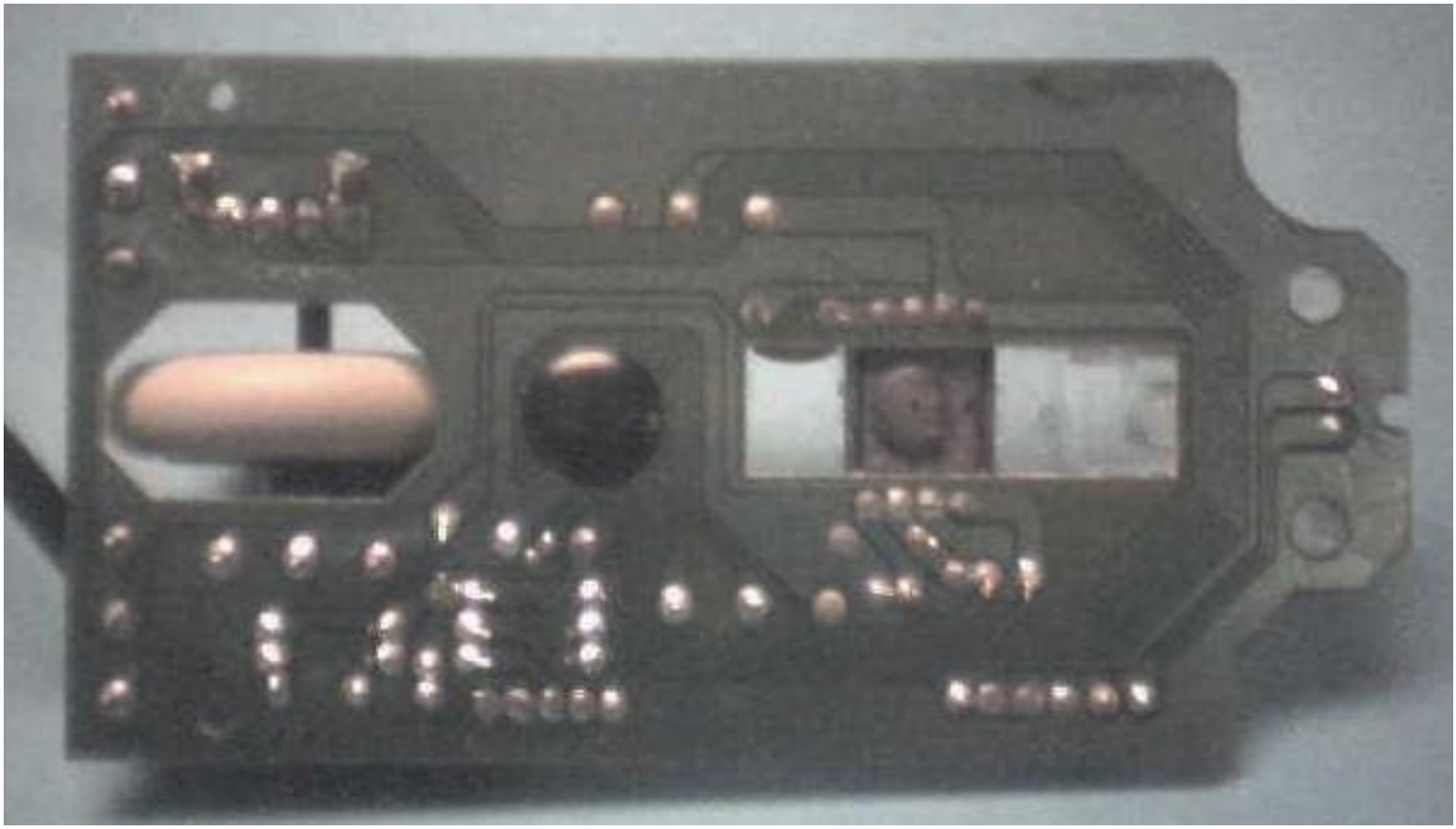
```
;

: RAW-PIXEL-DUMP ( addr -- )
   12 0 DO                \ Eighteen rows in the image.
      12 0 DO             \ Eighteen pixels in each row.
         @+2              \ Read data value and increment pointer by two.
         SWAP             \ Bring data value to TOS.
         00FF AND         \ Mask to clear unused bits.
         2 .H ."  "       \ Output  byte.
      LOOP
      CR                  \ Output newline.
   LOOP
   CR                     \ Output newline.
;

: GET-RAW-PIXELS
   R @ DOE-READ-PIXEL-DATA
   R @ RAW-PIXEL-DUMP
;

: MOVE-TEST
   0 DO
      DOE-DELTA-X
      00FF
      AND
      DUP
      0080
      AND
      0>
      IF
         FF00
         OR
      THEN
      ." Delta X: " .
      DOE-DELTA-Y
      00FF
      AND
      DUP
      0080
      AND
      0>
      IF
         FF00
```

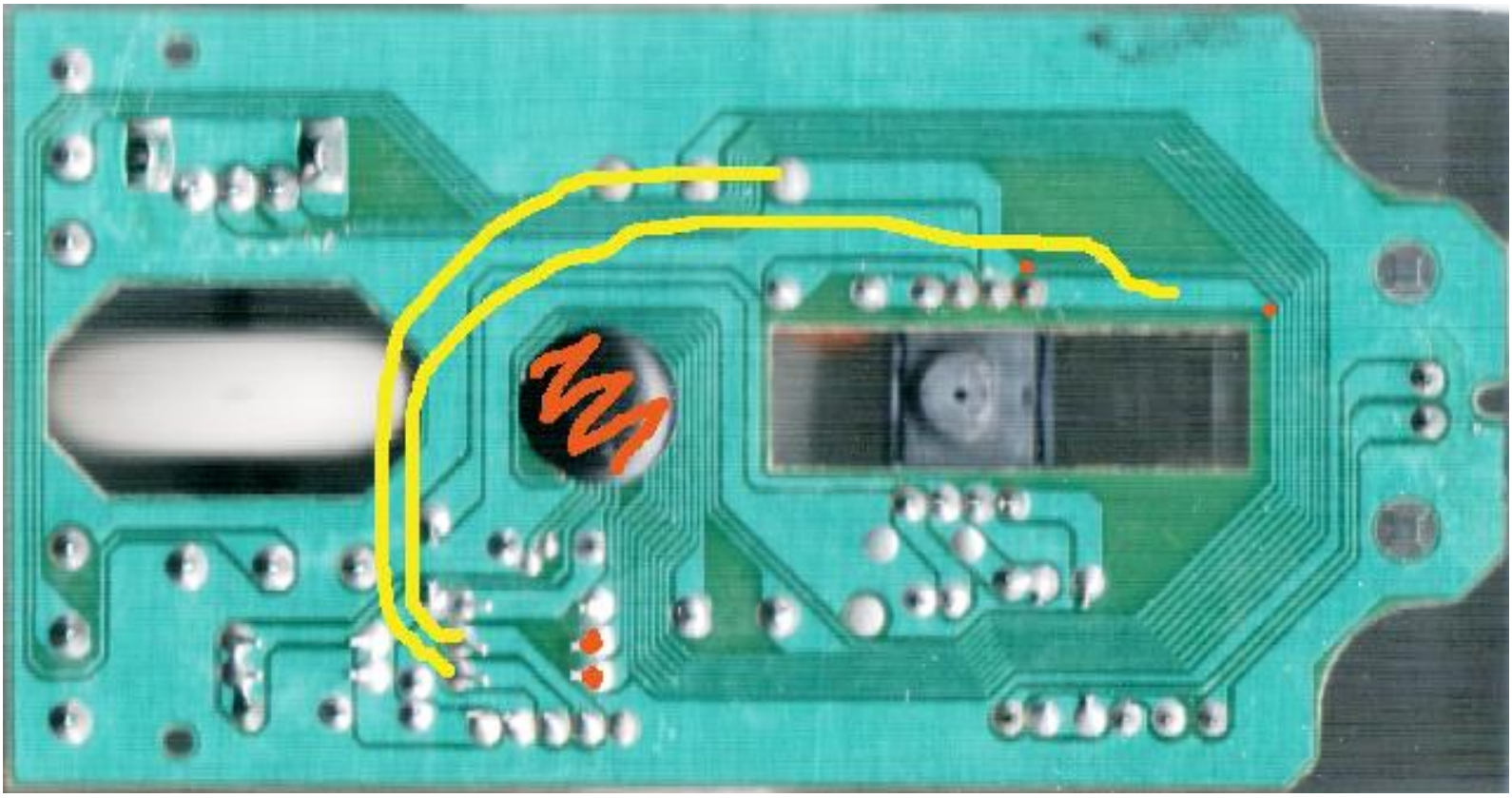
```
        OR
      THEN
      ." Delta Y: " . CR
   LOOP
;

: CALIBRATE-X-Y
   DOE-DELTA-X
   00FF
   AND
   DUP
   0080
   AND
   0>
   IF
      FF00
      OR
   THEN
   DECIMAL
   ." Delta X: " .
   HEX
   DOE-DELTA-Y
   00FF
   AND
   DUP
   0080
   AND
   0>
   IF
      FF00
      OR
   THEN
   DECIMAL
   ." Delta Y: " . CR
   HEX
;
```

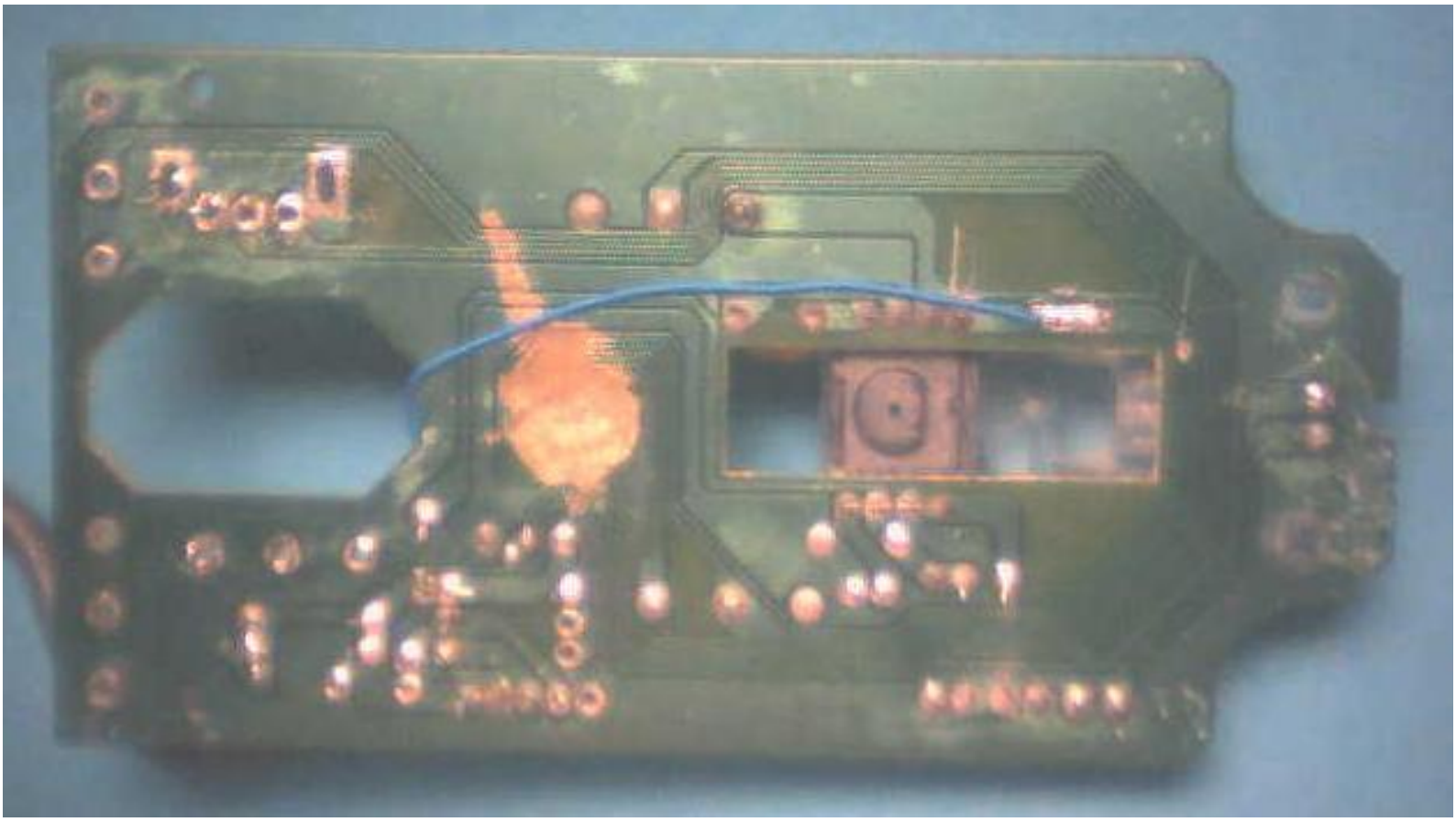

3D Optical Mouse
M850
Bringing elegance to the use of a computer mouse
New sensation, new satisfaction
Agilent enabled
BTC M850 OPT $ 6.99
BTC M850 OPTICAL MOUSE
2-BUTTON OPTICAL MOUSE
WITH SCROLL WHEEL
PS/2 COMPATIBLE
2:340
3521924

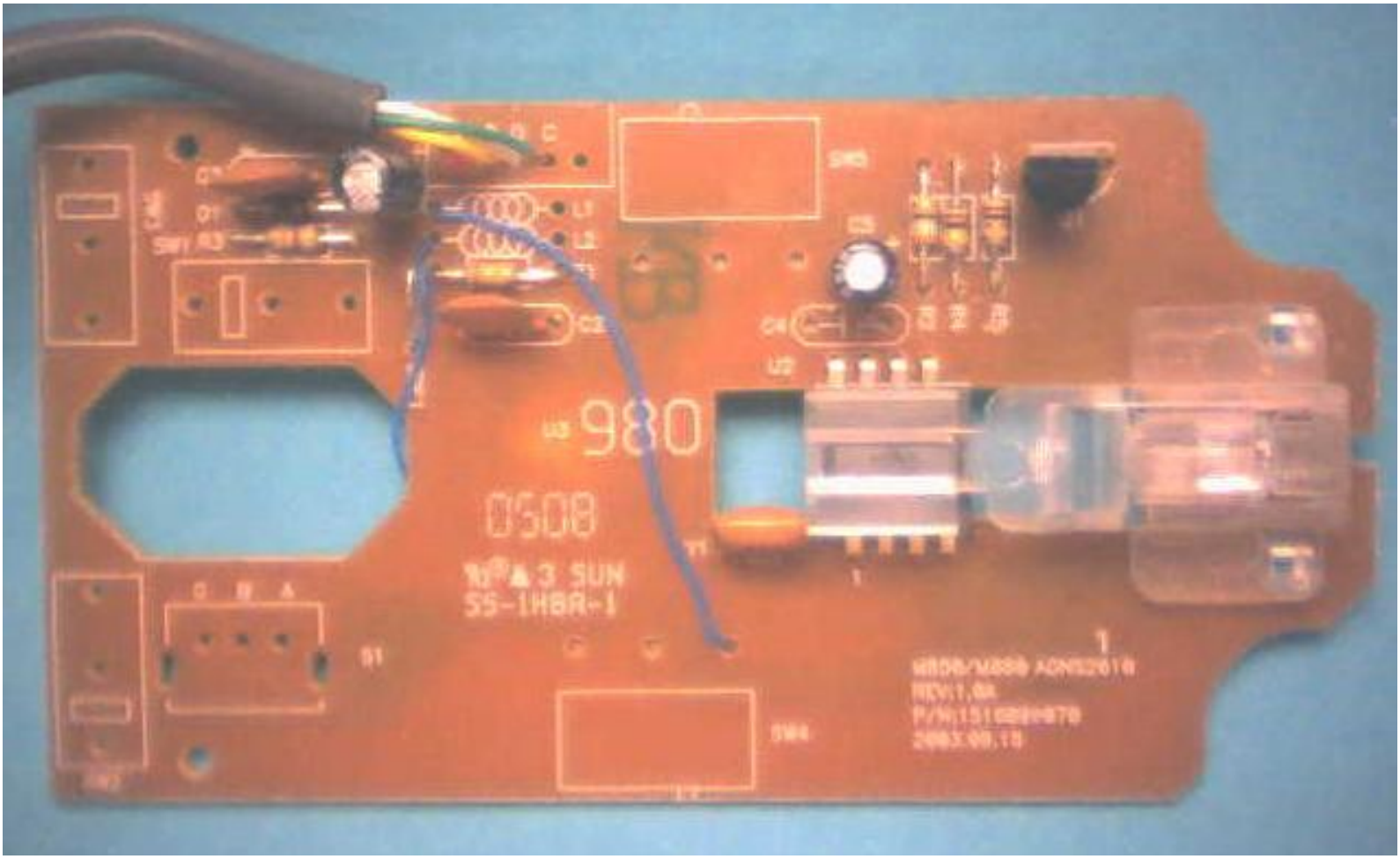

BTC


980
0608
SS-1HBR-1








980
0508
SS-1H8R-1

```
\  Dual synchronous serial port (DSSP) dual optical encoder (DOE) interface
HEX

46FE R-TOP !
4700 DUP H ! H-FENCE !

8080 CONSTANT DSSP_OUT_BUS_BITS
0101 CONSTANT DSSP_IN_BUS_BITS
0002 CONSTANT DSSP_WRITE_EN
0001 CONSTANT DSSP_SCLK_LOW
\ 8080 CONSTANT DSSP_FRAME_START
0080 CONSTANT DSSP_FRAME_START    \ Used for testing of one optical encoder.
\ 4040 CONSTANT DSSP_DATA_VALID
0040 CONSTANT DSSP_DATA_VALID     \ Used for testing of one optical encoder.

: DSSP_WRITE_BITS ( n -- )
   DSSP_OUT_BUS_BITS AND  \ Isolate the MSB of both bytes.
   DSSP_WRITE_EN OR       \ Maintain the SSP bus drivers on.
   DUP                    \ Make a copy of the data.
   DSSP_SCLK_LOW OR       \ Bring the SSP clock line low.
   0018 FOR NEXT          \ Extend duration of clock level
   1A G!                  \ Output the data.
   NOP NOP                \ Delay two clock cycles.
   1A G!                  \ Output the data, bring the SSP clock line high.
   0018 FOR NEXT          \ Extend duration of clock level
;

: DSSP_WRITE_BYTES ( n -- )
   7 FOR                  \ Enter loop to write out eight bits for each byte.
      DUP                 \ Make a copy of the data.
      DSSP_WRITE_BITS     \ Write out the MSB of both bytes.
      2*                  \ Shift the data left by one bit.
   NEXT                   \ Continue the loop.
   DROP                   \ Discard the trash data.
;

: DSSP_WRITE_REGISTERS ( data addr --  )
   DSSP_WRITE_EN 1A G!    \ Output data to turn on the SSP bus drivers.
   DSSP_WRITE_BYTES       \ Output address bytes onto SSP busses.
   DSSP_WRITE_BYTES       \ Output data bytes onto SSP busses.
   0000 1A G!             \ Output data to turn off the SSP bus drivers.
;
```

```
: DSSP_READ_BITS (  -- n )
   DSSP_SCLK_LOW 1A G!    \ Output data to bring the SSP clock line low.
   0018 FOR NEXT          \ Extend duration of clock level
   DSSP_IN_BUS_BITS       \ Set up mask to isolate the incomming bits.
   0000 1A G!             \ Output data to bring the SSP clock line high.
   0018 FOR NEXT          \ Extend duration of clock level
   1A G@                  \ Input the data.
   AND                    \ Isolate the LSB of both bytes.
;

: DSSP_READ_BYTES (  -- n )
   0                      \ Initially start with an empty bit field.
   7 FOR                  \ Enter loop to read in eight bits for each byte.
      2*                  \ Shift the data left by one bit
      DSSP_READ_BITS      \ Read in bits into LSB bit of both bytes.
      OR                  \ Combine new bits with shifted bits.
   NEXT
;

: DSSP_READ_REGISTERS ( addr -- data )
   DSSP_WRITE_EN 1A G!    \ Output data to turn on the SSP bus drivers.
   DSSP_WRITE_BYTES       \ Output address bytes onto SSP busses.
   NOP                    \ Pause for last bit before bus driver shut-off.
   0000 1A G!             \ Output data to turn off the SSP bus drivers.
   00C7 FOR NEXT          \ Loop 200 times for delay between write and read.
   DSSP_READ_BYTES        \ Input data bytes from the SSP busses.
;

: DOE_RESET ( -- )
   8080 8080
   DSSP_WRITE_REGISTERS
;

: DOE_POWER_DOWN ( -- )
   4040 8080
   DSSP_WRITE_REGISTERS
;

: DOE_KEEP_AWAKE ( -- )
   0101 8080
   DSSP_WRITE_REGISTERS
;
```

```
: DOE_LET_SLEEP ( -- )
   0000 8080
   DSSP_WRITE_REGISTERS
;

: DOE_CONFIG_PORTS ( -- data )
   0000
   DSSP_READ_REGISTERS
;

: DOE_STATUS ( -- data )
   0101
   DSSP_READ_REGISTERS
;

: DOE_DELTA_Y ( -- data )
   0202
   DSSP_READ_REGISTERS
;

: DOE_DELTA_X ( -- data )
   0303
   DSSP_READ_REGISTERS
;

: DOE_SQUAL ( -- data )
   0404
   DSSP_READ_REGISTERS
;

: DOE_MAX_PIXEL ( -- data )
   0505
   DSSP_READ_REGISTERS
;

: DOE_MIN_PIXEL ( -- data )
   0606
   DSSP_READ_REGISTERS
;

: DOE_PIXEL_SUM ( -- data )
   0707
```

```
   DSSP_READ_REGISTERS
;

: DOE_RESET_PIXEL_DATA ( -- )
   0000 8888
   DSSP_WRITE_REGISTERS
;

: DOE_READ_PIXEL_DATA ( addr -- code)
   DOE_RESET_PIXEL_DATA       \ Reset the pixel hardware.
   0640 FOR NEXT              \ Delay loop of 1600 loops.
   0143 FOR                   \ Loop 324 times to read in pixels
      0808                    \ Address for pixel data.
      DSSP_READ_REGISTERS     \ Read bytes.
      SWAP                    \ Bring address to TOS.
      !+2                     \ Write out values and increment by two bytes.
   NEXT                       \ Loop if all status bits are true.
   DROP                       \ Discard the address.
;

: DOE_SHUTTER_UPPER ( -- data )
   0909
   DSSP_READ_REGISTERS
;

: DOE_SHUTTER_LOWER ( -- data )
   0A0A
   DSSP_READ_REGISTERS
;

: DOE_INV_PRODUCT_ID ( -- data )
   1111
   DSSP_READ_REGISTERS
;

: CRUDE_FOCUS ( n -- )
   0 DO
      DOE_SQUAL
      00FF AND
      2 .H CR
   LOOP
;
```

```
: PIXEL-DUMP ( addr -- )
   ." {" CR              \ Output initial open brace for data.
   12 0 DO               \ Eighteen rows in the image.
      ." {"              \ Output open brace for row.
      12 0 DO            \ Eighteen pixels in each row.
         @+2             \ Read data value and increment pointer by two.
         SWAP            \ Bring data value to TOS.
         003F AND        \ Mask to clear unused bits.
         2* 2*           \ Shift left by two to increase dynamic range.
         ."  #"          \ Output color triplet indicator
         DUP 2 .H        \ Output first copy of byte for red color.
         DUP 2 .H        \ Output second copy of byte for green color.
         2 .H ." "       \ Output third copy of byte for blue color.
      LOOP
      ."  }" CR          \ Output close brace for row and newline.
   LOOP
   ." }" CR              \ Output close brace for data and newline.
;

: GET-PIXELS
   R @ DOE_READ_PIXEL_DATA
   R @ PIXEL-DUMP
;

: RAW-PIXEL-DUMP ( addr -- )
   12 0 DO               \ Eighteen rows in the image.
      12 0 DO            \ Eighteen pixels in each row.
         @+2             \ Read data value and increment pointer by two.
         SWAP            \ Bring data value to TOS.
         00FF AND        \ Mask to clear unused bits.
         2 .H ."  "      \ Output  byte.
      LOOP
      CR                 \ Output newline.
   LOOP
   CR                    \ Output newline.
;

: GET-RAW-PIXELS
   R @ DOE_READ_PIXEL_DATA
   R @ RAW-PIXEL-DUMP
;
```