

A Software Defined Edge Finder with CNC Compatible Output, Version 1.7

By **R. G. Sparber**

This design is dedicated to the memory of Dan Benoit.

Protected by Creative Commons.¹

Conclusion



My Software Defined Edge Finder (**SDEF**) has a repeatability better than ± 0.0001 inches. The cost of the electronics is estimated at under \$10. This device is compatible with Centroid's Computer Numerical Control's automatic touchdown feature.

Although developed for my milling machine, it works equally well on a lathe.

Scope

All details are disclosed enabling the reader to reproduce this design. Feel free to contact me if you have any questions or problems.

You can see a video of this device in action at

<https://www.youtube.com/watch?v=WBMOHrkyMIM>

¹ This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Background

An Electronic Edge Finder (EEF) is used to detect when its probe just comes in contact with a reference surface. A zero point can then be precisely set. This operation is commonly performed on a milling machine but can also be done on a lathe.



Some mill based EEFs consist of an internal switch. Upon physical contact with the reference surface, the contact either opens or closes. This state change is fed into CNC code which controls the motion of the mill or lathe.



Other mill based EEFs have an insulated probe. When it makes electrical contact with the reference surface, an electrical circuit is formed and current flows. This flow of current signals the CNC code.

Overview



As with all of my previous Electronic Edge Finder designs², The Software Defined Edge Finder connects to the mill table and to the spindle.

The probe is whatever electrically conductive object is in the spindle. ***No need to electrically insulate the probe from the spindle.*** The reference surface must also be conductive.

Unlike all of my previous designs, this edge finder will respond to touchdown by providing a contact closure compatible with most CNC programs.

Centroid's CNC12 has an automatic touchdown feature that will move to a reference surface at one speed, make contact, back away a given distance, and then move back in at another speed. All three of these parameters are user specified. See Appendix II.

Most of the functionality of this edge finder is in software. The amount of hardware is much less than in previous designs. This software runs on an Atmel ATTiny85.

² <http://rick.sparber.org/ReadMeEEF.pdf>

Limitation



The electrical resistance between the spindle and the rest of the machine must be at least 0.1 ohms.

On my mill, there are times when the resistance is less than 0.1 ohms but I can rotate my spindle a few degrees and it jumps back up³. The SDEF will detect when the resistance is too low and warn the user. The user can watch the warning light and see when the condition has been corrected.

³ The spindle is supported in roller bearings plus the spline engages with the drive pulley. Test show that for my mill, the roller bearings dominate the electrical resistance.

Contents

Conclusion	1
Scope	1
Background	2
Overview	2
Limitation.....	3
Features	5
User's Guide	6
Theory of Operation.....	11
Hardware Overview	14
Schematic	19
Power Switch	28
Bill Of Materials	29
Single-Sided Circuit Board	30
The Software	33
SkinnyPrint.....	39
Acknowledgments.....	40
Appendix I: Test Data.....	41
Appendix II: Automatic Touchdown using Centroid's CNC12.....	42
Appendix III: The code that runs on the ATTiny85	44

Features

- Performs the electronic edge finder function without an insulated probe.
- Works on a mill and on a lathe
- Repeatability is better than ± 0.0001 inches
- Compatible with the automatic touchdown function in CNC.
- Automatic power up
- Automatic power down after one minute with warning to both user and CNC computer.
- Automatic battery test.
- Performs over 2400 touchdowns on a set of 3 AA batteries.
- Built in diagnostics tell the user when and why they must not proceed.
- Estimated cost of the electronics is under \$10.

User's Guide

Overview



The SDEF's enclosure is placed on the mill table. This provides an electrical connection.

The back of the enclosure and the table must be free of debris.



The clip connects to the spindle just above the collet. This clip is designed to carry a large amount of current but the test circuit is passing less than 50 milliamps.



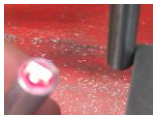
The output of the SDEF feeds into a jack that connects to my CNC hardware. It carries both the Probe Detect and Probe Touchdown signals.



An aluminum cylinder, called the Bridge, is touched between the probe and the conductive reference surface. This is an older picture showing a different clip.



A green LED inside the Bridge lights to acknowledge contact has been made. Then the light goes out to signal that the Bridge can be removed. After about one second, the green LED turns back on to signal that we are ready to start the automatic touchdown sequence.



When the probe makes contact with the reference surface, the LED turns red and the CNC system sees a contact closure. Back away, and the LED goes back to green while the contact opens.

One minute after that first touchdown, the SDEF automatically powers down unless told to stay on⁴. At the end of this minute, the probe flashes alternating red and green light for 10 seconds plus signals the CNC PC of a touchdown to prevent a probe crash. This is a design tradeoff since it does cause a false touchdown.

If there is no touchdown in 1 minute, the circuit powers down.

Sunny Day Procedure

"Sunny Day" means everything is working normally.

1. Plug the connector into the CNC PC and set it up for automatic touchdown.
2. Use alcohol and a wipe to clean the probe, reference surface, back of the SDEF, and area on the table that the SDEF enclosure will contact.
3. Attach the SDEF
 - a. connect the clip to the spindle
 - b. move the probe close⁵ to the reference surface
 - c. place the enclosure on the table
 - d. turn on power
4. Short the gap between the probe and the reference surface with the Bridge.
5. The green LED will turn on.
6. The CNC PC will detect the probe has been connected.
7. Wait for the green LED to turn off.
8. Remove the Bridge and hold it in your hand.
9. Wait for the green LED to turn back on.
10. Start the CNC auto touchdown cycle.
 - a. At touchdown, the red LED comes on.
 - b. When free, the green LED comes on.
 - c. Typically, after automatic touchdown there is no further movement so once the LED turns red, it will stay red until power down.
11. One minute after the first touchdown, the circuit will power down.
12. The CNC PC will detect that the probe has been removed.
13. To prevent accidental power up, turn the power off.

Warning: if the LED turns off, the circuit has powered down so cannot tell the CNC program about touchdown.

⁴ Placing the Bridge on the mill table signals to the SDEF to stay powered up.

⁵ I prefer a distance of about 0.02 inches.

Detailed Procedure including Rainy Day

"Rainy Day" means when things go wrong.

1. Install your probe in the spindle.
2. Use alcohol and a wipe to clean the probe, reference surface, back of the SDEF, and area on the table that the SDEF enclosure will contact.
3. Attach SDEF
 - a. connect the clip to the spindle
 - b. move the probe close to the reference surface
 - c. place the enclosure on the table
 - d. turn on power
4. Quickly short the gap between the probe and the reference surface with the Bridge. If you accidentally touch the Bridge to just one side and not the other for more than the initial 1 second, you will see a flickering green LED. Lift up the Bridge and wait 10 seconds. The device will power down. Then repeat this step.
5. The CNC PC will detect the probe has been connected.
6. Look at the LED in the Bridge
 - a. If you see a red/green flashing pattern, replace all batteries.
 - b. If red, flashes five times, and then turns off, either the Bridge is not shorting the gap between the tool and surface, the enclosure is not solidly on the mill table, or a wire is broken. Correct and start this procedure over again.
 - c. If green and on continuously, proceed.
7. Wait for the green LED to turn off.
8. Remove the Bridge from the probe/reference surface gap and hold it in your hand.
9. Look at the LED in the Bridge
 - a. If green and on continuously, we are ready to proceed.
 - b. If green and flickers, rotate the spindle by hand slowly until it stays on continuously.
 - c. If the flickering persists, touchdown cannot be detected due to excessively low spindle bearing electrical resistance so **do not proceed. Touchdown cannot be detected.**
10. Start the CNC auto touchdown procedure and keep an eye on the LED. **If more than one minute has passed without touchdown, the probe will alternate between red and green light plus tell the CNC PC there has been a touchdown. This is done to prevent a crash but does indicate a false touchdown. Then the circuit will power down.**

11. At touchdown, the LED goes from green to red; a contact closure will tell the program of the touchdown.
12. When the probe is not in contact with the reference surface, the LED is green.
13. LED behavior after automatic touchdown
 - a. At the instance of touchdown, the LED turns red and should stay on until automatic power down.
 - b. One minute after the first touchdown, the circuit will automatically power down unless the Bridge is resting on the machine. This should be plenty of time for auto shutdown to complete.
14. Proceed to next axis and repeat procedure.

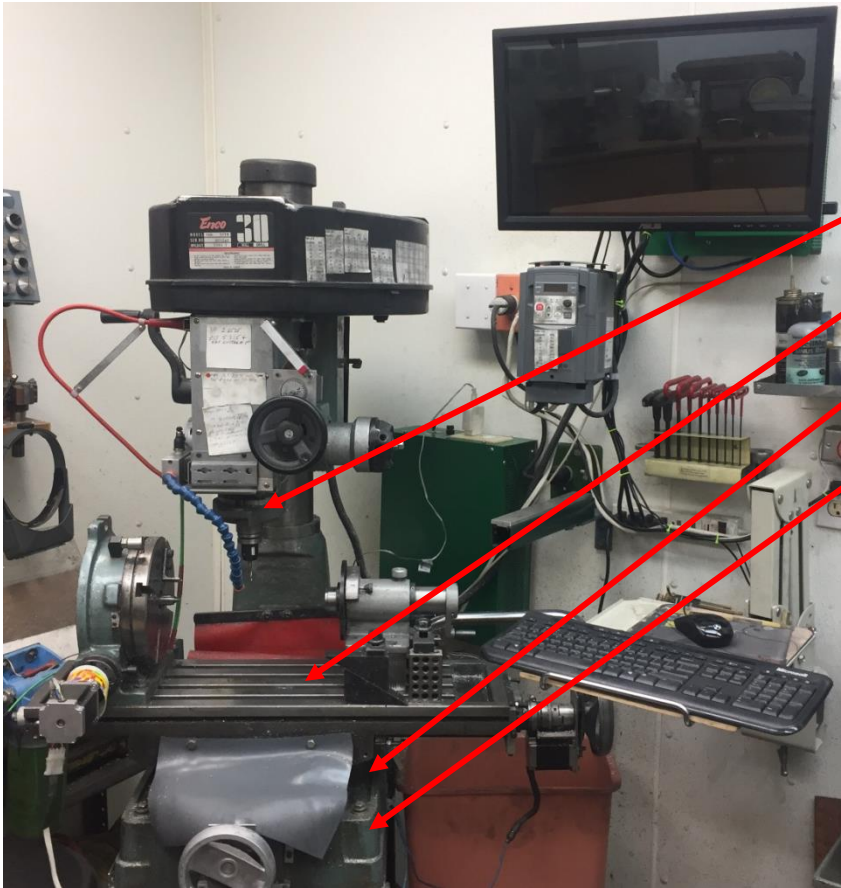
Printable Error Message Reminder Card

Error Messages and User Response

situation	message	meaning	possible reasons	user's action
power up	slow red/green flashing for 10 seconds and then powers off	low battery	The AA batteries are weak.	Replace all 3 batteries.
Bridge just applied	red flashes slowly 5 times and then powers off	probes have poor connections	enclosure not making good contact with mill table, clip not in full contact, probe wire broken.	clean bottom of enclosure and mill table under it; check clip, look for broken probe wire
Bridge just removed	green flickers	Free reading too small	spindle resistance too small	hand rotate spindle a few degrees; if flicker persists, SDEF is not compatible with this machine.

Theory of Operation

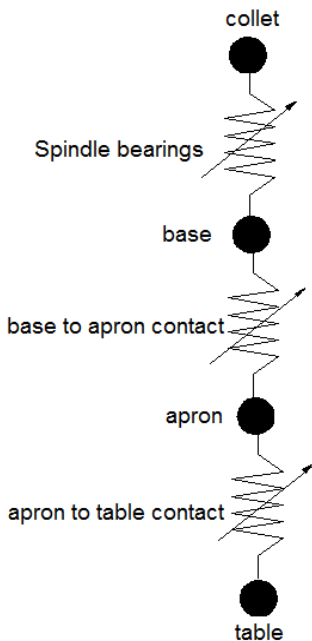
When viewed from a physical standpoint, there is nothing remarkable about my CNC equipped RF-30 mill/drill.



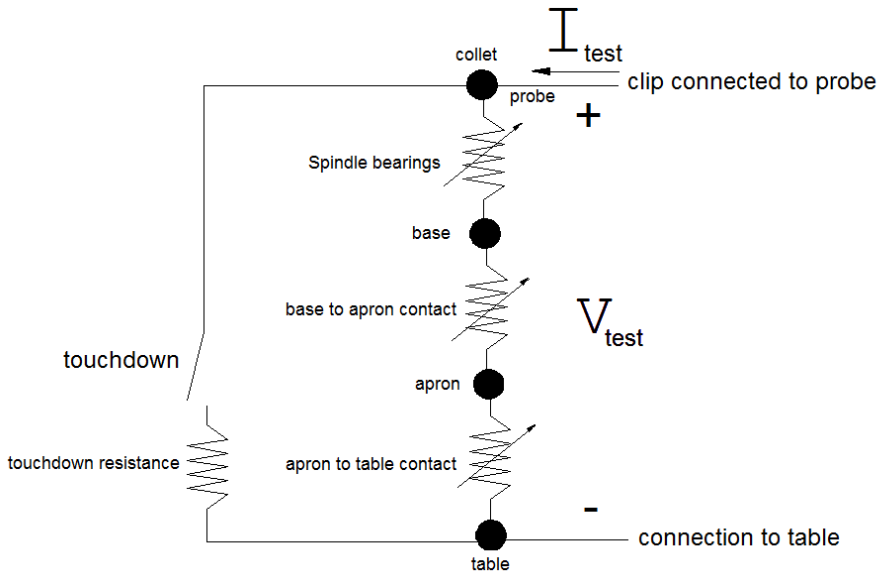
It has a spindle,
table,
apron,
and base.

Viewed from an electrical standpoint, it looks much different. Starting at the collet mounted in the spindle, we have the electrical resistance of the Spindle bearings⁶. These bearings are supported by the head but electrically, that is the same as the base. Then we have the Y axis ways which provide a base to apron resistance. Between the apron and table is the X axis ways so we pick up the apron to table resistance.

All of these resistances vary with physical motion but the spindle bearings dominate.



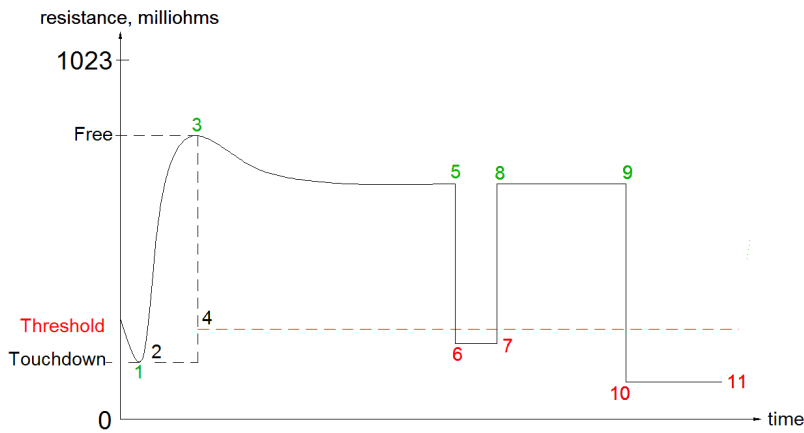
⁶ There is also a connection to the base via the spline but tests have shown this is not significant.



When the clip is attached to the probe and the SDEF's enclosure is placed on the mill table, we are able to measure resistance. Before touchdown, the SDEF sees a relatively large but variable resistance which is the sum of the spindle bearings, base to apron contact, and apron to table

contact resistance. When the probe touches down on the reference surface, we get a resistance much smaller than the sum of the other resistances plus there is little variation.

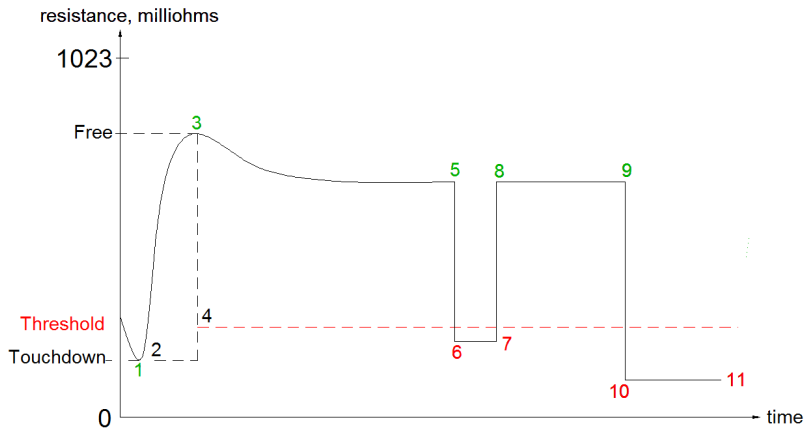
The SDEF applies a test current, I_{test} , and looks at the resulting voltage, V_{test} . When the Bridge is shorting across the gap between the probe and reference surface, the SDEF sees the touchdown resistance and remembers it. When the Bridge is removed, it sees the rest of the resistance. This is enough information to detect touchdown.



Here is a typical "Sunny Day" touchdown sequence⁷. The horizontal axis is time. The vertical axis is the measured electrical resistance between probe and reference surface. Typically this sequence is over in a few seconds.

At **0**, the Bridge is placed across the gap between the probe and the reference surface. This powers up the SDEF. After waiting half a second, it starts to measure the resistance. At **1**, the SDEF stops when it has found the smallest value. This value is called Touchdown. All of this takes about half a second. Then it turns off the green LED (**2**).

⁷ I am not showing the battery check which takes place at initial power up.



The Bridge is removed from the gap and the SDEF starts to look for the maximum resistance. Call the result the Free value. It finds Free at **3**. Then it calculates **Threshold** as a function of Free and Touchdown. Next, the SDEF turns the green LED on to signify it is ready.

As the probe moves relative to the reference surface, the resistance may vary as can be seen between **3** and **5**. The LED remains green. During this time, the CNC program is moving the probe at a relatively high feed rate.

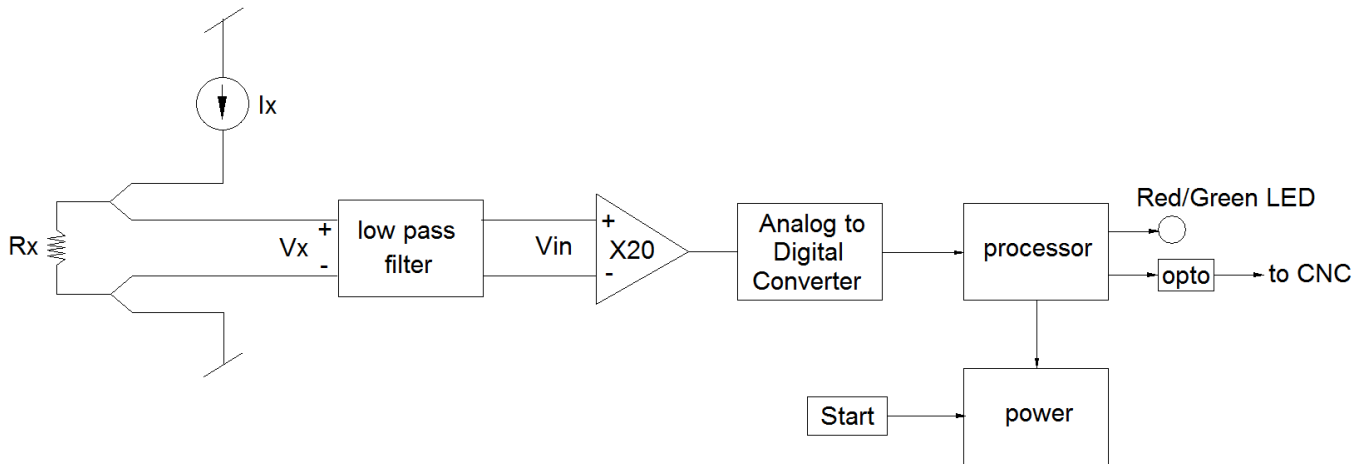
At **6**, the probe has come in contact with the reference surface so the measured resistance abruptly drops to below the **Threshold**. After the SDEF has recorded two consecutive values less than **Threshold**, the LED turns red and the CNC program detects a contact closure. The CNC program responds by quickly backing the probe away. At **7** the probe loses contact with the reference surface.

At **8** the resistance is again above the **Threshold** so the LED goes back to being green and the contact seen by the CNC program becomes an open. After moving the probe away a specified distance, the CNC program again feeds in but now at a relatively slow rate. At **9** the probe is in contact with the reference surface. The resistance drops to below **Threshold** at **10** and the LED turns red. It remains red until **11** when auto power down occurs.

Hardware Overview

The hardware is placed in one of two configurations: edge finder and battery check.

Edge Finder System Block Diagram



When operating as an edge finder, it becomes a glorified ohm meter.

A second opto, not shown, is driven when the circuit is powered up. This opto connects to the Probe Detect lead seen by the CNC PC.

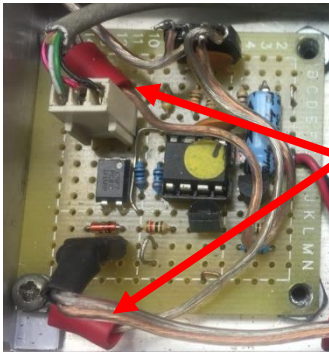
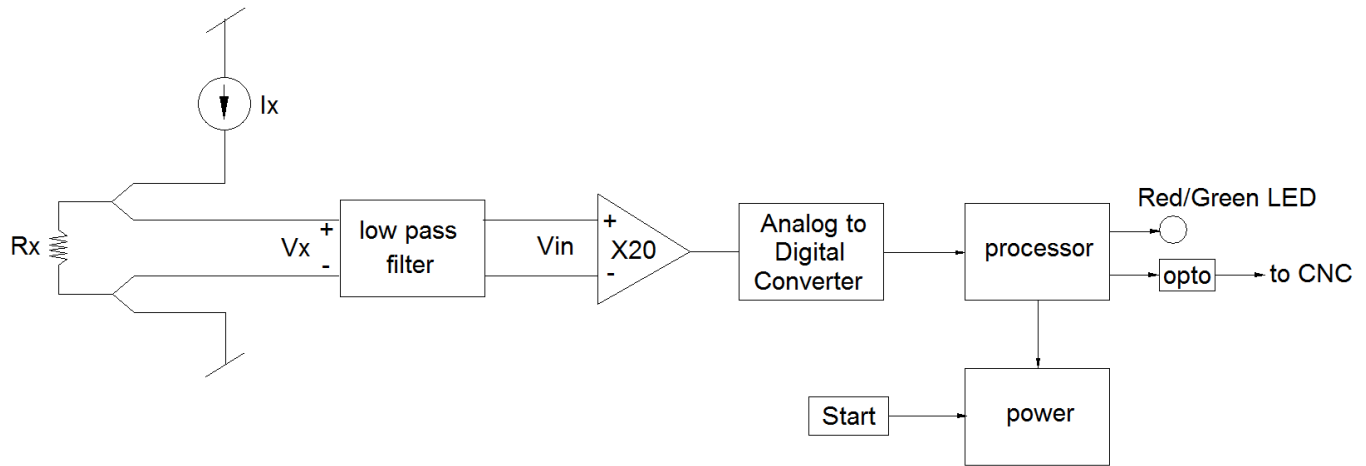
Modified Kelvin Connection

Starting at the left, we have the unknown resistance, R_x . Test current, I_x , passes through it to generate the voltage V_x . A modified Kelvin Connection is used here. In a Kelvin Connection⁸, the test current flows in one pair of wires and the resulting voltage is sensed through a separate pair of wires. In this modified Kelvin Connection, the test current runs separate from the voltage sense leads out to a connector. They are then joined together. This means that the contact resistance of the connector adds directly to R_x which is not good. The advantage is a simpler connection.



The connection to the top end of R_x is via this clip. The I_x wire is connected to the brass arm which is bolted to the clip. The $V_x +$ wire is connected to an insulated bolt. When attached to the spindle, R_x and I_x have separate contact points so do not interact.

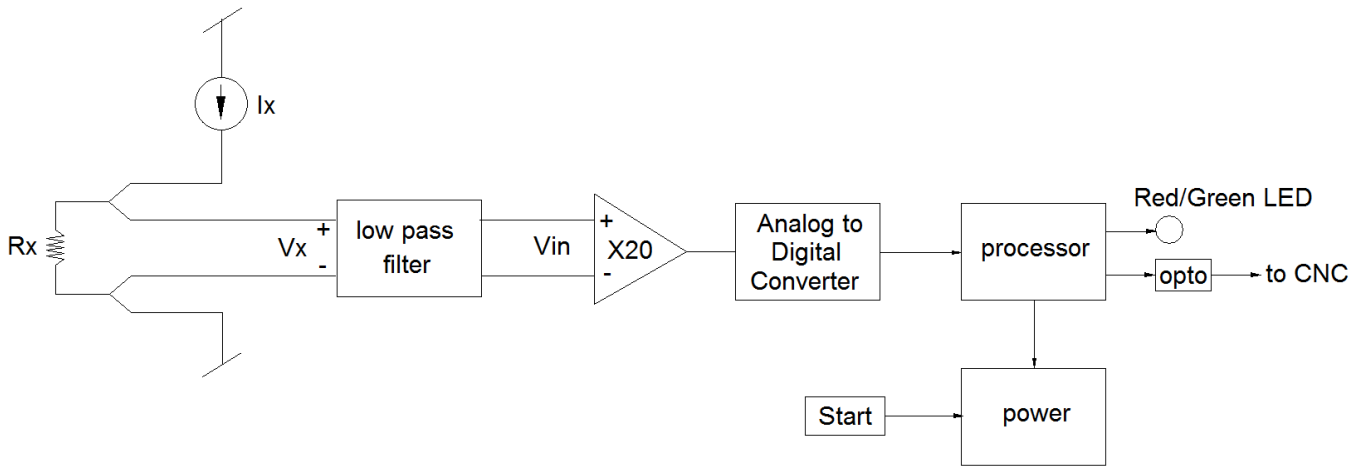
⁸ See https://en.wikipedia.org/wiki/Four-terminal_sensing



The connection to the bottom of R_x is via the bottom surface of the SDEF enclosure.

Note the two red lugs on the left corners of the board.

After all screws going through the back of the enclosure were tightened for the first time, the back was sanded smooth on a belt sander. It is essential that the entire back of the enclosure is in contact with the machine's table in order to provide the lowest possible resistance.



Given our test current, I_x , and our unknown resistance, R_x , we know that

$$V_x = R_x \times I_x \quad (1)$$

and

$$\Delta V_x = \Delta R_x \times I_x \quad (2)$$

Which means that a change in R_x times I_x equals a change in V_x . The SDEF is all about changes in V_x .

Low Pass Filter

V_x feeds into a low pass filter. It removes electrical noise above 500 Hz. This limit was just a guess but seems to work well. If I blocked frequencies a lot lower than this, it would start to delay the detection of Touchdown. If blocked a lot higher than this, it might let in too much noise.

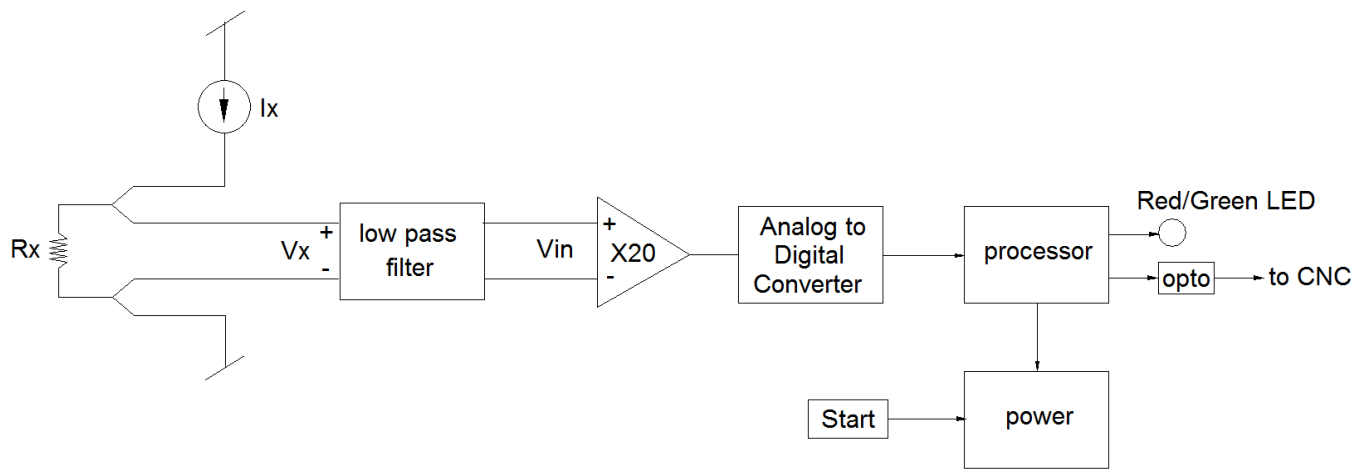
Assuming V_x is DC, all of it appears as V_{in} . The $\times 20$ voltage amplifier boosts V_{in} by 20 and sends it to the Analog to Digital Converter (ADC).

The Analog to Digital Converter

The ADC generates a number that is a proportional to its input. Ideally, when this number is 0, the input voltage is 0. When the number is 1023, the input voltage is 1.1 volts. This means that a single count is

$$\frac{1.1 \text{ volts}}{1023 \text{ steps}} = 1.1 \text{ millivolts per step} \quad (3)$$

There is an offset in the amplifier and ADC but it doesn't matter because the SDEF only looks at changes in input voltage. Any offset present in one reading is canceled by that same offset in the second reading. When the two readings are subtracted, the offsets cancels.



To generate a 1.1 millivolt step at the input of the ADC, we need a change of

$$\frac{1.1 \text{ millivolts}}{20} = 54 \text{ microvolts at } V_{in} \quad (4)$$

Using (2)

$$\Delta V_x = \Delta R_x \times I_x \quad (2)$$

we can figure out how much R_x must change for us to see a single count change at the output of the ADC. I_x is around 50 milliamps.

$$54 \text{ microvolts} = \Delta R_x \times 50 \text{ milliamps}$$

so

$$\Delta R_x = \frac{54 \text{ microvolts}}{50 \text{ milliamps}}$$

$$\Delta R_x \approx 1 \text{ milliohm}$$

Therefore, the hardware is able to detect a change in R_x of 0.001 ohms.

I will get into how the processor uses the output of the ADC in the software section.

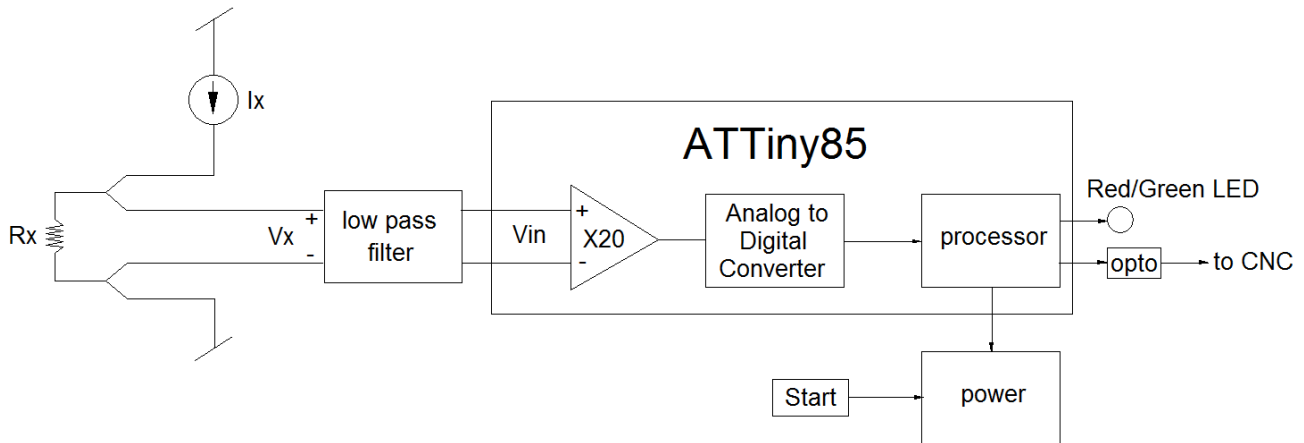
Processor Outputs

The processor has three outputs. One controls the red/green LED. It can be off, red, or green. It also controls the opto isolator. The output of the opto can either conduct current or not. When current is flowing through the opto, the CNC software interprets this as Touchdown.

And finally, the processor controls power. After power is manually turned on, the processor can turn it off unless manually overridden.

The ATTiny85

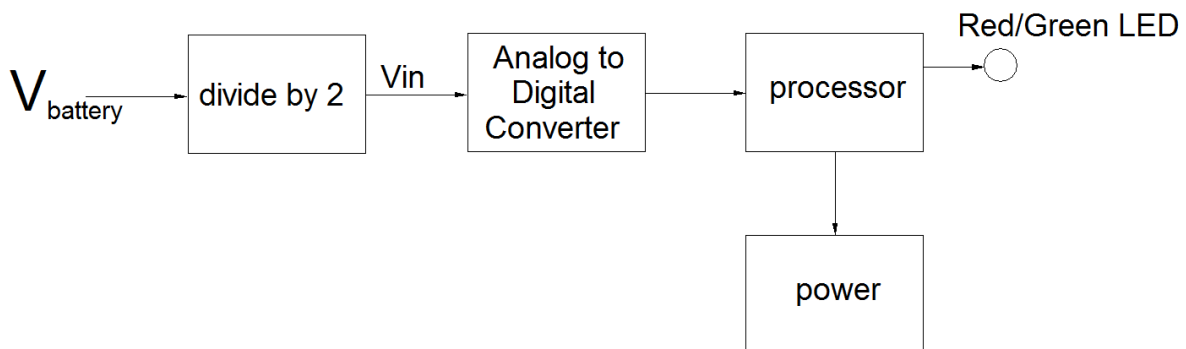
What I find both surprising and delightful is that so much of this functionality is inside the ATTiny85.



One third of all resistors and both of the transistors are related to power control. If the reader was happy with a simple on/off power switch, only 6 resistors, 2 capacitors, one LED, one opto isolator, and the ATTiny85 would be needed.

Battery Check System Block Diagram

The second configuration is for battery check.



The ADC is changed so when it puts out 1023, it is in response to an input voltage of 2.56 volts⁹ and not 1.1 volts. The maximum V_{battery} is 4.5 volts. The divider

⁹ This ignores offset but it turns out the overall accuracy is good enough.

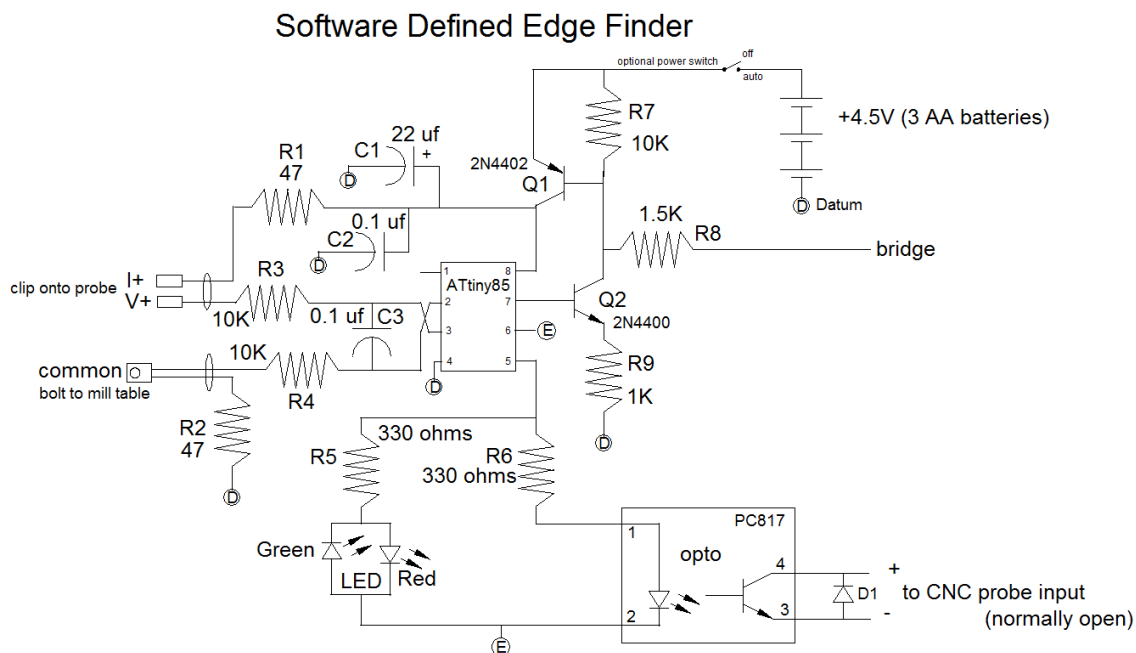
insures that the maximum V_{in} is $\frac{4.5 \text{ volts}}{2} = 2.25$ volts. Therefore, the ADC can handle the full range of battery voltages.

When $V_{battery}$ nominally gets below 3.3 volts, the processor sends a particular pattern of red/green flashes out for about 10 seconds and then powers down. This occurs right at power up so it is hard to miss. Time to replace all 3 AA batteries.

This reconfiguration of the system is all done with software. In fact, it only took 11 lines of code.

Schematic

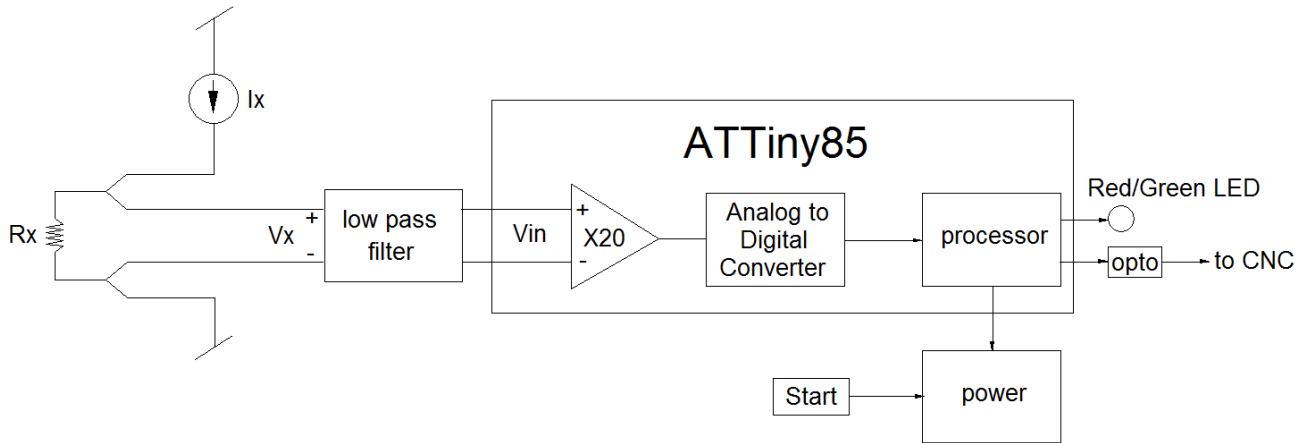
This circuit may look complex but hopefully I can break it down into more understandable parts. Let's start by describing the Edge Finder function and then address battery test.



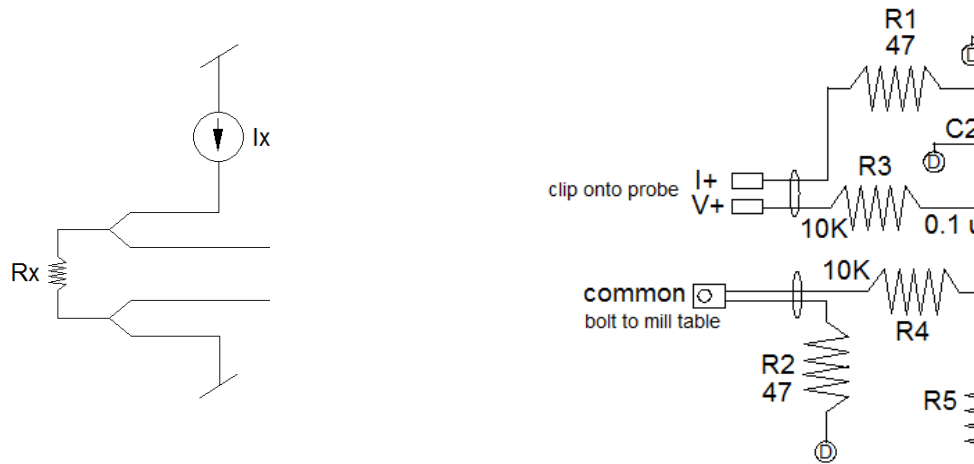
The "D" inside a circle is "datum" and should not be confused with ground which does not exist in this circuit. Datum is a point used by many components. The ATtiny85's "ground" (pin 4) is connected to datum and all of its inputs and outputs are with respect to datum.

A second PC817 opto with a 670 ohm resistor in series with its LED connects between pin 8 and 4 of the ATtiny-85. The opto's output connects to Probe Detect which is monitored by the CNC PC.

The Edge Finder Function



Current Source



The current source, I_x , is formed from R_1 and R_2 . Each is 47 ohms. Approximately 4.5 volts is applied across them. So why is it a current source?

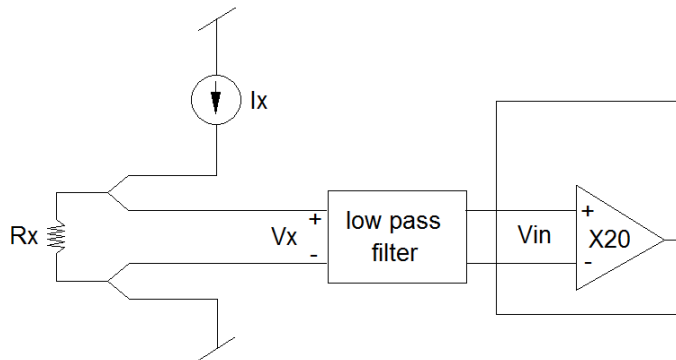
Consider the usable range of R_x . It can be as small as 0 and as large as what the ADC can handle which is $1023 \text{ steps} \times 1.1 \text{ milliohms per step} = 1.1 \text{ ohms}$. Assuming 4.5 volts being applied, the current that flows will vary from a low of

$$\frac{4.5 \text{ volts}}{47 \text{ ohms} + 47 \text{ ohms} + 1.1 \text{ ohms}} = 47.3 \text{ milliamps}$$

to a high of

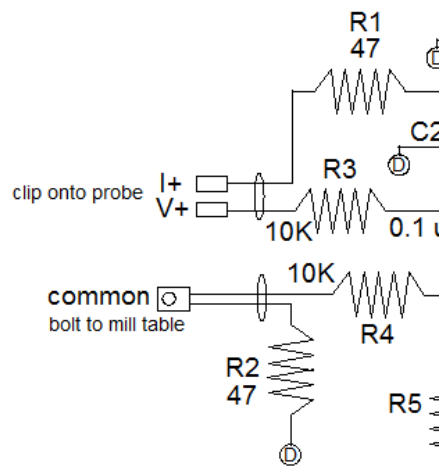
$$\frac{4.5 \text{ volts}}{47 \text{ ohms} + 47 \text{ ohms} + 0 \text{ ohms}} = 47.9 \text{ milliamps}$$

which is a change of only 1%. Typically the change in resistance will be less than 0.1 ohms which translates to a change in current of 0.2%. This is small enough to call constant.

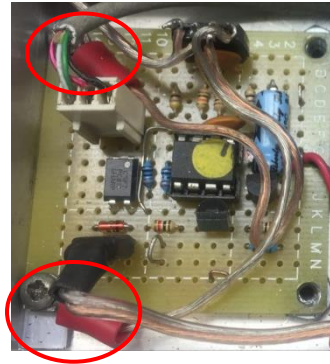


I could have set the current with a single 94 ohm rather than two 47 ohm resistors. But by using two resistors, I was able to center V_x with respect power and "ground" of the ATtiny85. This gave the times 20 voltage amplifier the maximum range to reject noise.

Probe Connections

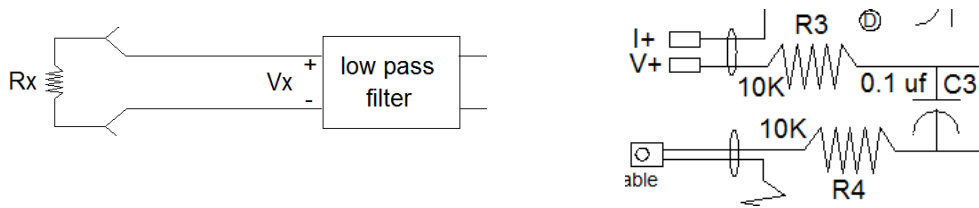


The modified Kelvin connections were shown on page 15. In the schematic they are labeled I+, V+ for the top two and "common" for the bottom two.



Remember that the "common" connection is between the bottom of the SDEF enclosure and the mill table. R2 connects to the enclosure via one lug and R4 connects via a different lug.

Low Pass Filer

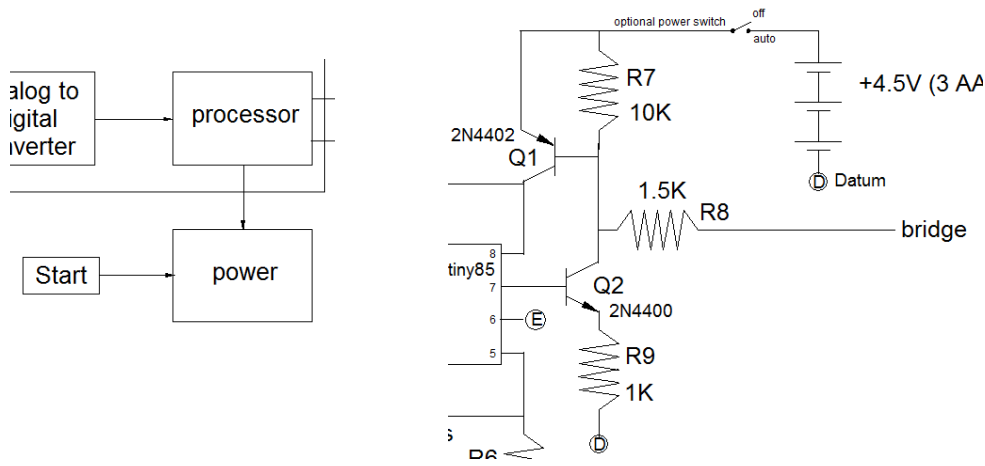


The low pass filter is formed by R3, R4, and C3. It has a corner frequency of $\frac{1}{2\pi(10K+10K)\times 0.1\mu F} = 80 \text{ Hz}$. These values are really a guess because they have only been tested on one machine. I measured the noise signal from the stepper motors and found a period of $5 \mu s$ which means a fundamental frequency of $\frac{1}{5 \mu s} = 200 \text{ KHz}$. If necessary, the capacitor can be made larger in order to start attenuating signals at a lower frequency.

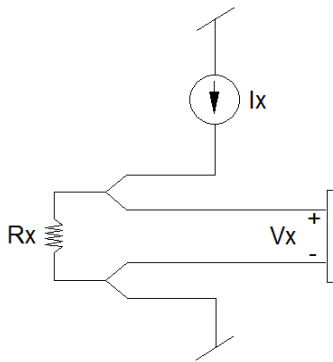
I empirically found that having the circuit in a sealed metal box is also essential to minimizing noise susceptibility.

Power

Power control is the largest sub circuit, formed by Q1, Q2, R7, R8, and R9.



In order to Start supplying power, current must flow in R8. This is accomplished by connecting the Bridge to either terminal of R_x . That is done when the Bridge is placed between probe and reference surface in order to simulate a touchdown.

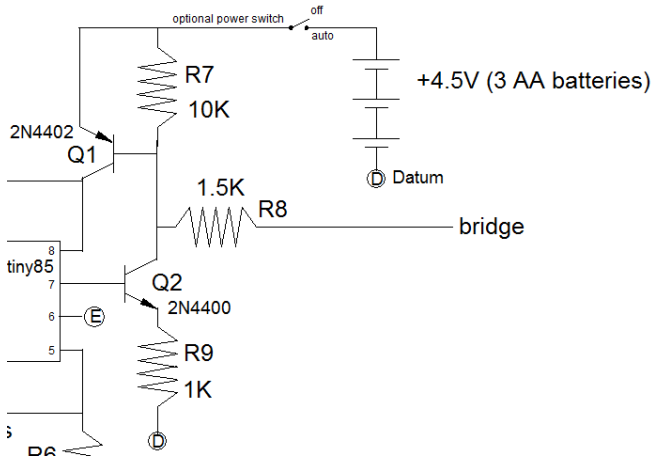


This current flows through R7 until Q1 fully¹⁰ turns on. Then the ATTiny85 gets power. One of the first things the software does is put a logic 1 on pin 7. This applies about 4 volts to the base of Q2. That causes the emitter of Q2 to be at about 3.4 volts which means 3.4 volts is across R9. A current flows in R9 equal to $\frac{3.4 \text{ volts}}{1K} = 3.4 \text{ milliamps}$.

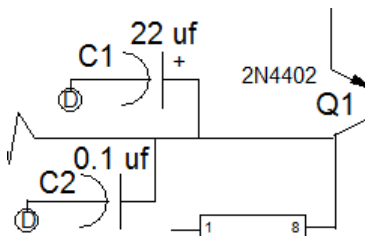
R9's current is pulled out of the emitter of Q2 and almost all of it is pulled into its collector. This is more than enough current to keep Q1 fully turned on.

¹⁰ The technical term is "saturation". In this state, the voltage between emitter and base is around 0.15 volts.

Assuming the bridge is placed on an insulated surface, current stops flowing in R8. When the software causes pin 7 to go low, Q2 will turn off which will cause Q1 to turn off. That will kill power to the ATTiny85. In this state, no current will flow from the batteries.

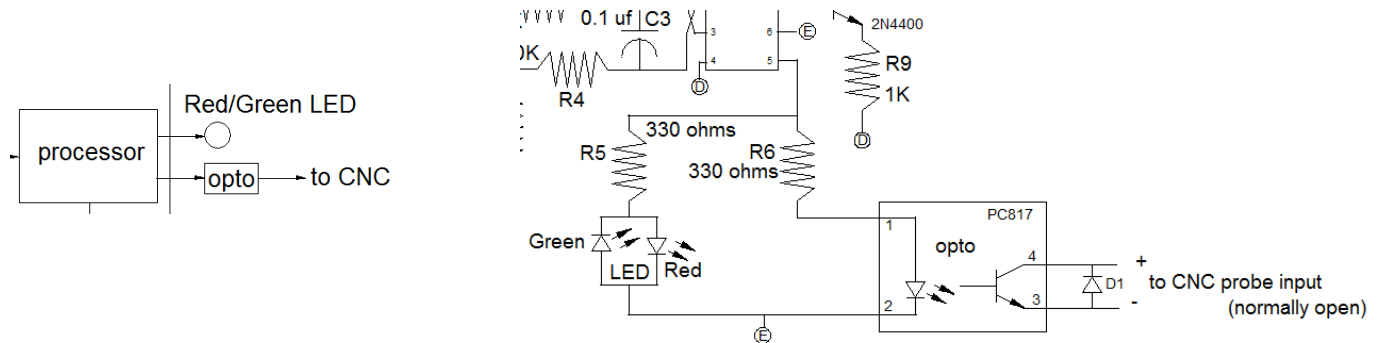


Also part of power are capacitors C1 and C2. C1 smoothes out low frequency noise while C2 deals with high frequencies. The ATTiny85 generates high frequency noise. Mechanical vibration can cause probe connections to move generating low frequency electrical noise.



Status Indicators

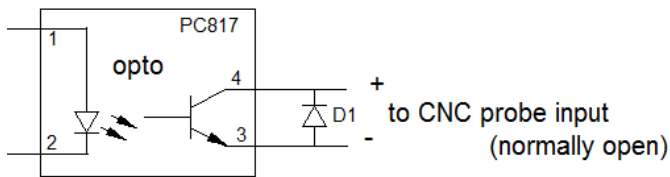
Note in the schematic that pin 6 of the ATTiny85 goes to a circle with an "E" in it. A second "E" is shown at the red/green LED and pin 2 of the opto. I drew it this way to untangle the schematic.



The processor controls the red/green LED via pins 5 and 6. If pin 5 is high and pin 6 is low, the red LED turns on. If pin 5 is low and pin 6 is high, the green LED turns on. Furthermore, when the red LED is on, the opto isolator turns on.

There are two off states. If pins 5 and 6 are both high or both low, neither LED is on and the opto is off.

SDEF to CNC Interface



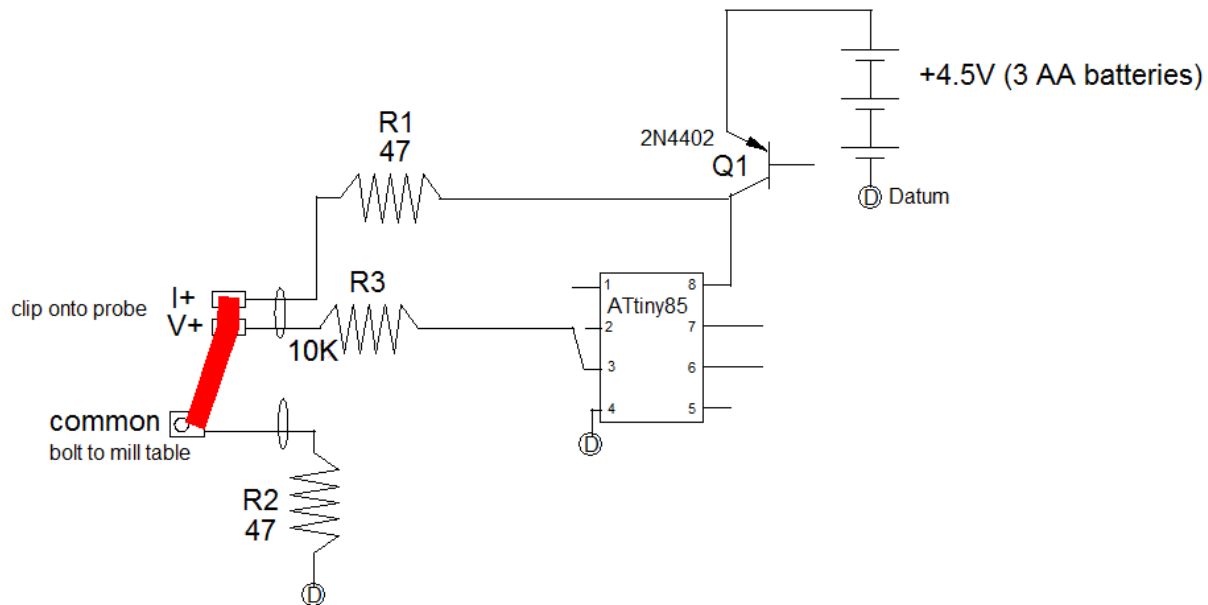
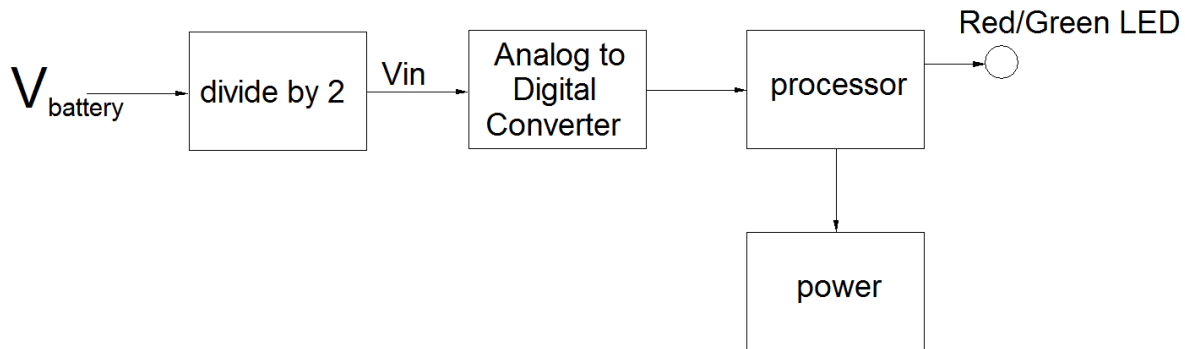
The output of the opto isolator has a hidden feature. If wired correctly, the CNC PC will only see current flowing when the probe has touched down on the reference surface. But if the wires

are flipped, the CNC PC cannot see touchdown because the opto is upside down. To prevent this "latent" failure, I have added diode D1. If the wires are flipped, the diode will conduct right away and indicate a false touchdown. This diode also protects the output of the opto from excessive reverse voltage due to miswiring.

A second opto is driven by the power supplied to the ATTiny-85 and feeds the Probe Detect signal seen by the CNC PC. When power turns off, the CNC PC will see the probe has been disconnected.

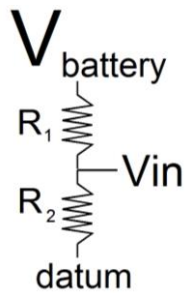
The Battery Test Function

The battery test configuration is more about looking at the existing circuit in a different way plus changing the internal state of the ATTiny85.



When the Bridge (red line) is first placed across the probe to reference surface gap, the ATTiny85 is configured for the battery test. It looks at the voltage on pin 3 relative to its datum (pin 4). The Bridge has zero ohms compared to R1 plus R2. The power control transistor, Q1, has near zero volts emitter to collector. Therefore, the right end of R1 is essentially connected to the positive side of the battery string. The bottom end of R2 is connected to the negative side of the battery string which is datum (D) .

Redrawing this circuit gives us



The result is that the ATtiny85's ADC is looking at half of the battery voltage. It can then detect when the AA batteries are at end of life.

Power Switch

The SDEF software does turn off its own power but there are times when it is good to manually turn it off.

If the user makes a mistake during set up, momentarily toggling the power switch to off resets the circuit.

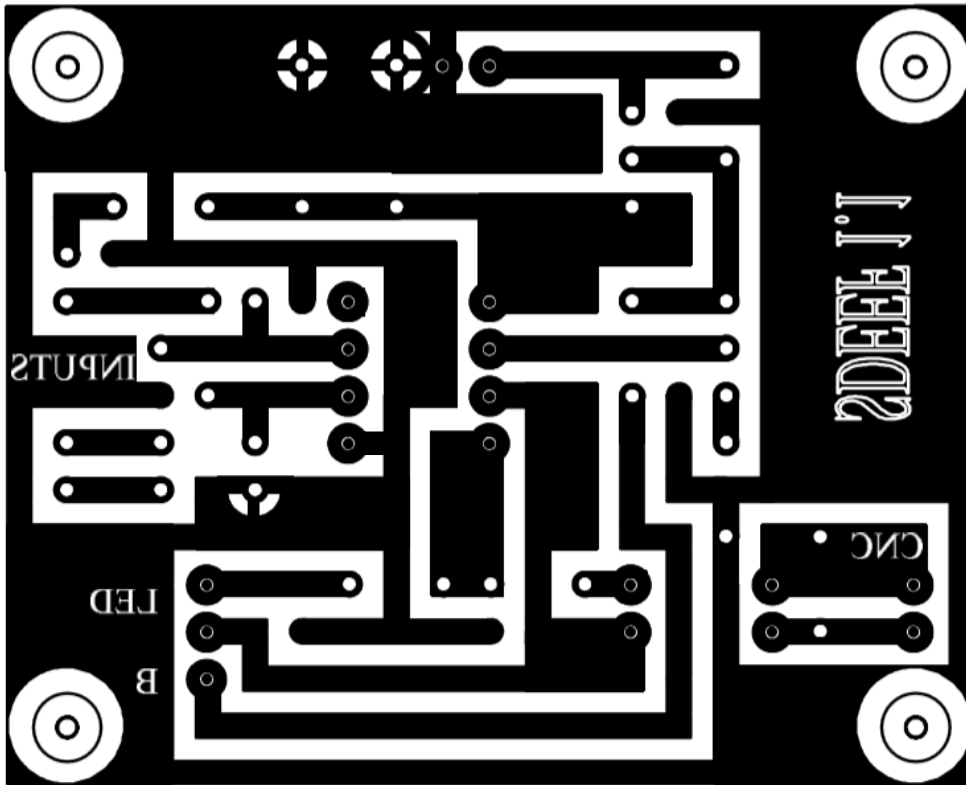
To prevent accidental power up during storage, place the switch in the off position.

Bill Of Materials

Names(s)	Quantity	description
R1, R2	2	47 ohms 5%, 0.1 watts
R3, R4, R7	2	10K ohms 5%, 0.1 watts
R5, R6	2	330 ohms 5%, 0.1 watts
R8	1	1.5K ohms, 5%, 0.1 watts
R9	1	1K ohms, 5%, 0.1 watts
R10	1	670 ohms, 5%, 0.1 watts (not in schematic; see text)
C1	1	22 μ F, 16V electrolytic
C2, C3	2	0.1 μ F, 10V or higher ceramic capacitor
opto	2	PC817 (one is not in schematic; see text)
Q1	1	Any general purpose PNP transistor like the 2N4402
Q2	1	Any general purpose NPN transistor like the 2N4400
D1	1	Any general purpose diode able to sustain 30V when off
Red/green LED	2	Any general purpose red/green LED
ATTiny85	1	Arduino compatible processor.
Perforated board	1	1-3/4" x 1-3/4"
None	1	8 pin DIP socket
None	3	Sockets for cables
Bridge	1	Machined aluminum rod.
Clip	1	Large copper plated steel clip (see page 6)
None	2	Crimp on lugs (see page 15)
None	1	Three AA battery holder
None	3	AA batteries
Enclosure	1	Metal box to hold circuit and batteries
power switch	1	Optional. Double Pole Double Through; low power
Green LED	1	Optional

If anyone orders these parts on-line, please send me the part numbers and price and I will include them along with placing your name in the Acknowledgements.

Single-Sided Circuit Board



This artwork is on a 0.1 inch grid. You may have to play with your print settings to get the correct scale. The design minimizes the amount of copper removed which saves on etchant. Having as much ground as possible also reduces noise.

The square feature indicates pin 1 of the ATTiny85 and of the opto.

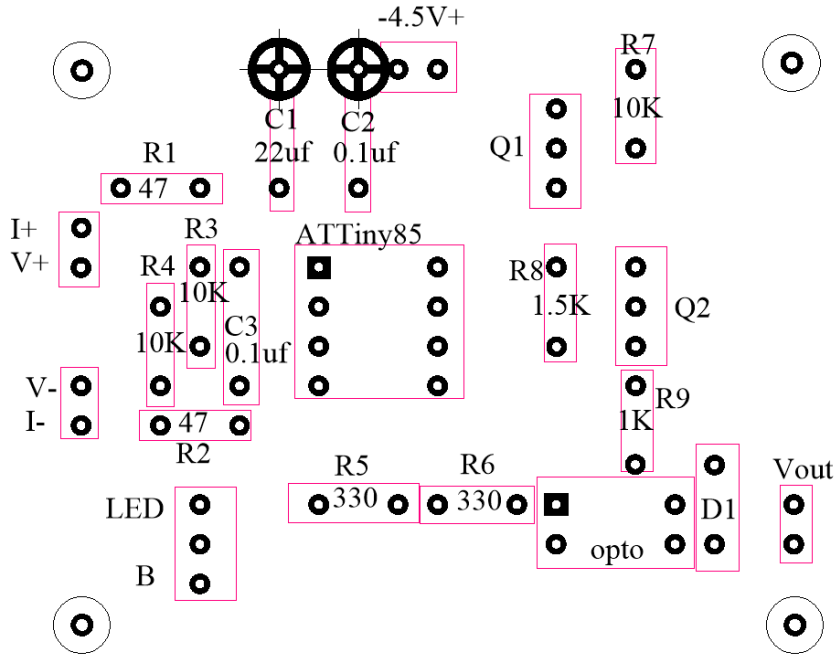
When making your contact print, be sure the text is right reading.

If you look closely, you will see a circle with no mark in the center that is the smaller size. Then there is a larger circle with a dot in the center. The smaller holes are drilled using a 0.0295 inch (0.75 mm) diameter drill. Use a 0.0350 inch (0.9 mm) diameter drill for the circles with the dot in the center.

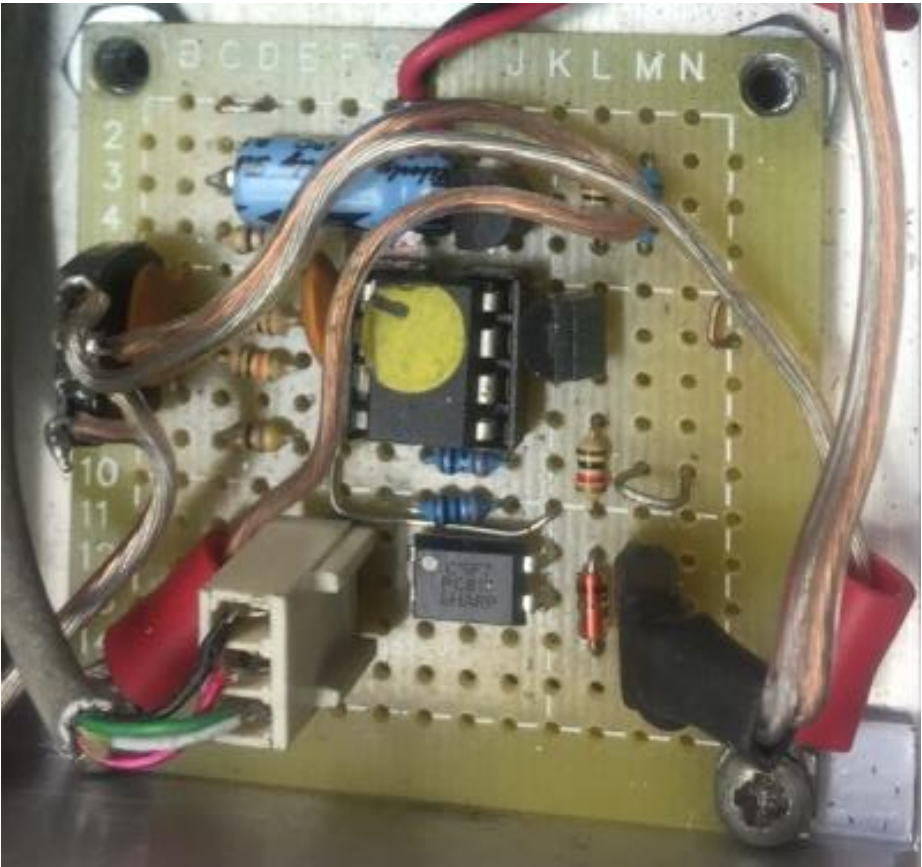
The larger holes must be accurately drilled or the integrated circuits and sockets will not fit.

The very large circles in the corners are for mounting screws.

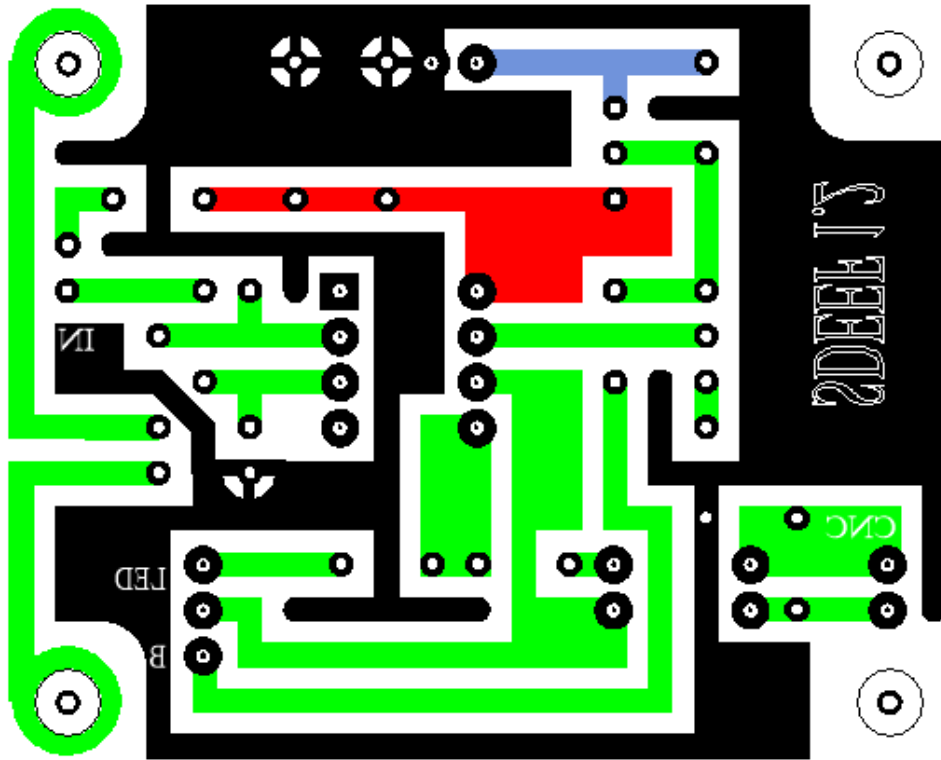
This artwork does not include the second opto or its resistor.



This is a top view of the board showing component placement. Refer to the schematic and part spec sheet for the orientation of Q1, Q2, and D1.



Although I have tried my best to create the artwork without any errors, I have not yet used it. However, the hand-wired prototype was mostly followed to create the component and trace placement.

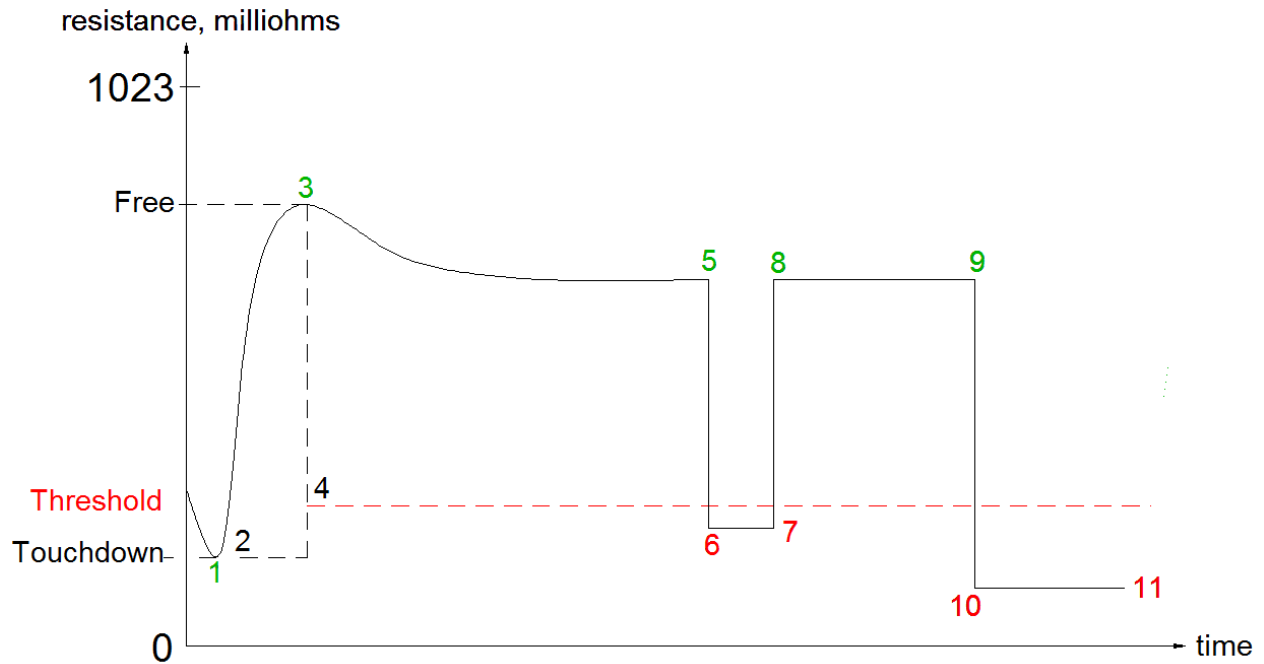


This version of the board connects inputs directly to mounting screw holes. This is good because it eliminates two wires and two lugs. It is bad if you want the flexibility of connecting these inputs elsewhere.

The Software

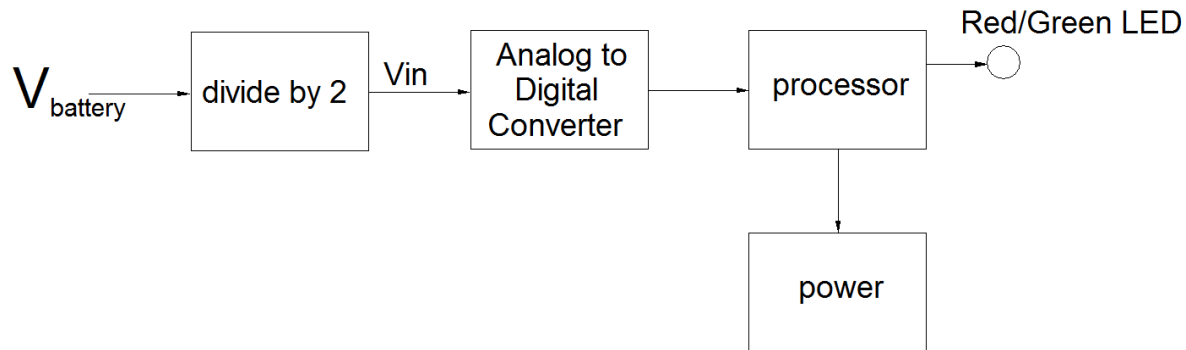
The entire program can be found in Appendix III.

Overview of Behavior



First the battery voltage is checked. Then Touchdown is recorded. After the user removes the Bridge, Free is recorded. Then Threshold is calculated. Live readings of the resistance are taken and compared to Threshold. When the resistance falls below the Threshold and stays there for two consecutive readings, touchdown is declared.

Battery Check



Some may find this a little scary, but the code directly writes to hardware registers. These registers are fully described in the ATTiny85's 234 page spec sheet. One such register is called ADMUX.

17.13.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
0x07	REFS1 REFS0 ADLAR REFS2 MUX3 MUX2 MUX1 MUX0								ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Here is where I select the ADC's reference voltage (the REFS bits). For battery check, I select 2.56 volts. I also select a voltage gain of 1 with an input referenced to the ground of the device (the MUX bits).

If the battery voltage is above 3.3 volts, the ATTiny85 is reconfigured to be an Edge Finder.

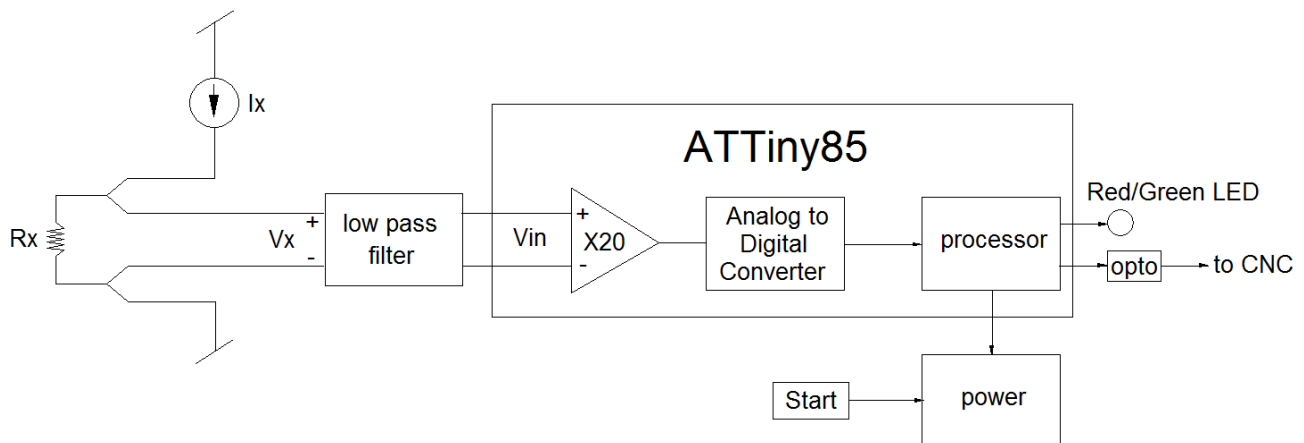
Normal touchdown cycles take less than one minutes and about 50 mA is pulled from the batteries. That is

$$(50 \text{ mA} \times 1 \text{ minutes}) \times \frac{1 \text{ hour}}{60 \text{ minutes}} = 0.83 \text{ mAHours} \quad (5)$$

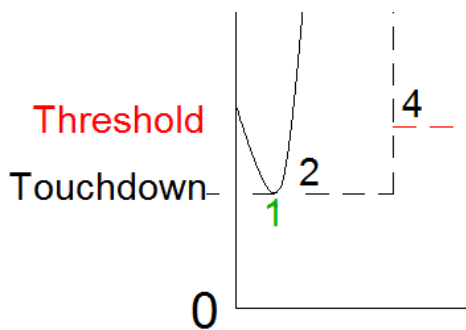
The batteries have a capacity of around 2000 mAHours. We can therefore estimate the number of touchdown cycles we get from a set of fresh batteries:

$$\text{number of cycles} \approx 2000 \text{ mAHours} \times \frac{1 \text{ cycle}}{0.83 \text{ mA Hours}} = 2400 \text{ cycles} \quad (6)$$

Edge Finder



The ADMUX register is set for an ADC reference voltage of 1.1 volts. The voltage gain is changed to times 20 and the input is set to differential.



The user is given one second to fully seat the Bridge against the probe and reference surface. Then we start to measure the resistance. We stop when either the resistance stops falling or when 2000 readings have been taken¹¹.

Each reading is the average of 10 ADC cycles¹². This reduces input noise.

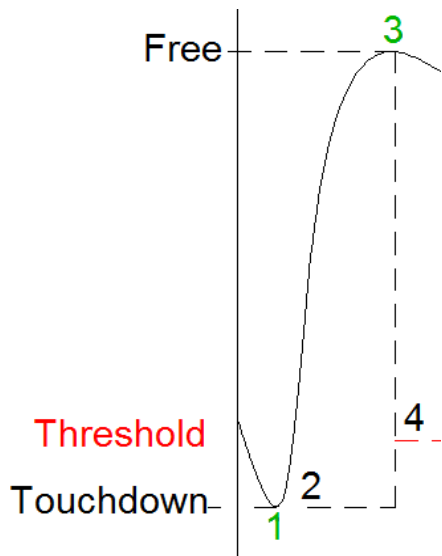
The resulting value is called Touchdown. If it is too close to the maximum value readable by the ADC, a flashing LED warning is generated followed by power down. There would not be enough swing to detect Free.

The ADC's maximum output value is 1023. The maximum permitted Touchdown value¹³ is 800. This translates to a V_x of nominally 40 mV or an R_x 0.8 ohms. Typically it is less than 0.005 ohms.

¹¹ In the code, the variable is called MaxNumberOfReadings.

¹² NumberOfCycles

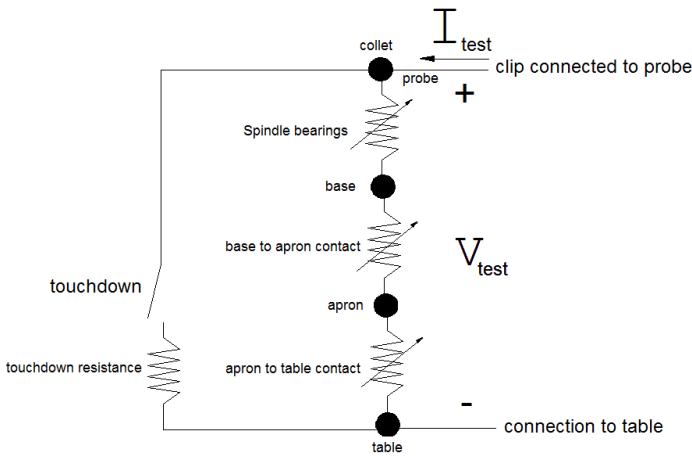
¹³ MaximumTouchdown



Next we signal the user to remove the bridge. After waiting 0.5 seconds, the SDEF starts to measure voltage again. It is looking for the maximum Free value (3). If the reading stops rising or 200 readings¹⁴ have been taken, we record Free.

Empirically, I have found that for reliable operation we need a change between Free and Touchdown of at least 5 millivolts. This translates to a swing of at least 0.1 ohms or $\frac{5 \text{ mV}}{0.05 \text{ mV per count}} = 100 \text{ counts}$.

If Free is less than 0.1 ohms away from Touchdown, there isn't enough noise margin to reliably detect touchdown. In this case, the green LED is made to flicker like a flame. This tells the user to slightly rotate the spindle which should raise the spindle bearing's electrical resistance. If successful, Free will increase enough that when touchdown occurs, the resulting drop in measured resistance will be more than 0.1 ohms.



Free equals the sum of the Spindle bearing, base to apron contact, and apron to table contact resistance.

Touchdown equals the above resistance in parallel with the touchdown resistance.

Threshold is derived from Free and Touchdown. It is set at 5% of the swing from Touchdown to Free. This amount is added to Touchdown and is called the Threshold. In other words,

$$\text{Threshold} = \text{Touchdown} + 0.05(\text{Free} - \text{Touchdown})$$

OR

$$\text{Threshold} = \frac{\text{Free} + (19 \times \text{Touchdown})}{20} \quad (7)$$

¹⁴ MaxReadings

For example, if Touchdown is 1.5 mV and there is no offset, the ADC will output $\frac{1.5 \text{ mV}}{0.05 \text{ mV per count}} = 30 \text{ counts}$. Given the minimum swing of 100 counts, Free would be 130. Plug these values into (7)

$$\text{Threshold} = \frac{\text{Free} + (19 \times \text{Touchdown})}{20} \quad (7)$$

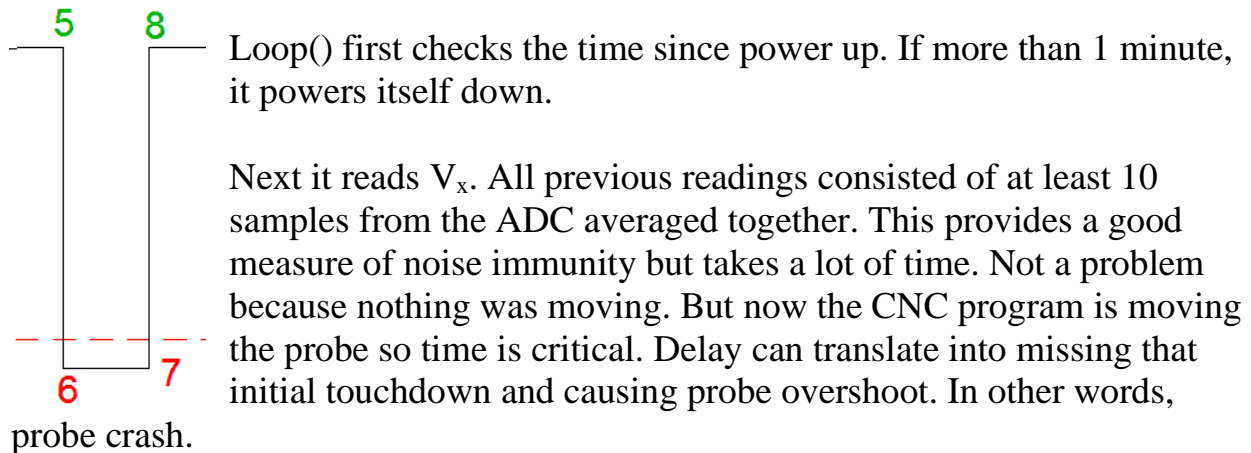
$$\text{Threshold} = \frac{130 + (19 \times 30)}{20}$$

$$\text{Threshold} = 35$$

Therefore, an input voltage of less than $0.05 \text{ mV per count} \times 35 \text{ counts} = 1.75 \text{ mV}$ will be called Touchdown. This translates to $\frac{1.75 \text{ mV}}{50 \text{ mA}} = 35 \text{ milliohms}$.

As Free increases, we can raise the Threshold to provide more noise margin.

This is the end of setup(). We now enter loop().



Therefore, a different strategy is used. Single ADC samples are taken and evaluated. When two consecutive ADC samples are less than Threshold, the SDEF calls Touchdown. Empirically, this strategy has provided accept noise rejection and no measurable overshoot when compared to manually moving the probe a tenth at a time.

By toggling pin 7 low before each ADC sample, I was able to see with my oscilloscope that it takes 300 microseconds to detect touchdown. Given a feedrate of 0.05 IPM, we can calculate how much the table moves while the SDEF decides it has touchdown:

$$\frac{0.05 \text{ inches}}{\text{minute}} \times \frac{1 \text{ minute}}{1,000,000 \text{ us}} \times 300 \text{ us} = 15 \text{ micro inches} = 0.000 015 \text{ inches}$$

Therefore, the table moves 0.000 015 inches during the 300 microseconds taken to determine we have touchdown. This is an acceptable error because it is 7 times smaller than what the CNC software will display.

The SDEF then signals the state change by turning off the green LED and turning on both the red LED and the opto (6). It then waits 200 milliseconds to be sure the CNC software sees the state.

If this is the first touchdown, it sets a flag which will later start a power down timer. Less than ten seconds after the first touchdown, the CNC auto touchdown code will be done so it is safe to power down the SDEF.

If the reading is greater than Threshold, the green LED stays on, the red LED stays off, and the opto stays off.

SkinnyPrint

Mixed into the Constants, Variables, and Subroutines sections is code used by a diagnostic tool called `SkinnyPrinter()` and its associated tool `SkinnyPrintByte()`.

`SkinnyPrintByte()` takes a byte and puts it in a stack that is unloaded by `SkinnyPrinter()`. As each byte comes off of the stack, it is transmitted via a single pin to a receiver which translates the stream back into a byte that is displayed on a serial monitor.

When this functionality is needed, Logical pin 2 (physical pin 7), is taken away from the automatic power control functionality and given to `SkinnyPrint()`. Manual power control is then used to keep the SDEF alive. For more on `SkinnyPrint`, see <http://rick.sparber.org/SPrint.pdf>

Acknowledgments

Thanks to Mark Cason for planting the idea of a Software Defined Edge Finder in my head so many years ago. Thanks to Darrell Crooks for finding my error in the corner frequency of the low pass filter. Thanks to G.L. Carlson, PhD, for finding a missing connection on the circuit board.

A posthumous thanks to Dan Benoit who used one of my previous design and had asked for an Electronic Edge Finder compatible with CNC automatic touchdown. I'm sorry he couldn't try this one out. I think it would have brought one of those famous smiles to his face.

I welcome your comments and questions.

If you wish to be contacted each time I publish an article, email me with just "Article Alias" in the subject line.

Rick Sparber
Rgsparber.ha@gmail.com
Rick.Sparber.org

Appendix I: Test Data

machine position	variation, thou
5.7048	0.1
5.7048	0.1
5.7047	0.0
5.7048	0.1
5.7048	0.1
5.7047	0.0
5.7048	0.1
5.7047	0.0
5.7048	0.1
5.7047	0.0
5.7047	0.0
5.7047	0.0
5.7048	0.1
5.7048	0.1
5.7047	0.0
5.7047	0.0
5.7047	0.0
5.7047	0.0
5.7048	0.1
5.7047	0.0
5.7047	0.0

These 20 data points were taken on 3/24/2018. The voltage between probe and reference surface was above 50 mV during the entire run. This maximizes noise margin.

The average machine position is 5.70474 inches so the worst case measured variation is

$$+ 0.00006 \text{ inches, } - 0.00004 \text{ inches}$$

The CNC display shows to the nearest 0.0001 inches.

machine position	variation, thou
5.70474	
5.7102	0.0
5.7103	0.0
flicker	n/a
5.7103	0.0
5.7102	0.0
5.7103	0.0
5.7104	0.1
5.7102	0.0
5.7102	0.0
5.7102	0.0
5.7102	0.0
5.7102	0.0
5.71025	

In this run the voltage between probe and reference surface was 7.7 mV during the entire run. This minimizes the noise margin. The third sample was declared invalid by the SDEF due to out of range low input voltage. Worst case measured variation was

$$+ 0.00015 \text{ inches, } - 0.00005 \text{ inches.}$$

Throwing about the single outlier (5.7104), the variation was

$$+ 0.00005 \text{ inches, } - 0.00005 \text{ inches.}$$

Appendix II: Automatic Touchdown using Centroid's CNC12

To get to the parameters page, press Setup (F1),

Setup	Load	MDI	Run	CAM	Edit	Utility	Graph	Digitiz	Shut Down
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10

Config (F3),

Part	Tool	Config	Feed	Smoothing Setup
F1	F2	F3	F4	F8

Params (F3)

Contrl	Mach.	Parms	PID	Test
F1	F2	F3	F4	F5

Machine Parameters P0 - P99									
000	-5.0000	020	72.0000	040	0.0001	060	0.0000	080	0.0000
001	0.0000	021	7.9778	041	0.0010	061	5.0000	081	-1.0000
002	0.0000	022	7.9778	042	54.0000	062	90.0000	082	0.0000
003	0.0000	023	7.9778	043	0.0000	063	128.0000	083	0.0500
004	8.0000	024	7.9778	044	0.0000	064	0.0000	084	3.0000
005	0.0000	025	3.3333	045	0.0000	065	1.0000	085	0.0000
006	0.0000	026	3.3333	046	0.0000	066	1.0000	086	0.0000
007	0.0000	027	3.3333	047	0.0000	067	1.0000	087	24.0000
008	2.0000	028	3.3333	048	0.1000	068	0.0000	088	24.0000
009	0.0000	029	212.0000	049	0.0000	069	1.0000	089	24.0000
010	0.0000	030	260.0000	050	0.0000	070	0.0010	090	24.0000
011	50006.0000	031	0.0000	051	0.0000	071	0.0000	091	0.0000
012	10.0000	032	0.0000	052	0.0000	072	0.0000	092	0.0000
013	0.0050	033	1.0000	053	0.0000	073	0.0500	093	0.0000
014	0.5000	034	8000.0000	054	0.0000	074	4.0000	094	1.0000
015	0.1000	035	0.0000	055	0.0000	075	0.0000	095	2.0000
016	10.0000	036	0.0000	056	4.0000	076	0.0000	096	2.0000
017	0.0000	037	10.0000	057	0.0000	077	0.0000	097	2.0000
018	0.0000	038	0.0000	058	0.0000	078	0.0000	098	2.0000
019	2.0000	039	200.0000	059	0.0000	079	0.0000	099	1.0000

Probing Recovery Distance

Prev. Table	Next Table	Save
F7	F8	F10

Which should take you to the first Machine Parameters page: Click on 013 to access Probing Recovery Distance. After the initial automatic touchdown, the probe is backed off by this distance.

It shows 0.0050 but I later changed it to 0.0040 inches.

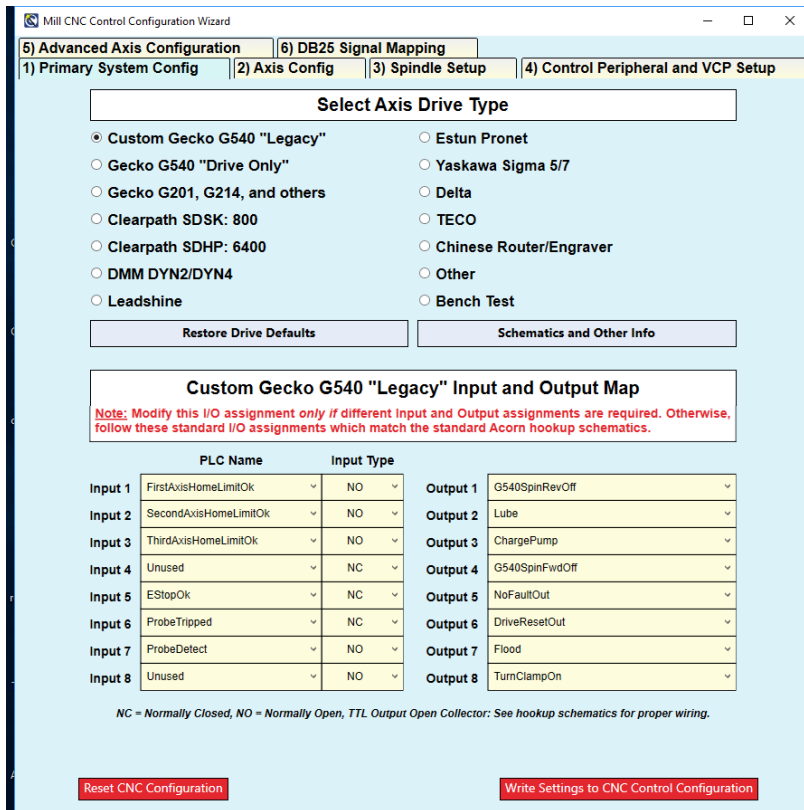
Press Enter after a change is made.

014	0.5000
015	0.1000
016	10.0000
017	0.0000
018	0.0000
019	2.0000
Fast Probing Rate	

Parameter 014 is the Fast Probing Rate. It is 0.5000 Inches Per Minute here but I later changed it to 0.2000 IPM. Any overshoot will bend the probe which is not good. Moving to it slowly minimizes the bending. On the other hand, it does take longer to get to the reference surface for the first time.

015	0.1000
016	10.0000
017	0.0000
018	0.0000
019	2.0000
Slow Probing Rate	

Parameter 015 is the Slow Probing Rate. It is 0.1000 IPM here but I changed it to 0.0500 later. These changes reduced the variation in touchdown value by about a tenth. ***Be sure to do a Save (F10) if any values are changed.***



It is also necessary to define the SDEF to CNCPC interface. This is done with the CNC12 Wizard.

The SDEF's output is and open circuit before touchdown and closed at touchdown. If the edge finder was not connected, it would look, to the CNC software, as if we were in the pre-touchdown state. To guard against this false indication, a second signal is used, ProbeDetect. It tells the software that the probe has been connected.

The SDEF will only make the ProbeDetect signal true if the jack has been connected to the CNC hardware and if the SDEF has power.

We must define a ProbeDetect input. The Input Type must be set to NO (Normally Open)¹⁵. Otherwise, CNC12 will think a probe is connected and limit the speed of the axes.

My SDEF's output connects to the CNC PC's Input 6. This input is defined as ProbeTripped. When touchdown occurs, the contacts close. Notice that this requires that the Input Type be set to NC (Normally Closed).

The SDEF provides continuity when it touches an edge. So "normal" for ProbeTrip is when the probe has tripped. Makes sense. "Normal" for ProbeDetect is when the probe has NOT been detected. When the probe is detected the line goes low and that is the off-normal state. Confused me.

¹⁵ With a Centroid probe, this input is pulled low when the probe is attached. This is an essential feature when the probe's output only goes low at touchdown. ProbeDetect provides a positive indication that the probe is connected before touchdown occurs.

Appendix III: The code that runs on the ATTiny85

```
/*
; *****
; * Software Defined Edge Finder          *
; * Version 2.0                          *
; This work is licensed under the Creative *
; Commons Attribution 4.0 International License. *
; To view a copy of this license, visit    *
; http://creativecommons.org/licenses/by/4.0/ *
; or send a letter to Creative Commons,    *
; PO Box 1866, Mountain View, CA 94042, USA. *
; * by R.G. Sparber                      *
; *****
;
;
;           INTERNAL 1 MHZ CLOCK
; =====
;  H A R D W A R E   I N F O R M A T I O N
; =====
;
; physical pin logical pin name use
; 1    PB5      NotReset, ADC0,PCINT5
; 2    PB3      ADC3 negative analog input
; 3    PB4      ADC2 positive analog input
; 4    ground
; 5    PB0      PB0  opto positive drive
; 6    PB1      PB1  opto negative drive
; 7    PB2      PB2  power control output
; 8    power
; =====
;  P O R T S   A N D   L O G I C A L   P I N   N A M E S
; =====
*/
const int PowerControlPin = 2;
const int TouchdownPositive = 0;
const int TouchdownNegative = 1;
const int SkinnyPrintPin = 2; //logical port that will be used as output
/*
; =====
;  C O N S T A N T S
```

```

; =====
*/
const long OneSecondMS = 1540;//in milliseconds
const long TouchdownWindowTimerMS = 60000;//If not activity after intial
touchdown within this time, shut down
const long HalfSecondMS = 770;
const long TenSecondsMS = 15400;//not used
const long ThirtySecondsMS = 46200;
const long OneMinuteMS = 92400; //note used
const unsigned int MinimumSwing = 100; //Vx swing is 0.05 mV times this
number
/*
The 10 bit ADC (0 - 1023) uses a 1.1V reference so one count is (1.1V/1023) = 1.1
mV. Then there is a X20 amp in front of it so one LSB reflected to the input is 1.1
mV/20 = 0.05 mV per count. The test current is 50 mA so one count represents
(0.05 mV/0.05 amps) = 1.1 milliohms. 20 counts is 22 milliohms. That is enough
to detect touchdown but noise margin is small.
*/
const unsigned int MaximumTouchdown = 800; //need room above Touchdown to
see Free
/*
; =====
;   V A R I A B L E S
; =====
*/
unsigned int Reading = 0; //put in due to debugger
unsigned int Touchdown = 0;//ADC output at touchdown
unsigned int Free = 0; //ADC output when free from reference surface
unsigned int Threshold = 0; //touchdown threshold
unsigned long OneMinuteTimerStartTime = 0;
boolean LastTD = false;
boolean TouchdownWindowTimerRunning = false;
boolean TouchdownWindowTimerDone = false;
unsigned long TouchdownWindowTimerStartTime = 0;
unsigned long LowBatteryTimer = 0;
unsigned long StartOfMinuteMS = millis();
unsigned int sample = 0;
/* _____ debugger
; =====
;   C O N S T A N T S

```

```

; =====
*/
unsigned int InterByteTimeMicroS = 250; //time from the transmission of the last
bit of one byte to the header of the next byte
unsigned int TimeUnitMicroS = 1500; //a TimeUnit is actually about 0.2
milliseconds long than this value
unsigned long A_MinuteMS = 60000; //used for overall power down
/*
; =====
;   V A R I A B L E S
; =====
*/
unsigned int Distance = 0;
byte DotCount = 0; //used by Heartbeat()
byte ByteBeingSent = 0;
byte Stack[12]; //holds 10 or fewer bytes waiting to be transmitted plus the
overflow symbol of FF FF
byte LocalTimeUnitCounter = 0;
byte SkinnyTransmitterCounter = 0; //used to shift stack down
byte StackPointer = 0; //next available location in stack
byte GreenDipRate = 0;
boolean StackOverflowQ = false;
boolean SendingStreamQ = false;
byte ReadCycle = 0;
byte FlashCycle = 0; //used to flash green LED to signal reading's proximity to
Threshold.

unsigned long start = 0;
unsigned long LastTransmission = millis(); //used by Heartbeat()

byte TimeUnitCounter = 0;

unsigned long TimeLastBitSentMicroS = micros(); //used to tell when to start
sending next byte

unsigned long TimeTimeUnitSentMicroS = micros() - 2*TimeUnitMicroS; //used
to tell when to send next TimeUnit. Initialized to insure immediate transmission

unsigned long TimeAllOfByteSentMicroS = micros(); // used to tell when to send
next byte. I set it so first byte is sent without delay.

```

```

//_____debugger

/*
;=====
;  S E T U P
;=====
*/
void setup(){
//direct writes to hardware registers to set up analog and ports
PORTB = 0b00000000; //turns off all pull up resistors
DDRB = 0b00000111; //sets both inputs and outputs except diagnostic port
ADCSRA = 0b10000011; //enable ADC but do not run it yet
ADCSRB = 0; //this is the default. I want Unipolar input, no input polarity
reversal, free running ADC although I am running it single.
DIDR0 = 0b00011000; //turn off unneeded digital inputs on analog pins

//first do battery test
ADMUX = 0b10010010; // set internal reference to 2.56 volts (bit 7 = 1, bit 6 = 0,
bit 4 = 1), analog inputs to single ended (ADC2) with gain of x 1 (MUX[3:0] =
0010 (bits 3, 2, and 0 = 0, bit 1 = 1).
PowerUp(); //keep power up, turn on LED to tell user to remove bridge. Start timer
that turns circuit off in 1 minute if no touchdown found
if (ReadVoltage() < 629 ){ //when the batteries get to 1.1V each, call for
replacements.
//do not proceed with edge finder function. Alternately flash red/green at 0.5
second intervals for 10 seconds and then turn off power. If auto powerdown
inhibited, keep warning user
LowBattery:
LowBatteryTimer = millis();
while((millis() - LowBatteryTimer)< 10000){ //flash LEDs for 10 seconds
GreenOffOptoAndRedOn();
delay(500);
GreenOnOptoAndRedOff();
delay(500);
}
//then kill power
PowerDown(); //due to filter cap, ATTiny85 stays on for about 2 seconds after
power is removed
goto LowBattery;//if power down has been inhibited, continue to warn user and do
not proceed

```

```

}
//set up for Edge Finder
ADMUX = 0b10000111; //set internal reference to 1.1 volts, analog inputs to
differential with gain of x 20. Prescaler at 8.

//pinMode(SkinnyPrintPin,OUTPUT); //assignment for debugger

delay(OneSecondMS); //keep LED on long enough to be seen
Touchdown = ReadMinimumVoltage(); //read voltage assuming user has bridged
gap. Record minimum voltage seen in NumberOfCycles samples. Machine is not
moving so readings should be clean; Touchdown is an unsigned int so can hold up
to 65,535 while the DAC can output up to 1023.
if (TouchdownTooLargeQ() == true){
SignalStartOverAndDie(); //Flash LED 5 times and power down unless inhibited by
user. Then we just keep trying to get user to fix problem. We do not proceed.
}

//only get here if Touchdown is small enough

SignalReady(); //Signal the user to remove the bridge. This is done by turning off
the LED and also keeping opto off.
delay(OneSecondMS); //give user time to react and remove bridge before starting
to read
ReadVoltageWhenOffOfSurface:
Free = ReadMaximumVoltage(); //read voltage assuming user has removed bridge
gap. Record maximum voltage seen. Machine is not moving so readings should be
free of noise.
if(LargeEnoughDifferenceQ() == true){
CalculateThreshold();
}else{
SignalRetry(); //flicker of LED tell user to turn spindle a little
GiveUpSoPowerDownQ(); //power down if running more than 10 seconds unless
power down has been inhibited by the user. Then continue to flicker LED.
goto ReadVoltageWhenOffOfSurface;
//user will see continuous flickering until Free is large enough or until 10 seconds
have passed
}
}
}

/*

```



```

;=====
;   L O O P
;=====
*/
void loop(){//scan input and report touchdown to CNC PC via opto closure
AutoPowerDownAfterOneMinute(); //don't run longer than one minute
if (TouchdownQ() == true){
SignalTouchdown();//tell user plus if this is the first touchdown, set LastTD to true
in order to start TouchdownWindowTimer.
}else{
SignalNoTouchdown();
}
NormalPowerDownQ();//power down if running more than
TouchdownWindowTimerMS after first touchdown detected
TouchdownWindowTimer(); //timer is used by NormalPowerDownQ()
}
/*
;=====
;   S U B R O U T I N E S
;=====
*/

void AutoPowerDownAfterOneMinute(){
if (millis() - StartOfMinuteMS > A_MinuteMS) PowerDown(); //don't run longer
than one minute
}

boolean TouchdownTooLargeQ(){
if (Touchdown > MaximumTouchdown){
return true; //touchdown too large to be able to detect Free
}else{
return false;
}
}

void SignalStartOverAndDie(){
TopOfStartOver:
int FlashCount=0;
while(FlashCount<10){
GreenOnOptoAndRedOff();
}
}

```

```

delay(100);
GreenOffOptoAndRedOn(); //turn on opto to prevent probe crash. User must
notice that we are not at touchdown yet CNC12 says we touched plus red led was
flashing to indicate problem. If auto power down is overridden, CNC12 will keep
getting touchdown signal.
delay(100);
++FlashCount;
}
PowerDown(); //turns off power to entire circuit
//if power down overridden, keep signaling to start over
goto TopOfStartOver;
}

void PowerUp(){
OneMinuteTimerStartTime = millis();//start normal power down timer
digitalWrite(PowerControlPin,HIGH);//keep power on until done
GreenOnOptoAndRedOff(); //ack power is on
}

unsigned int ReadMinimumVoltage(){

unsigned int ReadCycle = 0;
unsigned int MaxNumberOfReadings = 2000;

unsigned int LastReading = 1023;//start at maximum possible value
unsigned int CurrentReading = 0;

while (ReadCycle<MaxNumberOfReadings){
CurrentReading = ReadVoltage();
if (CurrentReading < LastReading){
LastReading = CurrentReading; //current reading is smaller so save it
}else{
return CurrentReading; //Current reading the same as last or larger so we are stable
or rising.
}
++ReadCycle;//still looking for minimum but only try up to
MaxNumberOfReadings times
}
return CurrentReading; //after MaxNumberOfReadings performed we are still
falling so just return current reading.

```

```
}
```

```
unsigned int ReadMaximumVoltage(){  
    unsigned int ReadCycle = 0;  
    unsigned int MaxReadings = 200;  
    unsigned int LastReading = 0;//start at minimum possible value  
    unsigned int CurrentReading = 0;  
    while (ReadCycle<MaxReadings){  
        CurrentReading = ReadVoltage();  
        if (CurrentReading > (LastReading)){  
            LastReading = CurrentReading; //current reading is larger so save it  
        }else{  
            return CurrentReading; //Current reading the same as last or smaller so we are  
            stable or falling.  
        }  
        ++ReadCycle;  
    }  
    return CurrentReading; //after MaxReadings readings we are still rising so just  
    return current reading.  
}
```

```
unsigned int ReadVoltage(){ //takes an average over NumberOfCycles  
    unsigned int sum = 0;  
    unsigned int result = 0;  
    byte NumberOfCycles = 10;  
    ReadCycle = 0;  
    while(ReadCycle<NumberOfCycles){  
        ReadADC(); //updates sample  
        sum = sum + sample; //add up samples to average out noise; sum is unsigned  
        integer  
        ++ReadCycle; //increment counter  
    }  
    result = sum/NumberOfCycles; //calculate average result; sum could be up to  
    64,000 but result will be small enough to fit into an int.  
    return result;  
}
```

```
void SignalReady(){//turn LED off to tell user to remove short but keep opto off
AllOff();
}
```

```
boolean TouchdownQ(){
//define TD as 2 consecutive readings < Threshold
//a single reading > Threshold returns false
byte HitCounter = 0;
while(HitCounter < 2){
ReadADC(); //it outputs sample
if (sample < Threshold){
++HitCounter;
}else{
return false; //a single reading > Threshold says no touchdown
}
}
return true;
}
```

```
void SignalTouchdown(){
GreenOffOptoAndRedOn(); //user sees LED go out and CNC PC detect
touchdown
delay(200); //wait 200 ms to insure PC sees touchdown
if (LastTD == false) LastTD = true; //signal that we have had a touchdown
}
```

```
void SignalNoTouchdown(){
GreenOnOptoAndRedOff(); //user sees LED on and CNC PC waiting for
touchdown
}
```

```
void GiveUpSoPowerDownQ(){ //if we have waited long enough to reach
touchdown but failed, turn off power
if ((millis() - OneMinuteTimerStartTime) > OneMinuteMS){
//to prevent a probe crash if we time out, force a touchdown signal to stop CNC
from advancing. This is not an ideal situation since it gives a false TD indication
but does prevent damage.
    for(byte i = 0;i < 10;i++){
        delay(500);
    }
}
```

```

        GreenOffOptoAndRedOn();
        delay(500);
        GreenOnOptoAndRedOff();
    }
    PowerDown(); //turns off power to entire circuit
    return; //if auto power down overridden, return to reporting touchdown state
}else{
    return;
}
}
}

```

```

void NormalPowerDownQ(){ //CNC is doing touchdown, back away, touchdown
sequence which will be done within TouchdownWindowTimerMS of first
touchdown so power down
if (TouchdownWindowTimerDone == true){
//power down since we have waited long enough for second touchdown
    PowerDown(); //turns off power to entire circuit
    return; //if auto power down is overridden, return to reporting touchdown state
}
}
}

```

```

void PowerDown(){ //due to large filter capacitor, it does take time for the device
to lose power
digitalWrite(PowerControlPin,LOW); //turns off power to entire circuit unless
inhibited by user
return;//if auto power down inhibited by user, we will still be running so return
}

```

```

void TouchdownWindowTimer(){
/*
Start the timer by setting LastTD flag true. Timer sets the
TouchdownWindowTimerRunning flag true until ten seconds are up. Then it
changes flag to false and sets the TouchdownWindowTimerDone flag true.
*/
if (TouchdownWindowTimerDone == true)return;//only get here if auto shut down
has been inhibited by user
if((LastTD == true) && (TouchdownWindowTimerRunning == false)){
//just had first touchdown so start timer if it is not running or done
TouchdownWindowTimerStartTime = millis();
TouchdownWindowTimerRunning = true;
}
}

```

```

TouchdownWindowTimerDone = false;
return;
}
if ((TouchdownWindowTimerRunning == true) && ((millis() -
TouchdownWindowTimerStartTime) > TouchdownWindowTimerMS)){
//time is up
TouchdownWindowTimerRunning = false;
TouchdownWindowTimerDone = true;
return;
}
}

```

```

void SignalRetry(){ //flash LED 20 times to tell user to turn spindle a little. It looks
like a flicker.

```

```

int i = 1;
while(i<21){
GreenOnOptoAndRedOff();//will be LED on and opto off
delay(2*i);//causes ramp up in on time
AllOff();
delay(55 - 2*i);
++i;
}
}

```

```

boolean LargeEnoughDifferenceQ(){
if (Free > (Touchdown + MinimumSwing)){
return true;
}else{
return false;
}
}

```

```

void CalculateThreshold(){
Threshold = (Free +(19*Touchdown))/20; //5% above Touchdown
}

```

```

void GreenOffOptoAndRedOn(){
digitalWrite(TouchdownPositive,LOW);
digitalWrite(TouchdownNegative,HIGH);
}

```

```

void GreenOnOptoAndRedOff(){
digitalWrite(TouchdownPositive,HIGH);
digitalWrite(TouchdownNegative,LOW);
}

```

```

void AllOff(){
digitalWrite(TouchdownPositive,HIGH);
digitalWrite(TouchdownNegative,HIGH);
}

```

```

void ReadADC(){
//a dip in physical pin 7 signals start of ADC read; used to measure cycle rate of
Vx sampling while looking for Touchdown
unsigned int lower = 0;
unsigned int upper = 0;
ADCSRA = 0b11000011;//ADC was enabled in setup. start conversion
NotDone:
if((ADCSRA & 0b00010000) == 0b00000000)goto NotDone;//wait until the ADC
Conversion is done. Bit ADIF goes to 1 when done. It is cleared on next start.
//read ADCL first and then ADCH
lower = ADCL; //read ADCL first and then ADCH
upper = ADCH;
sample = (lower | (upper<<8));//then put them together.
return;
}

```

```

/*
; =====
; DIAGNOSTIC SUBROUTINES
; =====
*/

```

```

void SkinnyPrintByte(byte data){
if (StackOverflowQ == true)return;
if (StackPointer > (sizeof(Stack) - 1)){
StackOverflowQ = true;
return;
}
}

```

```
Stack[StackPointer] = data; //populate next available stack location
++StackPointer; //point to the new free stack location above the data just added
}
```

```
void SkinnyPrinter(){
Heartbeat();
MonitorStack();
SendFromStack();
SendStream();
}
```

```
void Heartbeat(){
if(NoTransmissionsInTenSecondsQ == true){
SkinnyPrintByte(0xFE);
} //a transmission is going out so no need for heartbeat
}
```

```
boolean NoTransmissionsInTenSecondsQ(){
if (TimeUnitCounter == 0){ //so no transmission now
if((millis()-LastTransmission)< 10000){
return false;
}else{
LastTransmission = millis();
return true;
}
}
//otherwise, a transmission is going out now
return false;
}
```

```
void MonitorStack(){
if (StackOverflowQ == true){
//flush stack and transmit 255 twice in a row
StackPointer = 2;
Stack[0]=0xFF;
Stack[1] = 0xFF;
Stack[2] = 0; //make backfill of stack 0 rather than random data
StackOverflowQ = false;
}
}
```



```

void SendFromStack(){
if (SendingStreamQ == true)return;
if(StackPointer == 0)return; //no byte waiting to be sent
if ((micros() - TimeAllOfByteSentMicroS) < InterByteTimeMicroS)return;
PrepareToSendStream(Stack[0]);
SkinnyTransmitterCounter = 0; //shift down stack and adjust StackPointer
while(SkinnyTransmitterCounter < StackPointer){
    Stack[SkinnyTransmitterCounter] = Stack[SkinnyTransmitterCounter+1];
    ++SkinnyTransmitterCounter;
}
--StackPointer;
return;
}

void PrepareToSendStream(byte data){
SendingStreamQ = true;//we are now Sending a Stream
ByteBeingSent = data; //load supplied byte as the one about to be sent
}

void SendStream(){
if (SendingStreamQ == false)return;
if ((micros() - TimeTimeUnitSentMicroS) < TimeUnitMicroS)return;
if (TimeUnitCounter <7){ //header is 0 through 6
SendHeader();
}else{ //TimeUnitCounter should be 7 through 30
SendBits();//when done sending 8 bits it returns TimeUnitCounter to 0 and
SendingStreamQ to false
}
if(SendingStreamQ == true){ //if sending stream, advance TimeUnitCounter but if
done, do not advance it
TimeTimeUnitSentMicroS = micros();//prepare to send next TimeUnit
++TimeUnitCounter;//prepare to send next TimeUnit
}
}

void SendHeader(){
switch(TimeUnitCounter){
case 0:
    digitalWrite(SkinnyPrintPin,LOW);

```

```

    break;
case 1:
    digitalWrite(SkinnyPrintPin,HIGH);
    break;
case 2:
    digitalWrite(SkinnyPrintPin,LOW);
    break;
case 3:
    digitalWrite(SkinnyPrintPin,HIGH);
    break;
case 4:
    digitalWrite(SkinnyPrintPin,HIGH);
    break;
case 5:
    digitalWrite(SkinnyPrintPin,HIGH);
    break;
case 6:
    digitalWrite(SkinnyPrintPin,LOW);
    break;
}
}

```

```

void SendBits(){
LocalTimeUnitCounter = (TimeUnitCounter - 7)%3; //this is the relative
TimeUnit position within a bit
    if((1 & ByteBeingSent) == 0){ //this will extract the current LSB and if true we
must send a logic 0
        switch(LocalTimeUnitCounter){
            case 0:
                digitalWrite(SkinnyPrintPin,HIGH);
                break;
            case 1:
                digitalWrite(SkinnyPrintPin,LOW);
                break;
            case 2:
                digitalWrite(SkinnyPrintPin,LOW);
                break;
        }
    }else{
//the LSB is a logic 1

```

```

switch(LocalTimeUnitCounter){
  case 0:
    digitalWrite(SkinnyPrintPin,HIGH);
    break;
  case 1:
    digitalWrite(SkinnyPrintPin,HIGH);
    break;
  case 2:
    digitalWrite(SkinnyPrintPin,LOW);
    break;
}
}
if (LocalTimeUnitCounter == 2)ByteBeingSent = ByteBeingSent>>1; //shift bits to
the right one place so bit to the left of the previous LSB becomes the LSB
if (TimeUnitCounter >29){ //should be 30 after last bit sent

TimeUnitCounter=0;//prepare for next transmission
SendingStreamQ = false;
TimeAllOfByteSentMicroS = micros();//time stamp end of transmission
}
}

void SkinnyPrintInteger(int data){ //debugger
SkinnyPrintByte(highByte(data));
SkinnyPrintByte(lowByte(data));
}

void Diagnostic(){
GreenOffOptoAndRedOn();
delay(50);
GreenOnOptoAndRedOff();
delay(50);
AllOff();
}

```