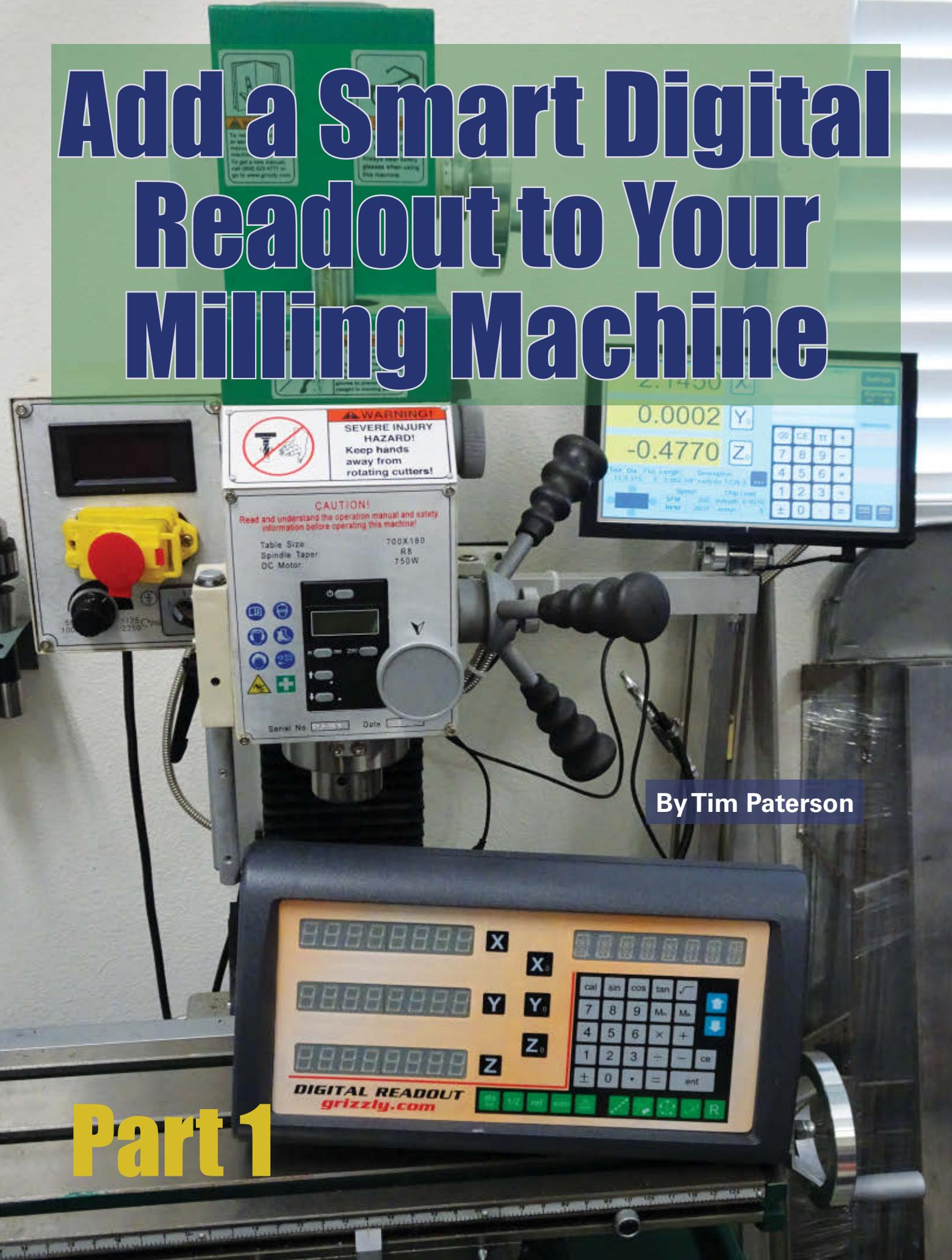


# Add a Smart Digital Readout to Your Milling Machine

By Tim Paterson

## Part 1



## If you already have a manual milling machine with a digital readout (DRO), then see if any of these experiences I've had seem familiar:

- I carefully establish my starting point using an edge finder and zero the corresponding DRO axis. I forget to account for the radius of the edge finder until I'm ready to cut and notice it just doesn't look like the cutter or drill is in the right place.
- I set zero on the first axis, then crank the table so I can apply the edge finder to another side. Once I have it nicely aligned, I hit the zero on the wrong DRO axis — the one that I just set — and have to go back and do that one over.
- With both X and Y axes all set up, I'm ready to start cutting with my favorite roughing tool: a 3/8" carbide end mill. If my drawing says to put an edge at, say, 2.145", I have to add (or maybe subtract) the cutter's 0.1875" radius to get the correct position to set the DRO. It's too many digits to trust doing it in my head, so I end up writing addition or subtraction problems at all the dimensions of my drawing.

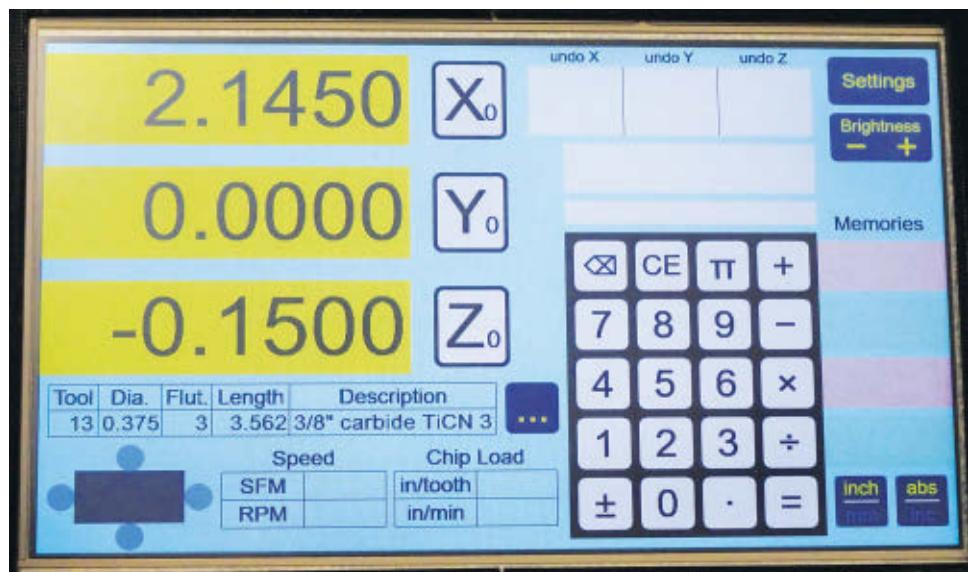
mentor for the FIRST robotics club at a nearby high school (go Ragin' C-Biscuits!) and they have two big Bridgeport mills, each with an ancient DRO. One of the displays actually has to warm up before it shows anything. Both of them have broken X axis position sensors. So, one day I was surfing the web to see what it would take to replace them, and I got to thinking that I could build a DRO — at least the display head.

This project idea had several things going for it. For starters, I'd get to fix the common pitfalls I just mentioned. I'd also been reading about touchscreen projects on the pages of *Nuts & Volts* and thought maybe it was time for me to get on the bandwagon. This seemed like a project I could actually finish (unlike, say, my robot) and I did, in about six months.

## Feature Set

How does my own design make the DRO easier to use? The crux is for the DRO to know the diameter of the tool. **Figure 1** shows the main screen, and you can see below the Z axis readout there's a small table with entries labeled "Tool," "Dia.," etc. The tool diameter can be entered by simply tapping in the value on the touch keypad, and then tapping the "Dia." column.

However, this is not enough because it needs to know what side of the stock is being cut — whether to add or



**Figure 1** – Main screen of the digital readout.

subtract the tool radius. Right below the tool table is a dark rectangle with a circle on each side. That represents a top view of the workpiece with the cutting tool on any of its four sides. Touching the spot where the tool contacts the work enables a radius offset for the corresponding axis.

Two adjacent sides can be selected at once, offsetting both X and Y. **Figure 2** shows the cutter offset enabled on two sides. The displays have some added color to remind you that the cutter offset is in effect. (The display resolution is 0.0002", so it can't display the exact offset of 0.1875").

It was somewhat of an afterthought that I realized I could easily add basic calculations for speeds and feeds. By entering a desired surface feet per minute, the corresponding RPM is calculated for the given tool diameter.

By adding an entry for the number of flutes on the tool, then the feed rate can be calculated in inches per minute for a chip load that you enter in inches per tooth. So, how would you know if you're cranking at the right feed rate? Anytime you crank the X and/or Y axes, your current feed rate appears in the dark rectangle.

You'll also notice the tool info table has an entry for length. If you have a way to measure the exact length of your tool as mounted in the spindle (such as the Tormach Tooling System), then this will automatically adjust your Z axis as you change to tools of different lengths.

The display allows you to directly enter these tool specifications on the main screen using the keypad as you change tools. However, the real power of this is to keep a tool library so you only have to enter the information once. The unit can store 500 tools in Flash memory, giving you access by a tool number that you assign or by scrolling through a list.

By tapping the “...” button to the right of the tool info table, you open the full tool library as shown in **Figure 3**. Here you can enter new tools, update existing ones, and select the next tool you're using. The list can be dragged up and down with your finger, or the yellow scroll thumb can be dragged to move through the list rapidly.

If you have a lot of tools or you want to stay synchronized with another tool library (for example, you use G-Wizard to calculate feeds and speeds), then manually entering all your tools isn't very convenient. That's why there's a USB port for a Flash drive that allows you to import your tool list from another source. It accepts a “comma separated values” (CSV) format that can be directly written by Excel. It can also export the list to save as a backup or transfer to another device or program.

But enough about tools. What about that problem of zeroing the wrong axis? Back in **Figure 1**, you can see areas in the upper right of the screen labeled “undo X,” etc. Every time you change the origin of an axis, the difference is recorded in the undo list. It saves eight levels of undo, showing the last three on screen. Like my original Grizzly display head, it supports two coordinate systems (called “abs” and “inc,” although there is no significance to the names) with separate undo lists for each one.

Finally, there's the keypad with calculator. You can tap an axis display to load a value into the calculator and just as easily move a calculated value into a display. There are four memories whose contents are always visible. Plus, the

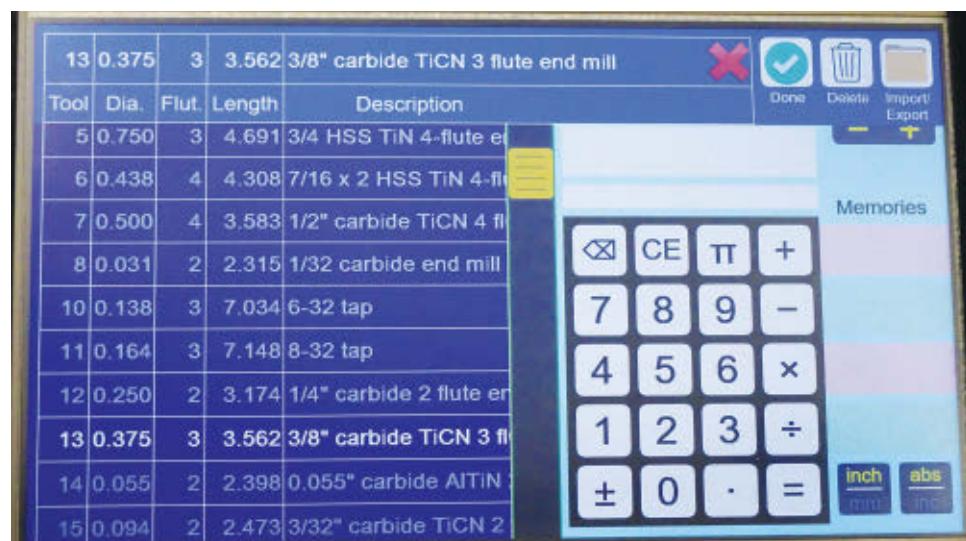


**Figure 2** – Cutter radius offset is enabled for both the X and Y axes.

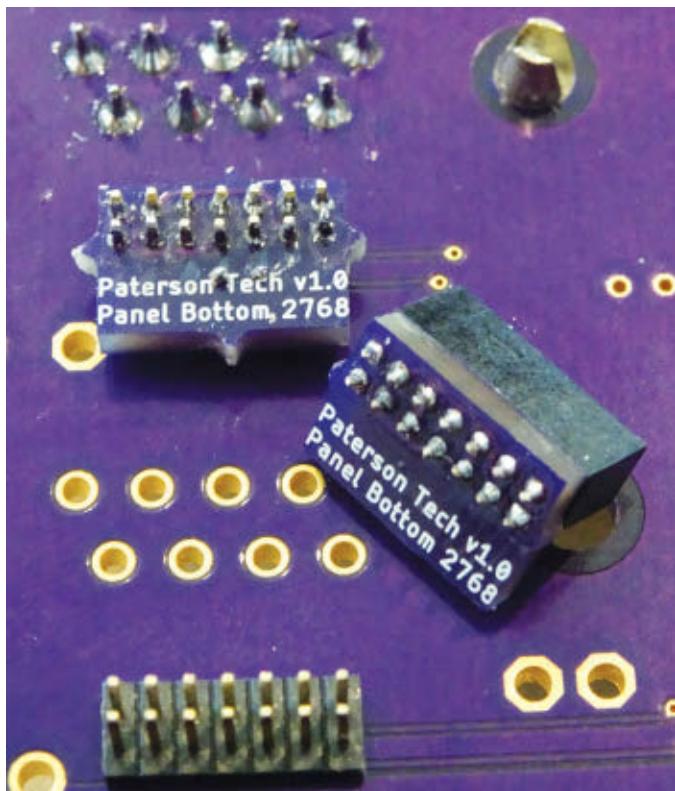
backspace key makes corrections easy while you enter.

## Mill Interface

As I shopped around for DRO kits, one thing that seemed consistent was that the position sensors connect to the display head using one nine-pin D connector (like a serial port) for each axis. What wasn't consistent was the pinout of the connector – when you could find it. The basic signals are two power pins providing 5V and two position encoder signals called A and B. These could be on any pins, and sometimes the A and B signals were each a differential pair:



**Figure 3** – Tool library screen, with selected tool highlighted in the scrolling list.



**Figure 4** – The tiny interconnect PCB can be soldered directly to the pin header or be removable using a socket.

A+, A- and B+, B-.

To have a chance at dealing with all the variations, I designed the circuitry to pass the connections for each axis

through a small 14-pin 50 mil header. That's enough to gather the nine pins of the D connector and the four power and signal connections to the circuitry. A separate tiny circuit board (0.25" x 0.36") routes the four necessary connections.

I use OSH Park for printed circuit boards (PCBs). They charge a flat rate of \$5 per square inch to deliver three boards. The price of this little interface board is 45 cents, shipping included! I could splurge and pay double for their "swift" service, and the price would still be less than a buck. That gets me all three I need for X, Y, and Z.

The tiny pinout adapter PCBs can be installed in one of two ways: the PCBs can be soldered directly to the pin headers, making them permanent; or the PCBs can be put on female sockets that plug into the pin header. I've done it each way on the two prototypes I've made, as shown in **Figure 4**.

I couldn't find documentation on the pinout for my Grizzly DRO, but it wasn't hard to figure it out with a voltmeter. Probing around the open connector on the DRO head quickly revealed where power and ground were. I then had to make a little adapter that gave me access to the pins while the position sensor was plugged in.

With ground known, I easily found the two pins that changed state when I turned the axis handwheels. The result: ground is pin 2; +5V is pin 7; A is pin 6; and B is pin 8 (A and B are interchangeable). I use that pin order to name the interface, so I call it type 2768.

A typical position sensor has a resolution of five microns (about 0.0002"). If an axis handwheel was turned at a constant speed, you would see a square wave on the A and B signals, with a 90° phase shift. The direction of the phase shift – whether B is leading or trailing A – gives the direction of travel. This is the classic quadrature signal pair of an incremental encoder. Each transition on either A or B is one step at the resolution.

For my own mill, I replaced an existing display and used the OEM-installed position sensors. The sensors are available separately. It appears there's optical ones you have to buy to length and magnetic ones that can be cut to length.

## Hardware Design

I decided I needed a 10" touchscreen, and the only candidate I could find was at **BuyDisplay.com**. The unit was about \$85 plus a surprising \$30 for shipping from China. That includes a resistive touch overlay and a control board the size of

## Compact. User Friendly. Highly Capable.

CNC MILLS | CNC LATHES | CNC PLASMA TABLES | CNC ROUTERS | AUTOMATED BANDSAWS



We make CNC machines **people can buy**. They **fit in places** other CNCs can't.

Those who are new to CNC love Tormach equipment because our very own PathPilot control is **easier to learn and use** because it's **more intuitive**.

Experienced machinists appreciate Tormach machines because they **reliably turn out parts with any CAD/CAM**.

Build yours at [tormach.com/loop](http://tormach.com/loop).



**TORMACH®**

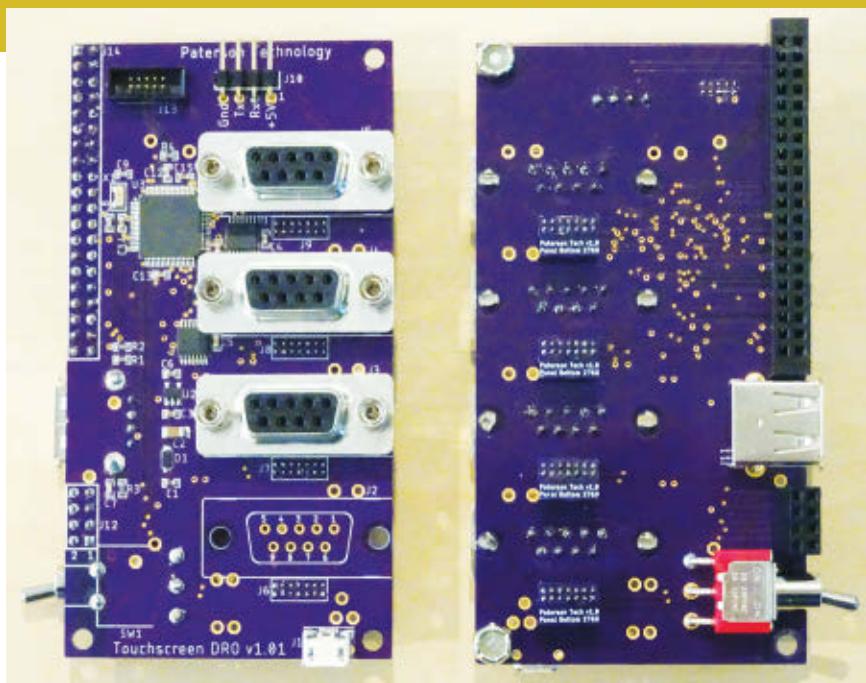
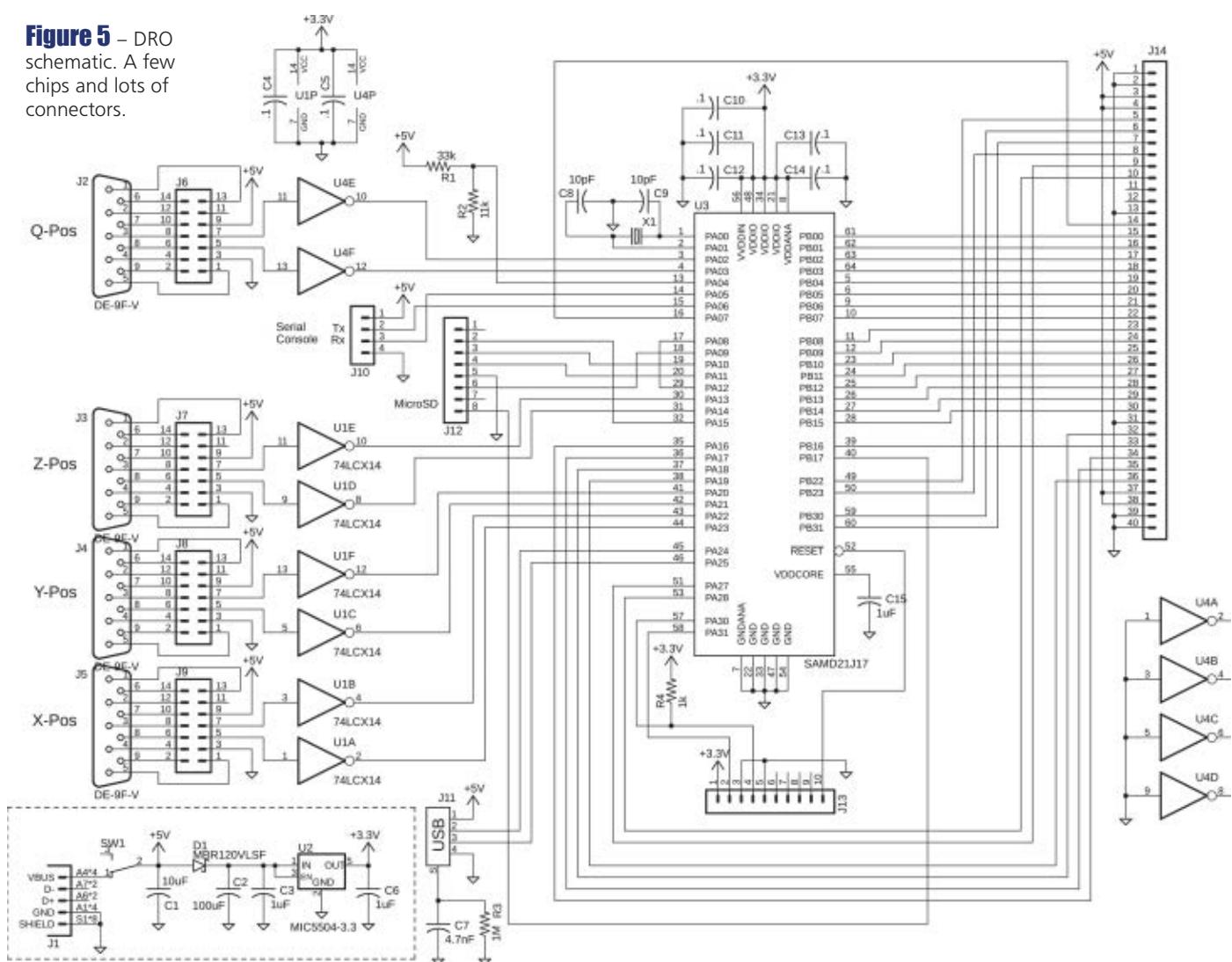
**Figure 6** – Front and back views of the assembled PCB.

the screen that has a graphics controller, 16 MB of video RAM, a resistive touch controller, and microSD card slot. I also ordered a 128 Mb (16 MB) serial Flash chip from them at the same time.

I love hardware design and making purpose-built digital circuits, so of course this would be no different. The schematic is shown in **Figure 5**. It's not much more than a Microchip SAM D21 microcontroller unit (MCU) and a bunch of connectors. The PCB made from this design plugs directly into the interface connectors on the touchscreen and is secured with a standoff.

Along the left side of the schematic, you see the interface to the position sensors. I

**Figure 5** – DRO schematic. A few chips and lots of connectors.



included a fourth "Q" input for a quill but haven't used it. U4, J2, and J6 are not needed at all to just have three input sensors for X, Y, and Z. I discovered the 74LCX logic family while researching for this design; it's perfect for interfacing the 5V sensors to the 3.3V MCU.

One part of the schematic I'd like to explain is the power-off detection. In the lower left corner, you see the power circuit in dashed lines. The touchscreen takes about 500 mA at 5V, while the MCU runs on 3.3V. When the power is switched off, diode D1 blocks capacitor C2 from discharging rapidly into the touchscreen, holding up the 3.3V supply for the MCU briefly. Near the top of the schematic, R1 and R2 are dividing the 5V supply by four.

What's not obvious is that the divided voltage is being routed to an analog comparator in the MCU with an accurate internal reference of 1.1V. The output of that comparator is on MCU pin 29, which is wired around to the non-maskable interrupt (NMI) on MCU pin 17. So, when the 5V supply drops to 4.4V, NMI triggers. Software takes over and quickly saves the current axis positions in Flash memory so they can be restored on power-up. I've measured 5-10 ms from the time NMI triggers until the 3.3V supply begins to droop; plenty of time to get the work done.

**Figure 6** shows the front and back of the assembled prototypes. The front of the PCB (which will actually face out the back side of the assembled unit) has a four-pin connector on the top for a TTL serial connection. At the bottom is a USB micro B jack just for power. On the back of the PCB, you can see the four tiny interconnect PCBs for matching the position sensor's pin-out.

There's also the standard USB-A connector where a

## Resources

Project on GitHub

[https://github.com/TimPaterson/  
TouchscreenDigitalReadout](https://github.com/TimPaterson/TouchscreenDigitalReadout)

FontGenerator

[https://github.com/TimPaterson/FontGenerator-  
embedded-systems](https://github.com/TimPaterson/FontGenerator-embedded-systems)

ScreenDesigner

[https://github.com/TimPaterson/ScreenDesigner-  
for-touchscreens](https://github.com/TimPaterson/ScreenDesigner-for-touchscreens)

Flash drive can be attached.

## Software Overview

I wrote the software for "bare metal" — no operating system or support libraries (except standard C runtime). I've had previous experience with some of the software elements like USB drivers and a FAT file system for the Flash drive. However, the graphics and touch drivers were all new to me and took a lot of development time.

When I ordered the touchscreen, I added a font serial ROM option. After a little experimentation, I concluded that the fonts it provided were of no help. Sizes were very limited and too small, and the fonts themselves were ugly.

My solution was to write a Windows app to generate font bitmaps from any Windows font. Called FontGenerator, you can check it out on GitHub (see **Resources**). I ended up using it to give me four fonts in heights of 24, 36, 48, and 96 pixels. The largest one is only used for the axis readouts and doesn't have a full character set (just space through '9', 0x20-0x39).

The RA8876 graphics controller chip used on the touchscreen has what they call a color expansion function. It will copy a rectangular black and white bitmap and substitute any two colors for one and zero (or you can choose to leave existing pixels unchanged on zero). This is perfect for writing individual characters in full color, and it's extremely fast.

My next problem was designing the graphics for the touchscreen.

**THE MOST COMPLETE PROGRAMMING TUTORIAL!**

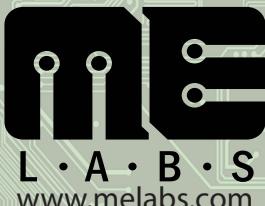


**\$99.99**  
Academic and bulk pricing available

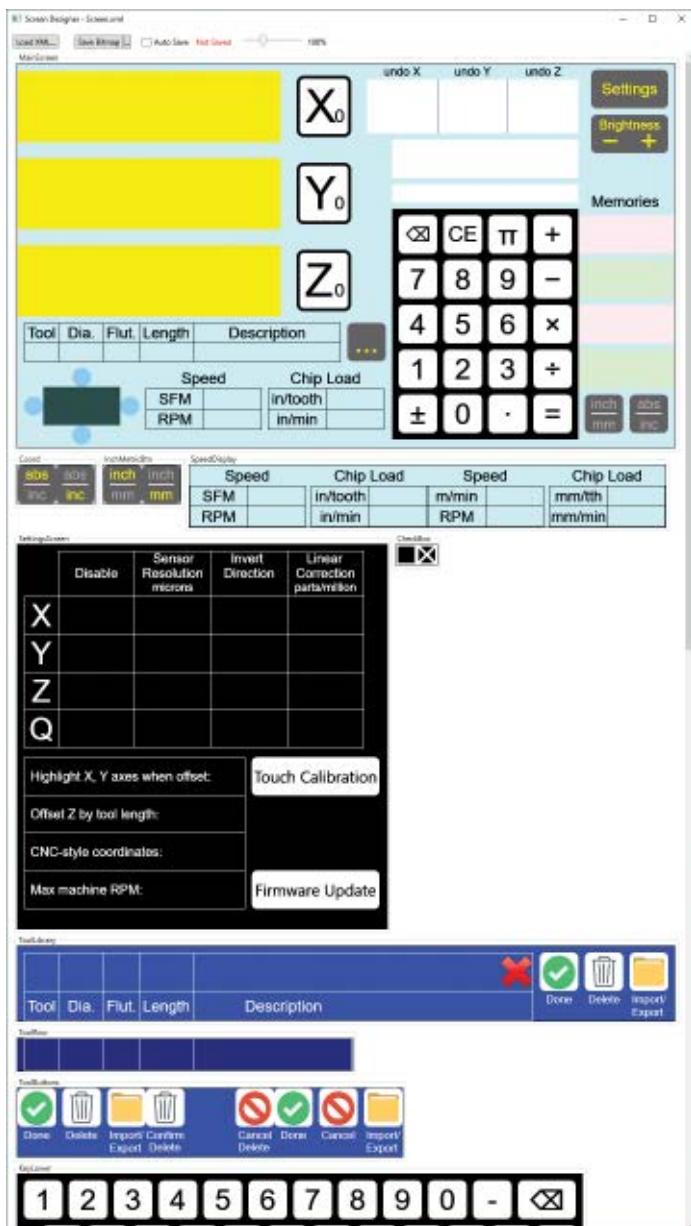
- Everything you need to learn to program PIC® microcontrollers is included in this package
- Learn the fundamental concepts of programming, including program flow, loops, coding techniques, binary manipulation, device-to-device communication, user interface design, in-circuit debugging, and more!
- Includes all the necessary the software, compiler, trainer board, cable, tutorial with exercises and sample code!

Check out our other popular products for PIC® developers including:  
U2 Programmer  
PBP3 Compiler  
Prototyping Boards  
Experimenter Boards

Contact us at [info@melabs.com](mailto:info@melabs.com)



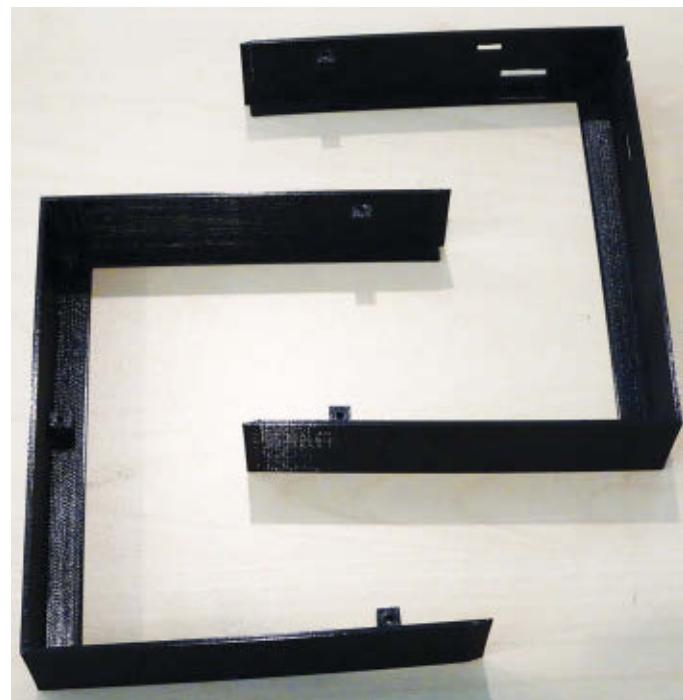
[www.melabs.com](http://www.melabs.com)



**Figure 7** – Some of the graphic images for the DRO, which all get loaded into the 16 MB Flash memory.

Not only did I need graphics, but I needed the size and location of things that acted as hotspots for touch or text areas. I searched for a tool that would work and didn't find anything. So once again, I ended up writing my own.

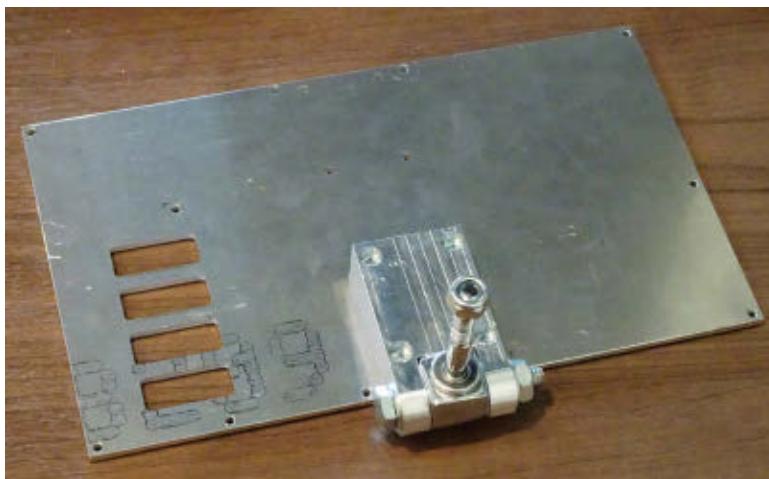
It's a Windows app called ScreenDesigner, also available on GitHub. It doesn't have graphical objects you can drag around on the screen and place visually – that's just over my head. Instead, it's essentially a programming language for graphics using XML. While creating the graphics, I used an XML text editor and as soon as the XML was saved to disk, ScreenDesigner updated its image. This was actually very usable. If I was tweaking the position of something by a few



**Figure 8** – The two halves of the screen bezel, 3D printed in PETG.



**Figure 9** – Mounting tab on the bezel showing 4-40 thread printed in.



**Figure 10** – Back plate is 1/8" aluminum to provide structural support.

pixels, I could type in a new number, save with Ctrl-S, and it would instantly appear in its new position.

**Figure 7** shows about half the graphics I created for the touchscreen. Most of it is 16-bit color, but the black and white images were done in eight-bit color to save upload time. Several images (see the first row after the main screen) are overlay arrays that are copied onto another screen.

For example, when you toggle between inch and millimeter units, the button you're pushing is overwritten by one that indicates the current state in yellow (the two-element array of buttons is indexed by the value of a Boolean). There are two-, three-, and four-element image arrays sprinkled in among the graphics.

FontGenerator and ScreenDesigner each spit out a binary file with fonts or images and a header file with all the data the program needs to use them. The two binary files get loaded into the serial Flash memory on the LCD. On startup, the RA8876 is instructed to copy the serial Flash to

video RAM, which it can do in one command. The total graphics size is about 4 MB, using only a quarter of available serial Flash and video RAM.

There's isn't room here to go further into the inner workings of the software. It's all on GitHub if you're interested (see **Resources**).

## Enclosure

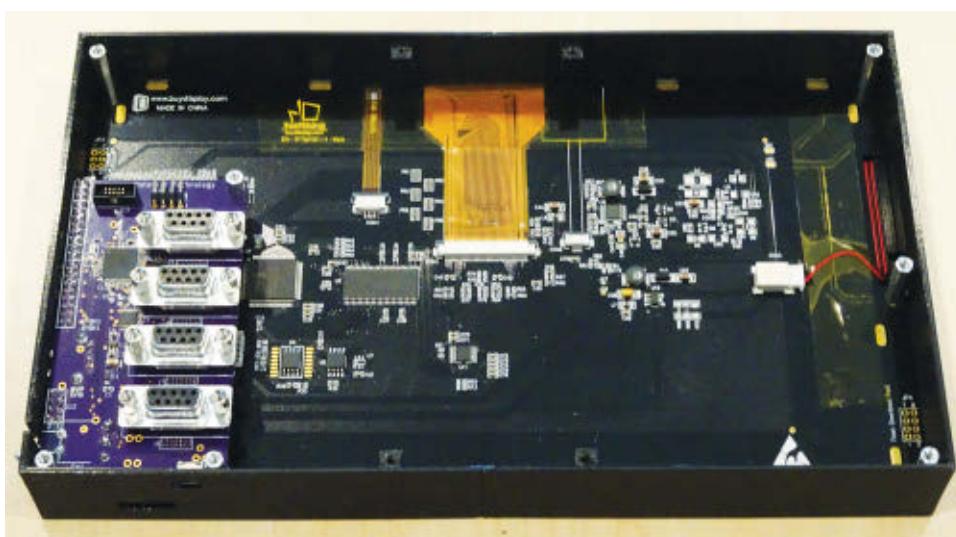
I modeled the touchscreen and electronics in Autodesk Fusion 360: a wonderful free-to-the-hobbyist CAD program. I used it to design an enclosure with 3D-printed bezel and an aluminum back. Unfortunately, the enclosure is about 1/4" longer than my 3D printer can print, so I had to split it into two. **Figure 8** shows the two plastic pieces, which would be interchangeable except for the cutouts in one of them.

**Figure 9** has a close-up of one of the small ears that the back plate mounts to. I was hoping you could see that it was printed with the 4-40 thread built in. The back side of the bezel pieces have similar threaded posts at all the mounting points of the touchscreen PCB. The threads of the standoffs go through the back of the PCB into the bezel which gives them a nice tight fit like a nylon lock nut. Except for acting as a nut in the way, the plastic pieces are not structural.

The back plate and its mounting bracket shown in **Figure 10** provide all the structural support. The piece sticking up is part of the original Grizzly mount. Ironically, I made these with my CNC mill. The back plate could easily be done on a manual mill and the bracket as well, if you don't bother with the rounded bottom. In **Figure 11**, you see the screen mounted in the bezel, ready for the back plate to be installed.

## Next Time

So far, I've tried to give you a good description of the hardware and software that makes up my project. Next time, I'll get into details of how to build one yourself and the basics of how to use it. **SV**



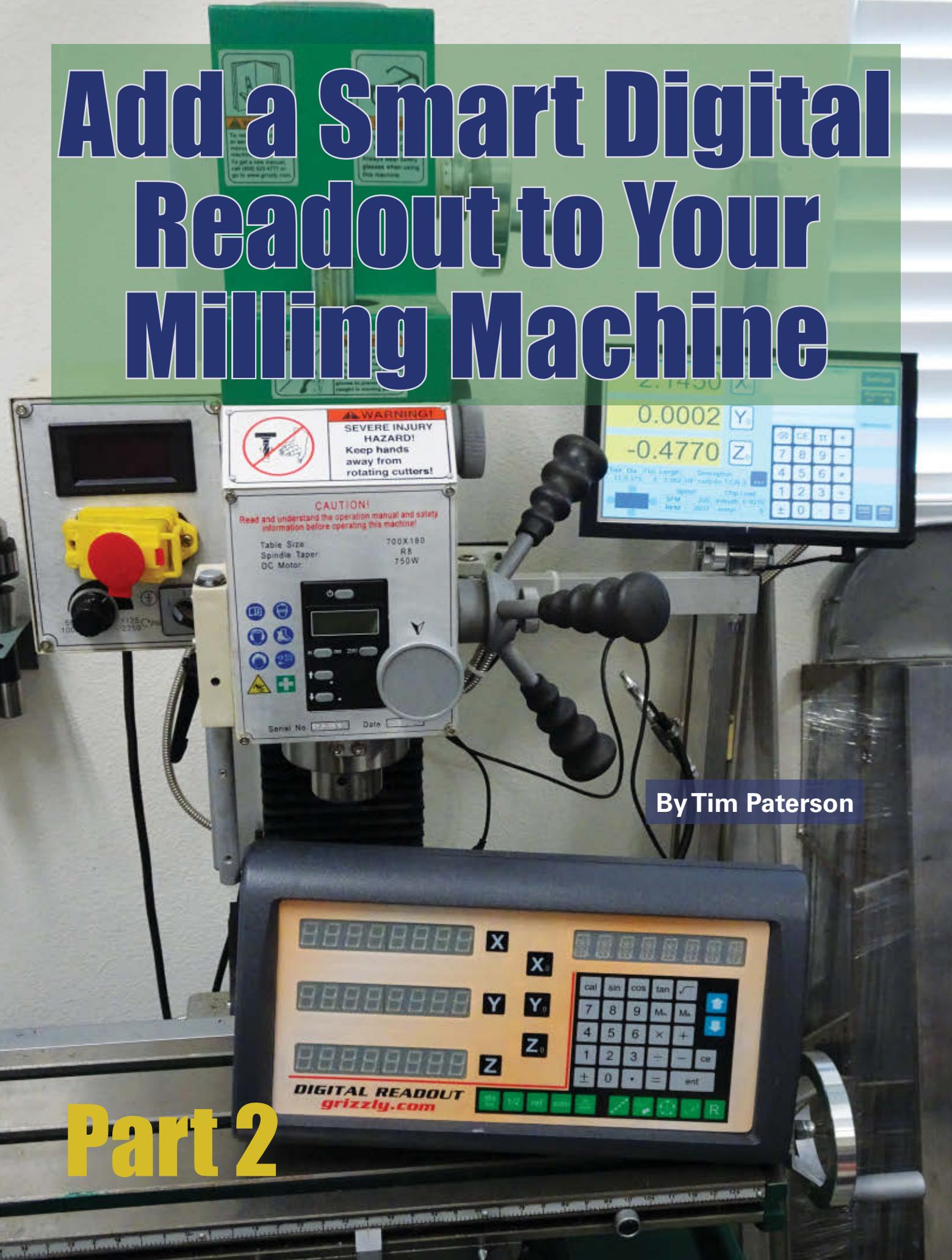
**Figure 11** – MCU PCB attached to display electronics, ready for back plate.

To post comments on this article and find any associated files and/or downloads, go to [www.servomagazine.com/magazine/issue/2020/05](http://www.servomagazine.com/magazine/issue/2020/05).

# Add a Smart Digital Readout to Your Milling Machine

By Tim Paterson

## Part 2



Last time, I described my home-brewed touchscreen digital readout (DRO) for my manual milling machine. This time, I'll cover the details you need to build one yourself. Starting from scratch, it might run about \$250 to make a single unit including some one-time costs. A second one would be more like \$160.

**E**verything that went into creating this project is on GitHub (see **Resources**). From source code to PCB (printed circuit board) files to Windows tools I made like ScreenDesigner (with links to their own GitHub repositories). It's all there. To build your own unit just like mine, all you really need is what's in the "Release."

A Release allows you to download just the essential files without cloning the repository or even figuring out what this Git stuff is all about. Just click on "Latest" in the Releases area on the right side of the main GitHub repository screen, then on the link for SmartDro.zip to download it. Unzip it into a folder. Among other things, you should notice the schematic and parts layout diagram there.

You can find a link for the touchscreen LCD that I used in the **Resources**. When ordering it from **BuyDisplay.com**, there are several options that must be specified. You need 5V operation, 6800 parallel interface, resistive touch panel, and pin header connections. You also need the 128 MB serial Flash chip they offer. It comes separately and you'll need to install it yourself on the display board at U4 (it's a big SMD that's easy to solder by hand).

You can generally substitute equivalent parts for those given in the **Parts List**. However, when ordering the MCU, be sure to get part number ATSAMD21J17D-Axx (typically -AU or -AUT). When I don't give a distributor part number, it's because I used parts I had on hand.

I created the schematic and PCB layout in Autodesk EAGLE which is free to use for the hobbyist. I use OSH Park for PCB fabrication; for this design, you get three boards for \$40. The project is shared on OSH Park, so you just need to find it on their site using the link in the **Resources**.

You also need a set of the tiny pinout adapter PCBs. If you have the same pinout as I do, you can order it directly from its shared project (again in **Resources**).

Otherwise, you'll need to update the schematic and PCB layout with

EAGLE and upload your own version.

A few years ago, I found a couple of SparkFun tutorials on solder paste stenciling and using a skillet for reflow soldering (see **Resources**). This was prompted by a project that used a gyro chip which was only available in a leadless package that was very hard to hand-solder. Now, I've completely converted to using stencils and a skillet for solder reflow for all my designs.

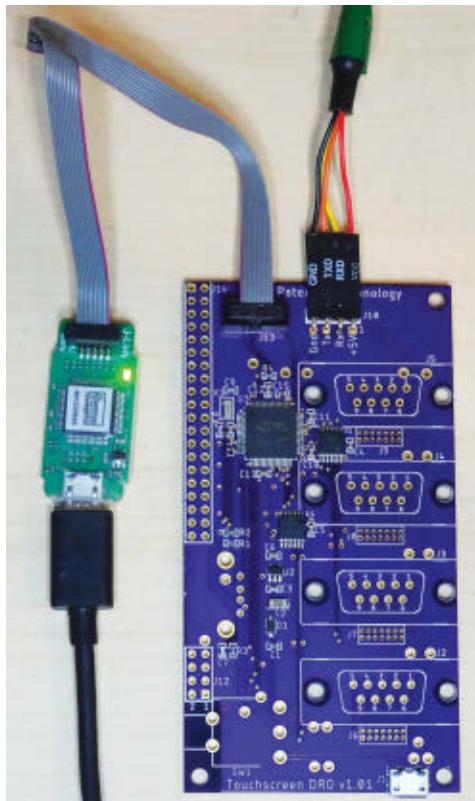
After placing an order at OSH Park, they provide a link to send the design directly to OSH Stencils. I pay the extra to get stainless steel; for this design, it costs about \$20. Still, there aren't many components on this board, and it can be hand-soldered. You just need a steady hand, a fine-point tip, and some magnification for the 0.5 mm lead pitch on the MCU.

After all the SMD components are assembled, the programming connector J13 and serial connector J10 should be added. This is the first chance to check basic functionality but some special equipment is required, starting with a device programmer. I've been using Atmel-ICE for several years, but recently tried the Segger J-Link EDU Mini. For around \$20, it seems to do all the same things, except it's limited to 3.3V systems.

Besides the programmer, you also need a TTL-level USB-to-serial converter. I have one from SparkFun I really like that came with a nice long cable and individual pin connectors on the end; it was only \$8 at the time (see the Recommended Equipment sidebar). The PCB markings on J10 match the markings on this cable.

For example, the cable connector labeled "TXD" connects to the PCB pin labeled "Tx." (This means the PCB could be considered labeled backwards, as its "Tx" pin receives serial data.) You power the PCB by connecting the VCC connector to the +5V pin. Or, you can install the power switch and power it from the micro-B USB connector.

**Figure 1** shows the programmer



**Figure 1** – Setup for initial programming, before most through-hole components have been assembled.

and serial cable connected.

Before you can “burn” the firmware into the MCU Flash memory, you need a terminal program running on your computer for the USB-to-serial converter. I’ve written my own for Windows and included it in the Release as ComTest.exe. Whatever you use, it needs to be set for 500,000 baud.

The software for the Segger programmer comes with two memory Flashing programs called J-Flash and J-Flash Lite. Either one is easy enough to use without reading the manual. You can also program the MCU Flash using Microchip Studio, but that’s a big download if you don’t have it already. Just choose the device (Microchip ATSAMD21J17) and select the DRO.hex file from the Release.

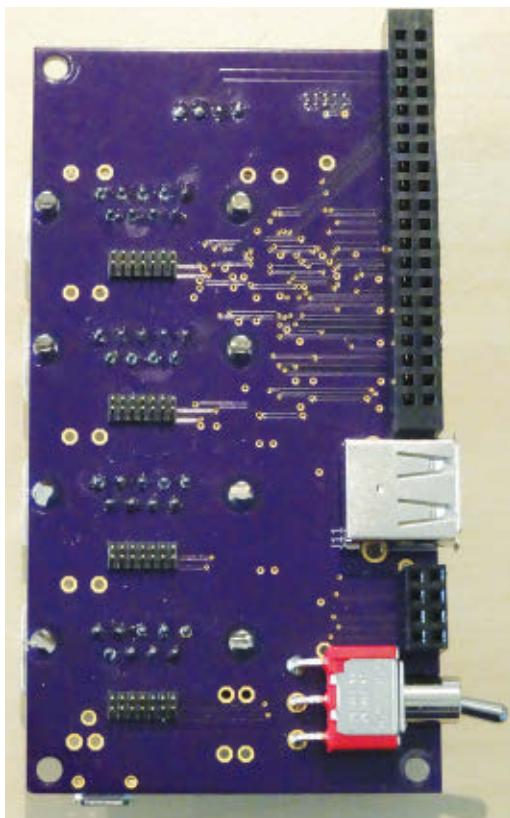
If everything goes right, the terminal program will spit out “DRO version x” to tell you the MCU is up and running. Typing the “x” key into the terminal program should cause the MCU to reset, reprinting the message.

Once this is working, it’s time to power down and complete the assembly of all the through-hole components. Several components outlined on the top side of the board are actually placed on the bottom. Look at **Figure 2** to make sure you mount everything on the correct side of the board.

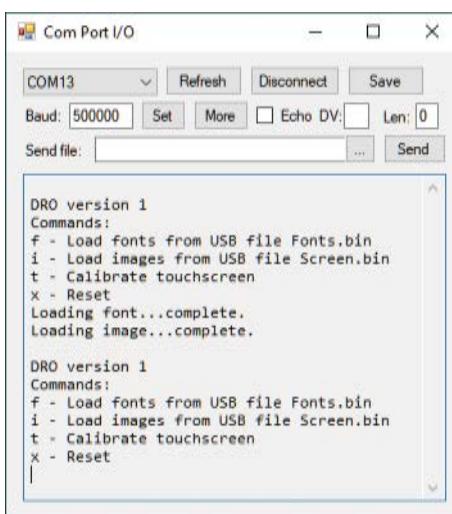
## Final Programming & Calibration

With PCB assembly complete, the board should be plugged into the display module for the final steps. As the display can take 500 mA by itself (and my three position sensors combine for another 200 mA), be sure to use a strong USB power brick that puts out a full 5V under load.

The remaining steps are to program the display’s serial Flash memory (you installed that, right?) with the images and fonts, and to calibrate the touchscreen. These steps require the serial connection again, but once they’re done,



**Figure 2** – Quite a few of the through-hole components mount on the back side of the PCB.



**Figure 3** – Screenshot showing using the serial console to load fonts and graphics into the display Flash.

it won’t be needed anymore. Don’t connect the +5V pin on the serial connector when the unit is powered from the USB port.

The images and fonts are programmed by copying the files Screen.bin and Fonts.bin from the GitHub Release to the root of a USB Flash drive. Plug this drive into the USB port on the PCB. Now at the serial console, just type “f” and you should get the response “Loading font...” which will finish in just a second. Next, you type “i” with the response “Loading image...” and then wait for a bit.

There’s over 4 MB of images and it takes around 40 seconds to program them into the serial Flash. Once the fonts and images are programmed, you can type the “x” to reset the MCU and the main screen should appear. **Figure 3** shows a screenshot of the serial console.

The touchscreen won’t work yet because it needs to be calibrated. Type “t” at the serial console and the touch calibration screen appears.

This simple image is generated by the MCU firmware, so it doesn’t rely on the serial Flash to be programmed.

To get a good calibration, use something pointy and soft-ish (plastic) to apply a medium pressure at the crosshairs on the screen. After touching the three locations, you can touch anywhere on the screen and get a cursor that will show you where it thinks you’re touching.

Just landing anywhere within the cursor is decent calibration. You can test the effect of pressure on the calibration; you need to press hard enough for the cursor to stop moving.

## Buttoning Up

The Release includes STL files for the two plastic bezel pieces that can be 3D printed. The back plate is 1/8" aluminum; I’ve included a fully dimensioned drawing in the Release. I didn’t include a drawing of my mounting bracket, as it’s dependent on what you’re mounting it to. I have included the full Fusion 360 model which has the bracket and even the CAM I used to machine



**Figure 4** – Stack standoffs to tighten them in the corners. A wrench or pliers may be needed to separate them.

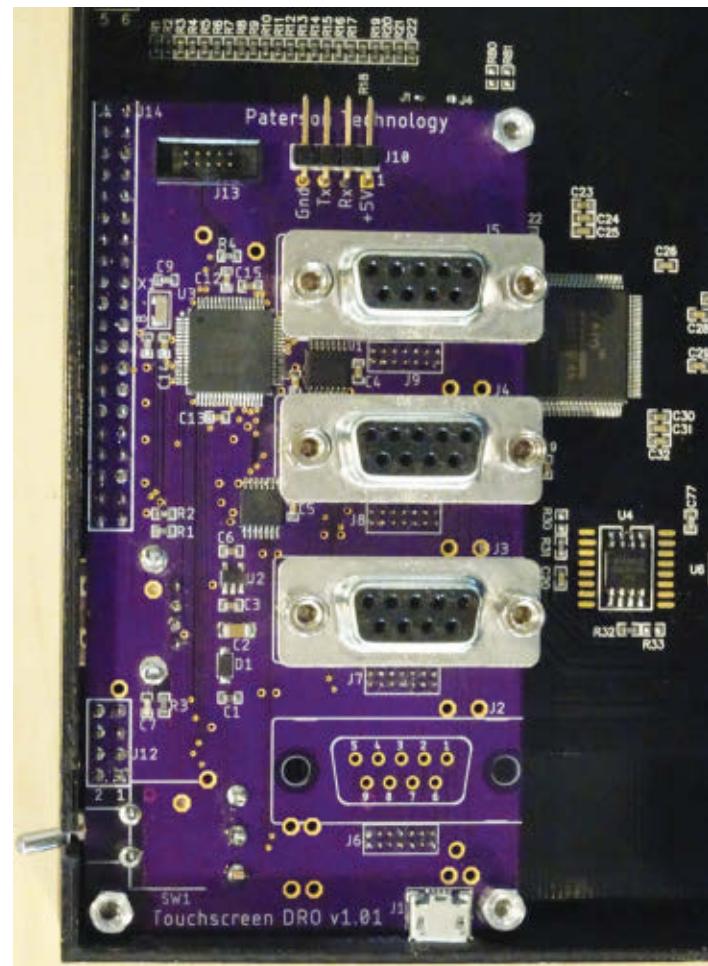
it. You can build it the same way or modify the design any way you want.

When assembling the screen into the plastic bezel, the standoffs fit very tight into the corners. **Figure 4** shows how I stacked standoffs to make them accessible. The corner where the MCU PCB mounts uses a 7/16" standoff, while the rest are 3/4".

Before installing the MCU PCB, put 1/4" standoffs in the holes above and below the D connectors. These should be held on with nylon-insert lock nuts to ensure they don't get loose.

Pushing the PCB onto its connectors is slightly tricky. The power switch handle rides down a slot in the bezel, but the plastic still needs to be bowed out a bit for the USB-A connector. Once you have the pins lined up, you're home free.

Secure the corner with a 1/4" standoff, which makes the height to the back panel 7/16" + 1/16" (PCB) + 1/4" = 3/4" — the same as the other mounting points. **Figure 5** shows it in place.



**Figure 5** – The MCU PCB is nestled in its corner, fastened with one standoff. Note U4 on the main board has the 16 MB Flash memory installed.

When you're ready for final assembly of the back panel, it's important to affix it with all 11 screws. The two above and below the D connectors are critical support when you push the position sensor connectors on. The four that go into the bezel keep the two halves from flopping around at the seams.

## Putting It to Work

The moment of truth is connecting your position sensors and seeing the readouts change when you crank the handwheels. The order of the connectors on the back is the same as the displays on the front, X axis on top. The sign (positive or negative) for a given direction of travel will be

### Recommended Equipment

Device	SparkFun Part No.	Digi-Key Part No.
Segger J-Link EDU Mini Programmer	PGM-15345	899-1061-ND
USB-to-TTL Serial Cable	CAB-17831	1568-CAB-17831-ND

**Figure 6** – Initial Setup screen. Sensor resolution and direction are the most critical.

random.

On my mill, the handwheels are marked so X increases as the table moves left, Y increases as the table moves away from me, and Z increases as the cutter is lowered. The original Grizzly DRO used the same sign convention. With a new setup, odds are some of these will be backwards, which we can correct in the Setup menu.

Touch the Setup button in the upper right corner; you'll get the screen shown in **Figure 6**. Now you can simply touch Invert Direction for an axis to correct any that were moving backward. You can also disable an axis altogether; for example, if you don't have a Z position sensor. If your sensors don't have the typical five micron resolution, you can change it here.

This would be a good time to talk about how to enter and use values. Tapping digits on the keypad enters a number into the calculator. Then, tapping on any field that uses a number copies it there.

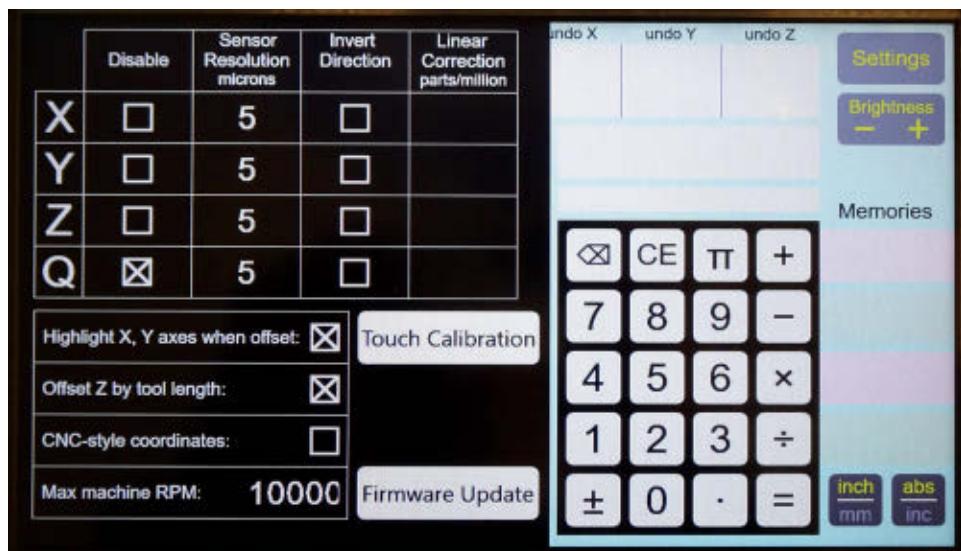
If you have two micron sensors, just tap "2," then tap the resolution box for each axis to change them all. This applies to any numeric field on any screen. You could enter a number, then tap one of the memory fields along the right side, and the value will be kept and displayed there.

To move a value back to the calculator, the calculator display must be blank. You get that with the CE (Clear Entry) key. So, tapping CE and a memory or CE and an axis resolution field will copy either of them back to let you use it in a calculation. The same rules apply to the axis displays and tool parameters on the main screen.

Some fields like the memories or the tool parameters are "blank on zero." If you're done with a value in a memory and don't want to see it there anymore, enter 0 in the calculator and tap the memory. Instead of showing the value zero, the display for that field is blanked. Blank on zero does not apply to some fields, like axis resolution or axis position.

Most of the remaining settings are fairly self-explanatory. I do want to mention the "CNC-style coordinates." On a CNC mill, the direction of both the Y and Z axes is reversed. This makes the coordinates match the normal orientation in a CAD program, where increasing Z is up.

You can see there is an option to select this



coordinate system.

What's important here is that checking the box does NOT reverse Y and Z for you. You still have to do that with Invert Direction. This box is for offset calculations (for example, whether to add or subtract the tool length to / from Z).

When you're done with the Settings menu, tap the Settings button in the upper right again. This will save any changes to a Flash memory dedicated to holding the parameters, and then bring you back to the main screen. If things are set up right, turning the handwheels will change the readouts in the expected directions.

This would be a good time to experiment with zeroing the axes, setting them to a specific value, and copying their current reading back into the calculator. You may find there's an easy mistake to make: You intended to copy an

## Resources

Project on GitHub

<https://github.com/TimPaterson/TouchscreenDigitalReadout>

BuyDisplay LCD

<https://www.buydisplay.com/serial-spi-i2c-10-1-inch-tft-lcd-module-dislay-w-ra8876-optl-touch-panel>

OSH Park - Main PCB

[https://oshpark.com/shared\\_projects/jTW1CESI](https://oshpark.com/shared_projects/jTW1CESI)

OSH Park - Pinout Adapter

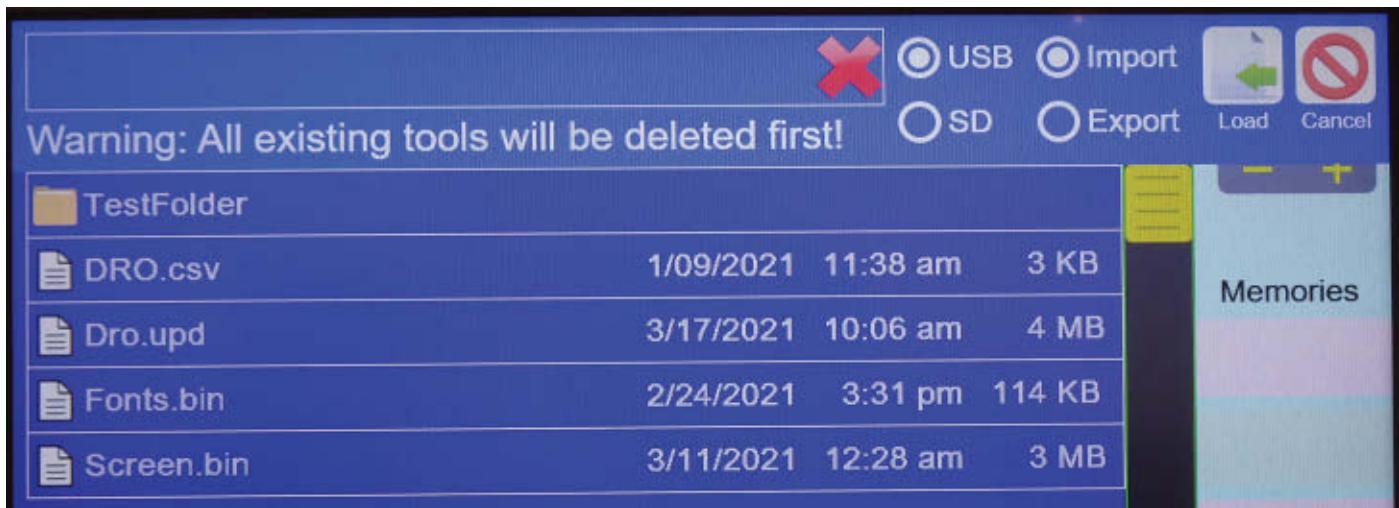
[https://oshpark.com/shared\\_projects/7Whg99iS](https://oshpark.com/shared_projects/7Whg99iS)

SparkFunTutorial

<https://www.sparkfun.com/tutorials/59#Hot%20Plate%20Reflowing>

Library Manager on GitHub

<https://github.com/TimPaterson/CNC-Tool-Library-Manager>



**Figure 7** – Contents of the Flash drive on the USB port of the DRO. The file DRO.csv can be imported to populate the tool library.

axis position to the calculator but didn't tap CE first to clear the calculator display.

The axis display will be set to the calculator value instead of the other way around. Fortunately, there's an easy fix! Tap the Undo button above the calculator for the affected axis and the readout is restored.

Next, give the radius offset feature a try. Enter the diameter of your favorite end mill in the calculator and tap "Dia." in the tool parameters.

Then, just below that, tap which side of the workpiece you're cutting, and note the change in the affected axis.

## Tool Libraries

If you don't measure your tool lengths and don't want the DRO to help calculate feeds and speeds (for which it needs a flute count), then you might prefer to just manually enter the diameter each time you change tools. If that's the case, go ahead and skip this section!

If you want the DRO to keep a permanent tool library for you, press the "... " button to the right of the tool parameters (refer to Part 1 of this article). To manually enter a tool, it must have a tool number (1-999) and any one other field set. You set the numeric fields in the usual way touching the entry row at the top with the value in the calculator.

To enter a 1/32" end mill, for example, you could enter "1 ÷ 32 =" then tap the Dia. field. The field display is truncated to three decimal places, but you can see the full value is still there by tapping CE and the field to copy it back.

(Internally, the value is stored in double-precision floating point.)

To enter a description, tap anywhere in the description field and an onscreen keyboard appears. Be sure to tap the Done button on the keyboard itself – not the one in the upper right corner – to save your changes.

Of course, there's an alternative to entering all your tools on the touchscreen: Import them from a USB Flash drive. By tapping the Import/Export button in the upper right corner of the tool library screen, you bring up a file browser. In **Figure 7**, it's displaying the contents of a Flash drive of mine.

The file DRO.csv has my current tool library in a "comma-separated values" format. The file was actually written in Microsoft Excel, and you can see a portion of its contents in **Figure 8**. By tapping the file

name in the list and then the Load button in the upper right, all 68 of my tools are instantly loaded.

So, where did this file come from? A few years ago, I created an Excel application using macros that translates between the different file formats of my tool libraries. Fusion 360 has the master library, which I can export and load into Excel. My macros then churn out versions that can be imported by my Tormach CNC mill, G-Wizard program, and now the Smart DRO.

I've shared my Tool Library Manager on GitHub (see **Resources**), so anyone can use it. It's designed to be easily adapted to other export formats and not be limited to just the machines and programs I use.

If you don't have a need to share tool libraries, Import/

DRO.csv - Notepad	
File	Edit
Tool,Diameter,Length,Flutes,Description	
1,0,4.557,,3D Tester	
2,0,03125,2.56,2,"1/32"" carbide TiCN 2 flute end mill"	
3,0,0625,2.573,2,"1/16"" carbide TiCN 2 flute end mill"	
4,0,125,2.877,3,"1/8"" carbide TiCN 3 flute end mill"	
5,0,75,4.6915,3,3/4 HSS TiN 4-flute end mill	
6,0,4375,4.308,4,7/16 x 2 HSS TiN 4-flute end mill	
7,0,5,3.583,4,"1/2"" carbide TiCN 4 flute end mill"	
8,0,03125,2.315,2,"1/32" carbide end mill	
Ln 1, Col 1	100% Windows (CRLF) UTF-8

**Figure 8** – A sample of the contents of a tool library file. It was exported to Microsoft Excel.

Qty	Reference	Description	Package	Digi-Key Part No.
1	C1	10 $\mu$ F Capacitor	0603	
1	C2	100 $\mu$ F Capacitor	1206	1276-1782-1-ND
3	C3, C6, C15	1 $\mu$ F Capacitor	0603	
7	C4, C5, C10, C11, C12, C13, C14	0.1 $\mu$ F Capacitor	0603	
1	C7	4.7 nF Capacitor	0603	
2	C8, C9	10 pF Capacitor	0603	
1	D1	MBR120VLSFT or MBR120LSFT Diode	SOD123F	
1	J1	USB microB Connector		609-4616-1-ND
4	J2, J3, J4, J5	DE-9F Connector		L77SDEH09SOL2RM8-ND
4	J6, J7, J8, J9	14-pin 50-mil Header		2057-HPH2-A-14-UA-ND
4	J6, J7, J8, J9	Optional Socket		609-3756-ND
1	J10	Four-pin 100-mil Header	Right Angle	1849-PR20204HBNN-ND
1	J11	USB-A Connector		ED2989-ND
1	J12	Eight-pin 100-mil Socket		S7107-ND
1	J13	10-pin 50-mil Header		1175-1627-ND
1	J14	40-pin 100-mil Socket		S7123-ND
1	R1	33K Resistor	0603	
1	R2	11K Resistor	0603	
1	R3	1M Resistor	0603	
1	R4	1K Resistor	0603	
1	SW1	Toggle Switch		EG2362-ND
2	U1, U4	74LCX14 Logic IC	TSSOP14	MC74LCX14DTR2GOSCT-ND
1	U2	MIC5504-3.3 Voltage Regulator	SOT23-5	576-4764-1-ND
1	U3	SAMD21J17D MCU	TQFP64	ATSAMD21J17D-AU-ND
1	X1	32.768 kHz Crystal	3.2x1.5 mm SMD	2195-CM7V-T1A-32.768KHZ-7PF-10PPM-TA-QCCT-ND
4		4-40 x 3/4" Hex Standoff		1772-1003-ND
3		4-40 x 1/4" Hex Standoff		1772-1230-ND
1		4-40 x 7/16" Hex Standoff		1772-1001-ND
11		4-40 x 1/4" Screw		
2		4-40 Nylon Insert Lock Nut		

From [BuyDisplay.com](http://BuyDisplay.com)

1 ER-TFTM101-1 LCD w/options: pin header 6800 parallel; 5V; resistive touch; pin header microSD.

1 Winbond W25Q128JV serial Flash (install on LCD at U4).

Export can just be viewed as a way to back up the tool library you manually entered into the DRO. The unit has a microSD card slot, and back in **Figure 7** you can see there's a choice for "SD" instead of "USB." It's not very convenient to take cards in and out of the slot, but you can just leave one in there and use it as a backup.

When you make changes to your tool list, you would just go ahead and export it to the microSD card.

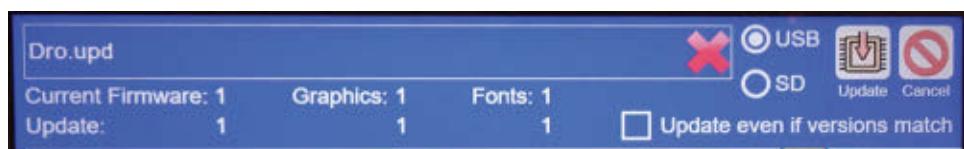
Note that when you export a file, the file system needs a date and time to timestamp it. Selecting Export will cause a Set button to appear to set a clock in the DRO. The clock will remain set until the power is switched off.

## Firmware Update

Back on the Settings screen in **Figure 6**, there's a large



**Figure 9** – Browsing for a firmware update file. It's the one with the ".upd" extension.



**Figure 10** – The firmware update file has been opened and verified. You can compare the current DRO versions with those in the file.

white button labeled Firmware Update. This will bring up a file browser that's similar to the one used for Import/Export shown in **Figure 9**. The build process for the DRO firmware automatically creates an update file with a ".upd" extension, which you see in the file list. This file combines all three elements that were programmed separately when the unit was first programmed: firmware, fonts, and images. Each Release on GitHub includes the corresponding update file as a separate download.

When you select the file and tap the "Inspect" button, the DRO verifies it's a valid update file and displays the

versions of the components it contains as shown in **Figure 10**. In this case, all the components in the update file match the current firmware in the unit.

If you were to tap what is now the "Update" button, it would update the MCU firmware from the file, but the graphics and fonts would not be updated because they're already correct.

You can force the graphics and fonts to update anyway by checking the box, but this should only be needed if a previous update had gone wrong.

## And We're Done!

I enjoyed working on this project, learning lots of new stuff, and maybe – most of all – declaring it finished! I also like to help others take advantage of what I've learned, which is why I'm writing about it and sharing it all on GitHub.

The GitHub project includes Discussions and Issues sections where you can make comments and ask questions. I'll help if I can. **SV**

To post comments on this article and find any associated files and/or downloads, go to [www.servomagazine.com/magazine/issue/2020/06](http://www.servomagazine.com/magazine/issue/2020/06).

## Compact. User Friendly. Highly Capable.

CNC MILLS | CNC LATHES | CNC PLASMA TABLES | CNC ROUTERS | AUTOMATIC FEED BANDSAWS



We make CNC machines **people can buy**. They **fit in places** other CNCs can't.

Those who are new to CNC love Tormach equipment because our very own PathPilot control is **easier to learn and use** because it's **more intuitive**.

Experienced machinists appreciate Tormach machines because they **reliably turn out parts with any CAD/CAM**.

Build yours at [tormach.com/loop](http://tormach.com/loop).



**TORMACH®**

Check out the  
**SERVO** Junkbox  
on Page 74!

