KDØRC

**TeensyMaestro**

Builder’s Guide

Len Koppl, KDØRC

2/14/2023



1. Introduction

The TeensyMaestro is a Teensy (Arduino work-alike) based set of controls for the FlexRadio 6000 series transceivers. It provides knobs, buttons, display, keyer and microphone switch. It is designed to augment SmartSDR (SSDR), not replace it. There are many functions that the TeensyMaestro cannot perform that still need to be done using SSDR or a real Maestro.

The TeensyMaestro uses a USB cable for power and programming, and an Ethernet cable to connect with the radio. All communication with the radio is via TCP/IP over the Ethernet cable, not the USB cable.

The primary design goal was to provide the most used controls in the form of knobs and buttons while retaining all the power and convenience of the computer-based SSDR. It uses the extensive TCP/IP Application Programming Interface (API) provided by FlexRadio. It also leverages the Flex Arduino library built by Enzo, IW7DMH. Without this API and library, the TeensyMaestro would not have been built.

While the TeensyMaestro has a color touch screen, it does not display a panadapter or waterfall. It does show basic slice information in a similar way as the “flag” does in SSDR. Similarly, the touch screen is a rudimentary resistive type, so it does not have cell phone like capabilities but provides some simple functions. There are no functions that rely solely on the touch screen; the physical controls cover all touch screen functions.

1. Getting Started

This project primarily consists of gathering the components and soldering or plugging them together with jumper wires then drilling and cutting the front panel. In some ways, the electrical connections are the easy part; for most of us, it is the front panel that presents the greater challenge.

This is not an advanced project, but it is probably not for first-time builders either. There are no SMD parts to solder, but there is a lot of point-to-point wiring that can be a bit tedious to connect. With reasonable care and plenty of patience, this project has a high likelihood of success (quite a few have already been built as of this writing). Note that you do not have to do any programming or compile anything, nor do you need to have or understand the Arduino Integrated Development Environment (IDE).

Included in the [GitHub](https://github.com/KD0RC/Teensy-Maestro-for-Flex-6000-radios) files is a little [loader](https://github.com/KD0RC/Teensy-Maestro-for-Flex-6000-radios/blob/master/teensy.exe) provided by PJRC that is used to load the Teensy board. It is very easy to use, and complete instructions are given later on how to use it to load the firmware.

I will often refer to either the software or firmware and use these terms interchangeably. They mean exactly the same thing in the context of this project.

As you begin the project, take a look at the Flex Community forum, [TeensyMaestro thread](https://community.flexradio.com/discussion/8023866/teensymaestro/p1). It will give you ideas of what others have done with theirs, and provides a centralized place for questions. I prefer this to personal email so that others can benefit from your questions. I generally check the forum multiple times a day (unless I am skiing, fishing or sleeping on the couch…).

If you have trouble with something that is not getting solved on the Community, I am very willing to set up a Skype or Zoom call to help you out; just let me know. I have done this with a couple of hams now to great effect. I can often spot something visually that was missed in the written exchanges.

* 1. Bill Of Materials
     1. **Minimum Requirements**

The firmware will run with no controls or display if you so desire. It still has some functionality, even in this cut-down configuration. It is able to post fixed spots to SmartSDR (I use this to show nets and repeaters that I am interested in) and can show out-of-band operation based on license class and mode (configurable for any country). This is all driven from the MMConfig.ini file on the SD card. More details about these features are available in the User Manual.

If you want to dabble, but not fully commit to this project, you can copy the sample MMConfig.ini file to an SD card, load the software to a Teensy board, plug it in to your LAN and it will work. This is not a bad way to start; it gives you the peace of mind that you have a working Teensy without soldering anything except for the Ethernet header, one capacitor and the jack.

Several builders have built their TeensyMaestros with less that the full complement of functions. Some do not operate CW and elected to leave off the message buttons, one multiplexer and key jacks. Mike, M0MDS very rarely uses two slices, and so only built the Slice A controls. This saved cost, complexity and desk space. How cool does this look?



* + 1. **Teensy 4.1 Board with Ethernet**

Most parts are not critical. The Teensy board, however, ***is*** critical. There are fake Teensy boards available that do not meet specs, and are not recommended. The original Teensy 4.1 boards from [PJRC](https://www.pjrc.com/store/teensy41.html) are only $35-ish USD including the [Ethernet connector](https://www.pjrc.com/store/ethernet_kit.html) and cable (sold separately). This is a good deal for such a powerful, compact board. Make sure that you get the Teensy board that includes the Ethernet chip. Due to parts shortages during the Covid pandemic, PJRC now produces some Teensy 4.1 boards without Ethernet, so be careful what you order.

* + 1. **Encoders**

The impetus for this project was the unit built by Enzo, IW7DMH which uses expensive (but really good) Leo Bodnar encoders. I followed suit and used them in my TeensyMaestro. They are very high quality, are reliable and have a great feel to them. Most of the hams who have built one of these used the much cheaper Adafruit or Bourns encoders. While they don’t feel quite as good as the Leo Bodnar units, they are certainly high enough quality to be worthy of consideration. The BOM spreadsheet has both types specified so that you can choose for yourself.

The Leo Bodnar encoders are 1:1 (i.e., one detent click per value change) while the others are generally 4:1 (one detent click moves the value ahead or back by a count of 4). I will discuss how to manage this in the MMConfig.ini file later in this document (spoiler alert – it is a very easy change to make). Note that each encoder has a line in the config file, so you can mix and match the encoders any way you like.

* + 1. **Buttons**

I found some dime-a-dozen pushbuttons on Amazon. Their reliability was horrible and I wound up replacing them with slightly better buttons (approximately $1.00 USD each). The cheap buttons give very unreliable results, making the TeensyMaestro frustrating to use. The medium value buttons work very reliably and make the unit a joy to use. There is a value in the config file that can be used to change the debounce value for the buttons if you have problems with your switches double-striking.

* + 1. **Multiplexers**

If you don’t want the CW message buttons, you can leave the second multiplexer out. The firmware will work either way. Other builders have used knock-off boards successfully, but I would rather pay the extra couple of dollars and support SparkFun’s engineering and development efforts (not to mention getting better quality).

* + 1. **Touch Display**

The display is another critical component. The software expects the exact model specified in the BOM. That being said, you do not have to include the touch controller. There are no functions that use the touch controller that aren’t available using the buttons or menus.

The original touch controller board (STMPE 610) specified in the BOM was discontinued by Adafruit. The replacement unit (TSC 2007) is now in the BOM and is the one that should be purchased. If you already have the older one, it can still be used. The code figures out which one you have and behaves accordingly.

* + 1. **Wiring**

I originally built my TeensyMaestro with .1 inch header pins and jumper wires. This was a great way to go as I was experimenting and designing the circuit and associated software. However, once I had the unit packaged up in a case and started hauling it around to Field Day and friend’s houses to show it off, I started having problems with the display going into the “white screen of death”.

A restart cured the problem – until the next time. As this got more and more frequent, I realized that the cheesy jumpers were not staying put and/or were oxidizing. The display is particularly prone to problems if the wires are loose or the contacts are dirty.

The solution was to solder things down, especially the display lines. If you experiment a bit with jumpers, be sure to solder the wires at some point or you will likely have a time-consuming task ahead… Soldering as you go is ***much*** easier that going back once the box is all built.

Wire quality plays into the equation as well. Some of the ultra-cheap wiring is made of who-knows-what. The percentage of copper is likely very low or non-existent, resulting in very brittle connections and poor solder-wetting. The display is the place most affected by this. Failure of this wire is directly proportional to the urgency with which you are operating. Got rare DX? Wires will break…

Dave, W4WKU (builder of TeensyMaestro #2) used some cheap jumpers and had to replace the wiring to get a stable TeensyMaestro. I used the jumpers sold by Adafruit and so far, have had no problems with them.

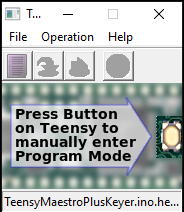
* 1. First Steps
     1. **Connections and power**

Before wiring anything (except for the Ethernet components), power up the Flex and start SmartSDR. Plug the Teensy ethernet cable into the same network (LAN) as the Flex. If this is a DHCP-served router or modem (most are), the TeensyMaestro will obtain an IP address for connection to the radio. If you use a network switch to connect the computer (running SSDR), the radio and the TeensyMaestro together without benefit of DHCP (i.e. no modem/router), then you may or may not need to change settings on the SD card (see section 6.3 in the User Manual).

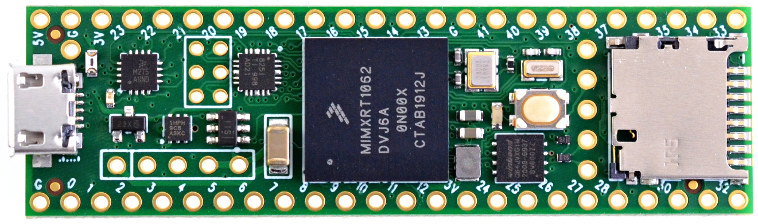
Turn on the TeensyMaestro by plugging one end of a USB cable into the back of the TeensyMaestro and the other end into any USB port on your computer that provides 5 VDC. The TeensyMaestro does work without SmartSDR running and connected to the same Flex radio, but generally, that is not the preferred way to operate. The rest of setup will assume that the Flex is powered up (green light on steadily), SmartSDR is running and connected and that the TeensyMaestro is connected to the same network as the radio. It is also assumed that the TeensyMaestro is powered up *after* the radio has a solid green light and SmartSDR is running and connected to the radio.

* + 1. **Loading the Software**

Before using the TeensyMaestro for the first time, the software must be loaded. Teensy.exe is used to load the .hex file (executable object code). Start Teensy.exe and plug the Teensy into an available USB port:



Click *File*, *Open HEX File* and select **TeensyMaestroPlusKeyerV00x.xxx.ino.TEENSY41.hex** from whatever location you have it saved (use the latest version that is available on GitHub). Press the Program Load button on top of the Teensy board and the software will load and start.



Use this SD Card slot

USB

Program Load button

If the software is working, you should see a small pop-up window in the lower right-hand corner of SmartSDR indicating a successful connection from the IP address of the Teensy.

* + 1. **SD Card**

The Teensy board has a micro-SD card slot that can be used to hold configuration information. Edit MMConfig.ini to fit your needs and copy it to a FAT32 formatted micro-SD card (see section 6 of the User Manual for information on editing the file). A very small card can be used, as MMConfig.ini is the only file needed.

The display has an SD card slot on it, but it is not used in this application. Be sure to put the SD Card in the slot on the Teensy board itself, not in the one on the display.

1. Panel Layout
   1. Front Panel
      1. **Make it Your Own**

The layout that I used meets my needs and fits my style of operation. If it fits yours, great! If not, don’t hesitate to arrange things so that they make sense to you. My first prototypes were built into cardboard boxes. This allowed me to move things around quickly and easily without the time and expense of cutting multiple aluminum front panels.



I used this layout (above) for a while before deciding (on a whim…) to include the color touch display. I constantly grabbed the wrong controls which told me that my layout was not ergonomically sound for me. I also realized that I would run out of General Purpose I/O (GPIO) pins long before I had all of the annunciator LEDs and additional buttons wired in.



Even this nearly-final layout (see above, right) did not work out all that well. I thought the layout of the Hi and Lo cut controls stacked vertically made a lot of sense, until I kept grabbing the AGC-T knob for Lo cut and the Lo cut when I wanted Hi cut. Everything else seemed about right. That last cardboard prototype was a lifesaver and got me where I wanted to be. I used it this way for quite a while before committing to cutting and drilling the aluminum panel.

As you start the wiring process (section 4, below), keep the front panel layout in mind so that you have adequate wire lengths for the controls. If you plan on using my layout, then you might want to prepare the front panel, and mount the controls first.

If you think that you might want your own layout, I suggest using a cardboard box as a prototype to test your ideas before cutting and drilling a real panel.

* + 1. **Panel Graphics**

There are several ways to get a good-looking front panel. I landed on a laser-printed, plain paper graphic under a clear ABS plastic sheet from ACE Hardware. This was cheap and looks reasonably good. The Word document for the front panel is posted on the GitHub site. Cutting the ABS sheet without scratching it was a challenge, to say the least.

If you look at the pictures that I have been using in my documentation, there are some very obvious scratches. In normal room lighting, they are not particularly noticeable. Under the harsh light used to take the picture, they stand right out.

Front Panel Express makes custom aluminum panels with engraved, color-filled labeling that look fantastic, but are fairly expensive. If you go this route, print some prototypes from paper first so that you can be certain that everything fits and that the knobs do not obscure the engraved lettering.

1. Wiring
   1. General

As you wire in the various components, you should test as you go. Some builders wired everything, then powered it on for the first time. This did not generally go well. Sorting out which connection was causing an issue was a nightmare (you know that there will be some issues, right?).

The schematic is a bit misleading as to which way to connect the encoders, so you need to wire in an encoder, then test it for proper rotation. It is an easy matter to swap wires if the encoder you just installed operates backwards. Once all the controls are soldered in, it is much more difficult to find and reverse the wiring.

Further, if you are off by a pin on the Teensy board, you will likely propagate that error to all subsequent controls. Testing as you go lets you know exactly which component wiring caused the issue.

* 1. Teensy Board

While the controls can be soldered directly to the Teensy board, I recommend a proto board with sockets for the Teensy. All other connections should be soldered to the proto board. The rest of the wiring discussion will assume this method of construction. I used the “Perma-Proto” boards from Adafruit, but any proto board with .1 inch spacing and solder pads should work.

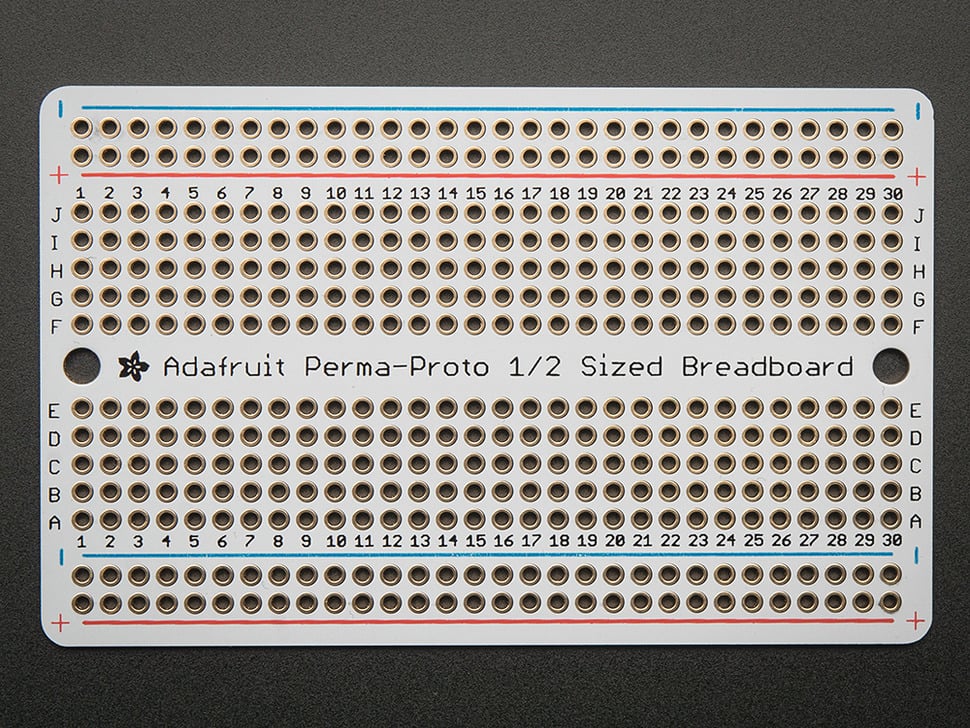
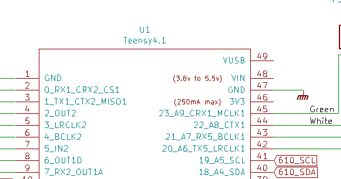
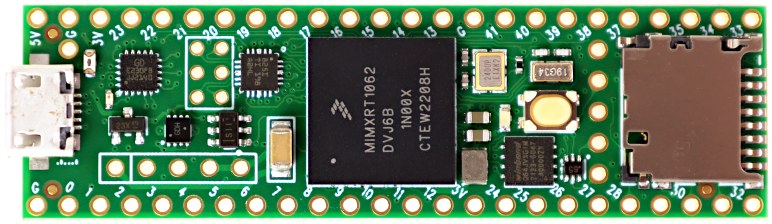


Image courtesy of Adafruit Industries.

The Teensy board schematic component has a confusing set of pin numbering on it. Each pin has an easy-to-read number around the outside edge. These numbers do not relate to anything and are not referenced anywhere.

On the inside of the component are signal names that start with a number. These numbers reflect what is printed on the actual Teensy board. As you make connections from the controls to the Teensy board, be sure to use the proper pin numbers.



Wrong pin numbers

Correct pin numbers

Image courtesy of PJRC.

Before soldering in the Teensy sockets, plan on where they will go on the proto board. I mounted mine so that the SD card would be right at the opening in the case to make removing and editing the card much easier.





SD Card

Mike, M0MDS and Stefan, VA3AR both included an SD card extension cable on their units to allow them to place the main board anywhere without worrying about access to the SD card.

Once the Teensy sockets are soldered in, bring the 3.3 V and GND connections out to the power rails on the proto board. All 3.3 V connections are internally connected and all GND connections are internally connected so don’t worry about cross connecting them. I simply brought the power connections out from the closest side of the Teensy board to their respective power rails. Be careful – the order of the + and – rails might not be what you expect on the left and right sides of the board.

Use an ohm meter to test for shorts from 3.3 V to GND before plugging in the Teensy board. Also test to be sure that the + and – rails are connected correctly before applying power via the USB cable.

If all that is OK, plug in the Teensy board and connect the Ethernet cable. Power it up by plugging in the USB cable. It should connect to the Flex as it did in “First Steps”, above.

* 1. Display

I recommend wiring the display first. With it, you will be able to more easily test the other components as you wire them in.

Start by soldering across the IM2 jumper on the back of the display. This puts the board into SPI mode which is used by the TeensyMaestro. There are two sets of connectors on the display board, one for SPI and one for 8-bit parallel data. We will use only the SPI side of things.

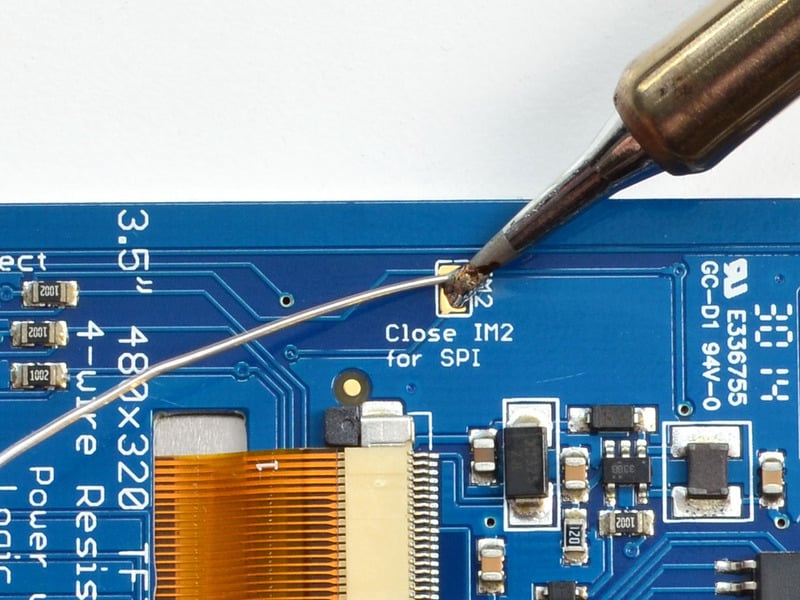
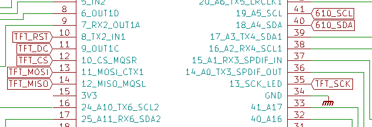


Image courtesy of Adafruit Industries.

The display is designed to be used in either 3.3 V or 5 V systems. It is ***very*** ***important*** to only power the display with 3.3 V, or you will kill the Teensy board. There is a pin labeled 3-5V that should only be connected to 3.3 V in this application. Do not use the 3.3 V out pin.

On the schematic, the display connections have flags that start with “TFT\_”.



The two touch screen controller pins start with “610\_”, even if you have the newer TSC2007 controller.

Display board pins not used by the TeensyMaestro are Card Detect, Card CS, IM0 – IM2, Lite and 3.3V out.

The X-, Y-, X+ and Y+ lines are connected to the corresponding pins on the touch screen controller (if you are using one).

Even with no controls connected, the display should follow SmartSDR. Change the frequency, volume, AGC-T, etc. and the corresponding item on the TeensyMaestro display should show the same values as SmartSDR.

The Serial Peripheral Interface (SPI) bus that the display data travels over operates at HF clock frequencies, so depending on the length of your wiring, quality of your soldering and the phase of the moon, your display will work fine, or show very odd behavior (torn image, mirror image, inverted colors, etc.).

There is a config item in the MMConfig.ini file that will allow you to change the frequency of the SPI clock to something that your display will be happy with. The Adafruit default is 16 MHz, but I have seen up to 40 MHz work. Most seem to be happy in the 16 to 24 MHz range. Pin headers with push-on connectors can exacerbate display issues, so I highly recommend soldering all the display lines.

* 1. Controls and Interfaces
     1. **Encoders**

The VFO encoders specified in the BOM are designed to work at 5 – 24 VDC. Most work just fine at 3.3 V. If you have problems, you can connect the red wire to Vin on the Teensy board to get 5 V. If you make this change, use an oscilloscope to verify that 5 V does not appear on the quadrature signal lines of the encoder. 5 V on a Teensy GPIO pin will instantly kill the Teensy.

At the time I designed this, the VFO encoders were not manufactured with internal pull-up resistors, so encoder power never shows up on the signal lines. I have no idea about other versions or subsequent designs of these encoders, so be sure to test them before using them at 5 V.

The rest of the encoders should be wired in and tested one at a time. It is much easier to fix the rotation direction or other problems of a control that you just wired as opposed to wiring them all in and having to dig through all the wiring to find errors.

Most encoders are 4:1. That is, they move four units for one detent position. This is not a random thing. They are designed that way so that the detent position always lands with the internal encoder switches in the open position and therefore, the circuitry is not loaded down with grounded inputs. In the case of the TeensyMaestro, this is immaterial, so 1:1, 2:1 or 4:1 encoders can be used.

In the MMConfig.ini file change the encoder steps to match the encoders that you are using.

Vol A encoder steps: 4 ; number of encoder steps per detent click

Vol B encoder steps: 4

AGC A encoder steps: 4

AGC B encoder steps: 4

Low A encoder steps: 4

Low B encoder steps: 4

High A encoder steps: 4

High B encoder steps: 4

CW encoder steps: 4

* + 1. **Multiplexers**

Seven of the eight wires that connect the Teensy to the to the two multiplexers are the same. Only the Enable line is connected to different pins on the Teensy board. This means that you can run one ribbon cable to both multiplexers. This will save quite a bit of wire congestion in the finished unit.

The schematic components for the MUX boards are labeled in a very confusing way. On the SparkFun boards, the 16 switch connections are labeled C0 – C15, but the schematic is labeled I0 – I15. The board shows the common signal line as SIG, but the schematic calls it COM. The board shows the enable line as EN, the schematic shows E. All the other lines are labeled the same. This is ridiculous, as the schematic component specifically references the SparkFun board…

* + 1. **Buttons**

Nothing special to say about wiring in the buttons, other than a reminder that the really cheap ones are probably not suitable. I do recommend a .01 uf capacitor across each button. Before I bypassed my buttons and encoders, I could knock the TeensyMaestro off-line by drawing a static arc when I sat down to operate.

I recommend testing each button as it is being wired in for the same reasons as testing the encoders as they are wired in. My personal choice was to wire the encoder, test it for proper rotation, and then wire and test the corresponding switch. That way if you are looking at the wrong switch connection, you will be able to fix it right away. Otherwise, you might be off on all subsequent switches.

* + - 1. **Power Button**

The TeensyMaestro can be powered on and off by unplugging the USB cable, or even (on some computers) by powering off the computer that the USB cable is plugged into.

I find it more convenient to have a button on the back panel to toggle the unit on or off. Bring the ON\_OFF signal from the top of the Teensy board to a button. I use a .1 inch header and jumper for this so that I don’t have to unsolder anything to remove the Teensy board.

By default, it takes a 50 ms long press of the power button to turn on the Teensy, and a 5 second press to turn it off. You can use the config file to set this up to your preference in the MMConfig.ini file (see the TeensyMaestro User Manual for more info). My preference is:

Power Button: Power Fast

This gives me instant on and off, which is convenient for me as I power it on and off a ***lot*** when debugging code. You may prefer the standard Power option which retains the 5 second off requirement. This prevents accidentally shutting it off with a bump of the button.

* + - 1. **Programming Button**

I have thin fingers and can get to my programming button by sliding my pinkie in over the SD card. This will be most inconvenient for most people, so I recommend bringing the PROGRAM signal from the top of the Teensy board to a button using the same technique as the power button. Be sure to place the button where it won’t get accidentally pressed. If you press it by mistake, pull the power connector out then put it back. It will boot up normally.

* + 1. **Touch Screen Controller**

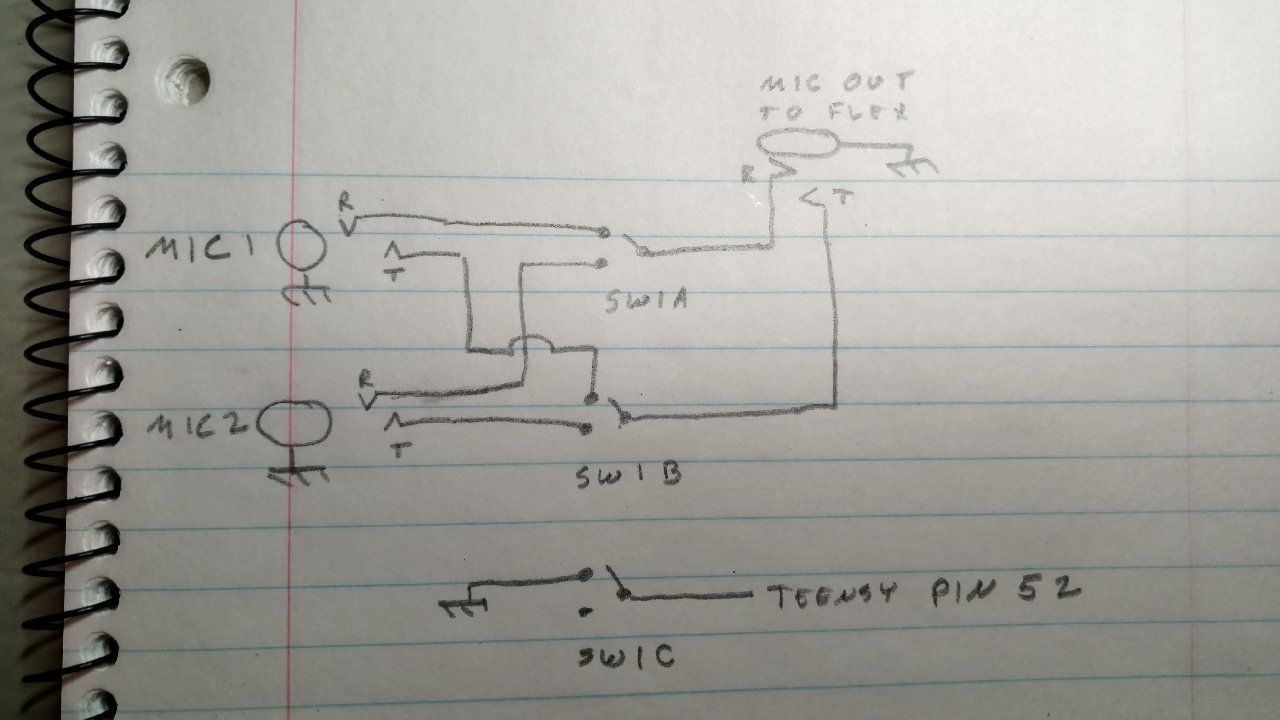
There is not a lot of functionality provided by the touch screen, but it is handy for a couple of things (see the TeensyMaestro User Manual for more info). Its main advantage is the screen reset touch area in the lower right-hand corner. If you are having display problems, touch this area and you will get a screen reset without restarting the whole TeensyMaestro.

* + 1. **CW Jacks**

The Paddles jack needs to be a Tip, Ring, Sleeve (TRS) stereo jack. The Radio Key Line and Straight Key jacks can be either stereo or mono. Only the tip and sleeve connections will be used. Bypass the connections at the jack with .01 uf capacitors.

* + 1. **Mic Jacks and Switch**

Here is the schematic for the mic jacks and switch. The switch is three pole, double throw toggle switch (i.e., not a momentary switch).



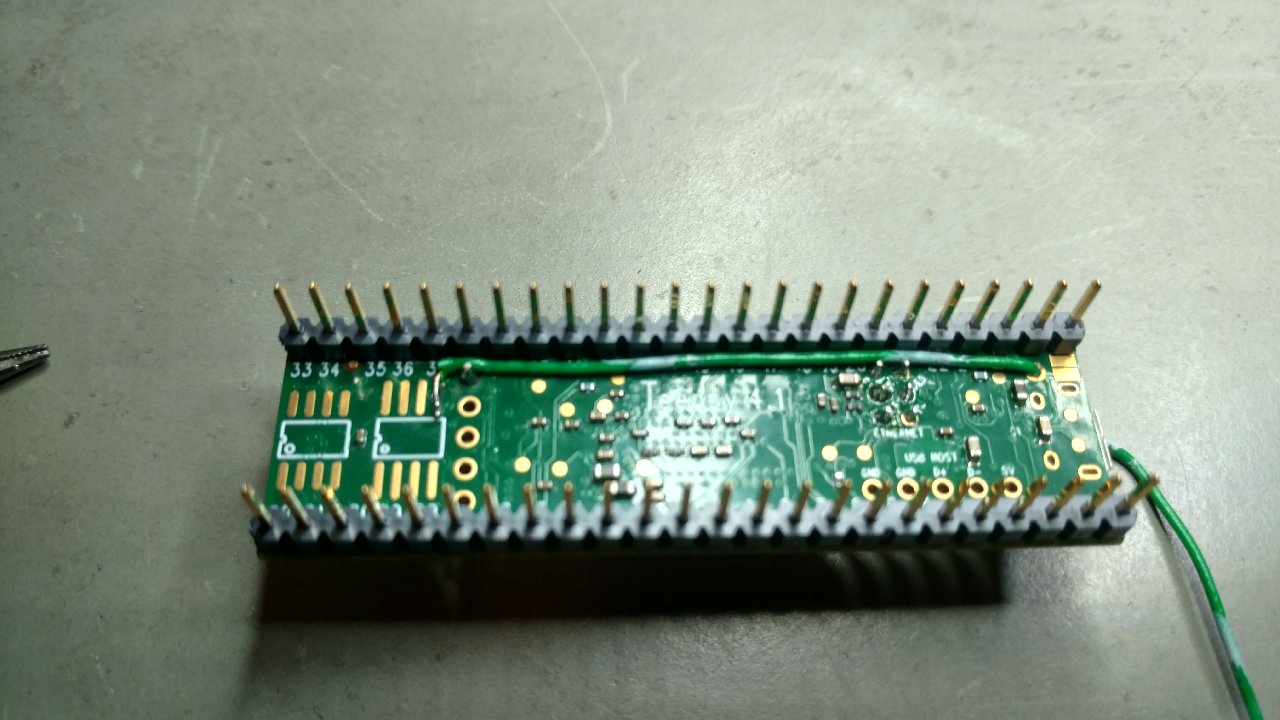
The reason for this is to allow you to plug two mics into the back of the TeensyMaestro. The switch selects the mic. The magic happens in the third pole of the switch. This is where a pin on the Teensy board is either grounded of left high. The TeensyMaestro program then selects the proper mic profile for the selected mic based on the third pole. You specify the mic profiles in the MMConfig.ini file:

Mic Profile Switch Overrides Global: ON ; ON, OFF

Mic Profile 1: <Len FHM-3> ; Mic Profile Switch position 1 (Pin 52 gnd)

Mic Profile 2: <Len CM500> ; Mic Profile Switch position 2 (Pin 52 open)

The only tricky part of this is soldering a wire onto the bottom of the Teensy board to get access to pin 52.



Pin 52 is leg 5 of the second FRAM chip location. I just soldered some #20 solid telephone wire to it and ran the wire along the pin header and out a thru-hole on the board. This provides strain relief so that the wire is not accidentally pulled from the board. For extra safety, I super-glued the wire to the board. The other end of the wire is terminated in a .1 inch header pin so that the Teensy board can be removed without unsoldering anything.

So why use pin 52, of all things? Well, all the other pins were used…

1. Show Off Your Work!

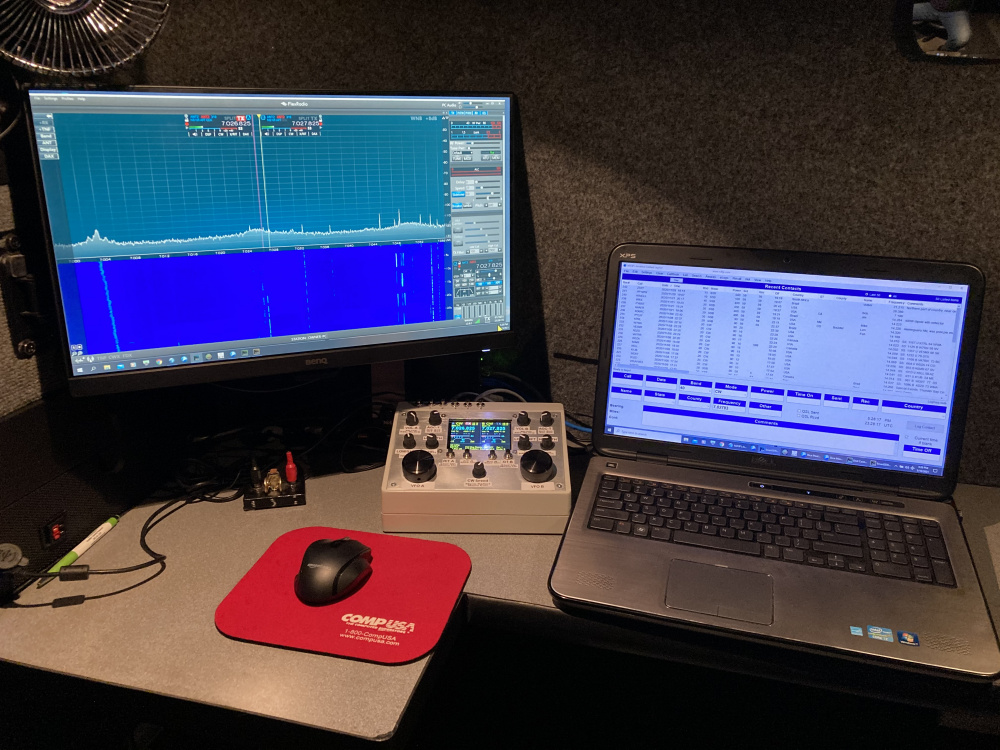
Post pics of your handiwork on the Flex Community forum. There have been quite a few put together now, and each builder has brought a little something to the mix. Here are a few samples:



Len, KD0RC showing the back panel, including mic jacks and switch.



Dave, W4WKU in his Ham Van with TeensyMaestro #2.



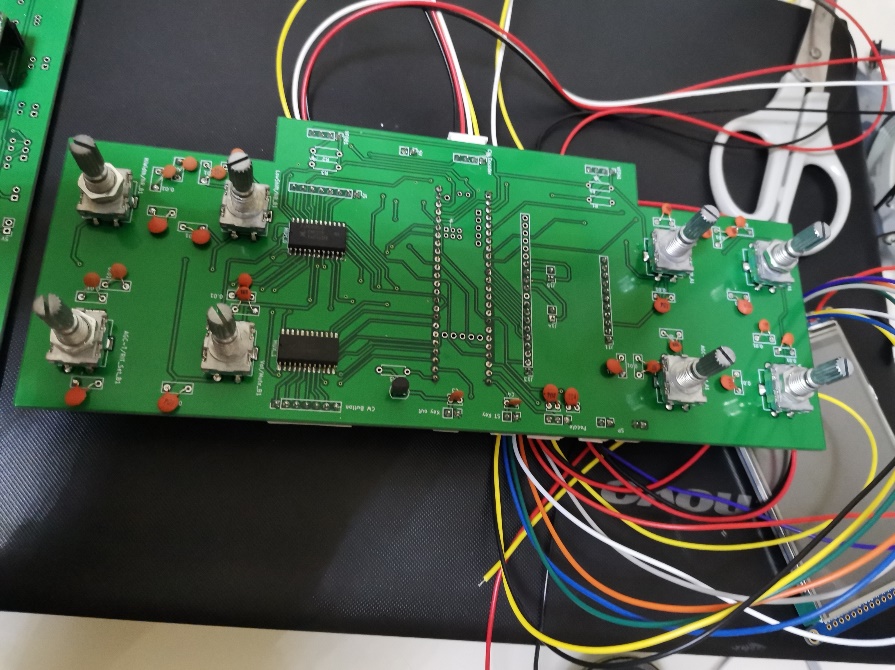
Dave, W4WKU



Steve, AA3SS (Front Panel Express, black anodized with engraved, color-filled labeling – Nice!)



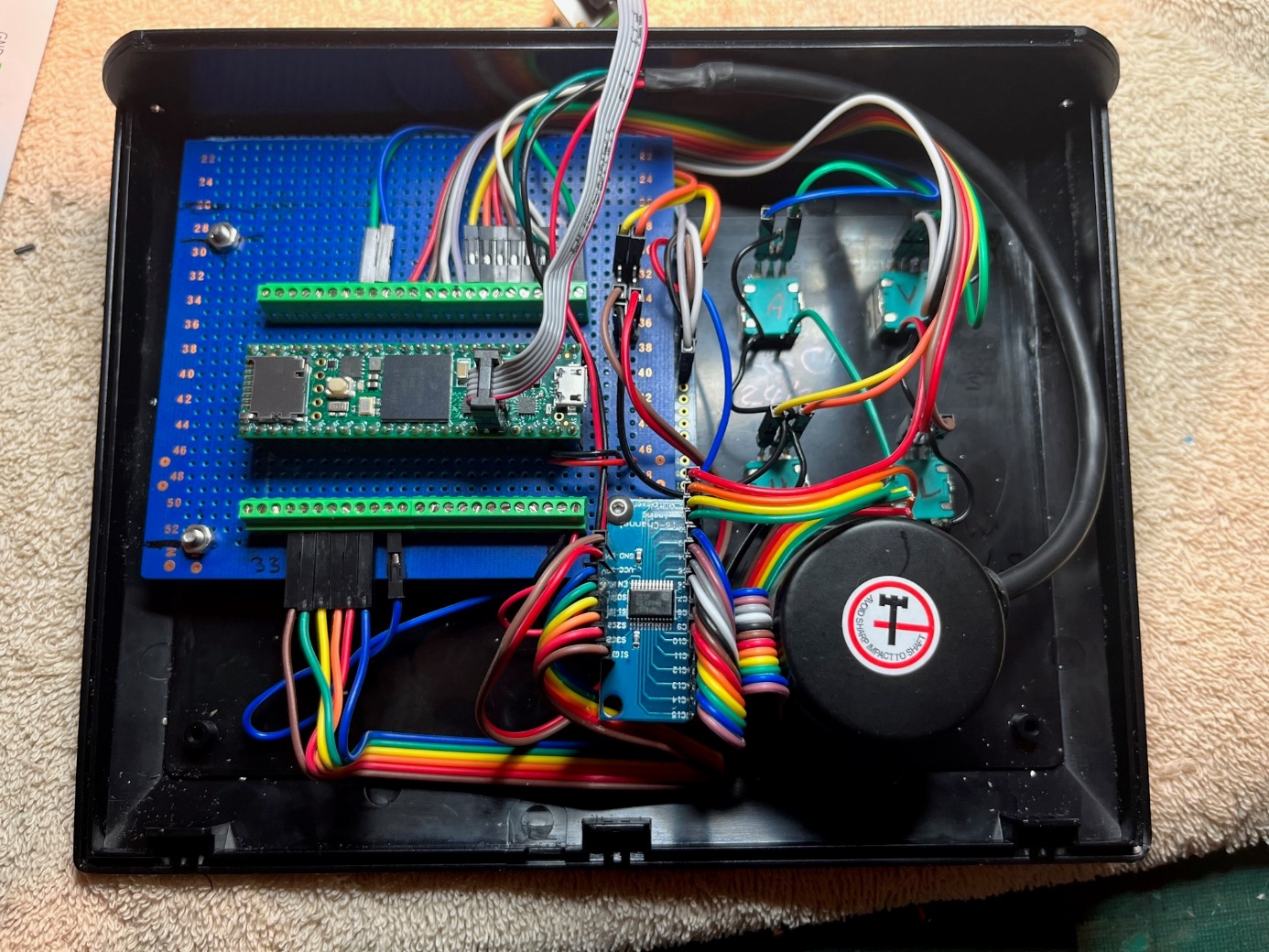
Stephan, VA3AR (still has the protective plastic on it).



Nice board from Ben, VR2VIY.



Ben, VR2VIY



Neat layout from Mike, M0MDS. Note the use of screw headers with .1 inch jumpers. This might be the best of both worlds. The ease of solderless jumpers with the security of screw headers.



Mike, M0MDS