A simple dual transverter IF switch and sequencers for the Elecraft K3S.

By Mark Horn M0WGF.

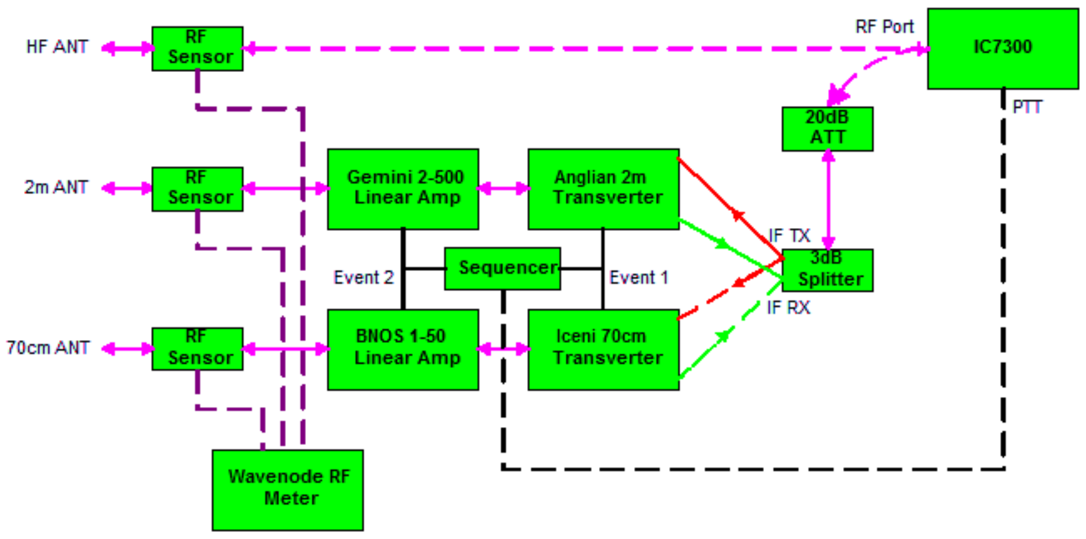
Interests in radio.

My main interest in amateur radio is in Earth-Moon-Earth (EME), Meteor Scatter (MS), Machine Generated Modes(MGM) and the UKAC (UK Activity Contest) primarily on the 2m band as well as the 70cm band.

Consequently over the past few years I have been attempting to build a high performance station built using a G4DDK transverters along with a HF transceiver to give the best performance I can afford on my budget.

Overview.

I have a medical condition called Muscular Dystrophy which results in the muscles of body slowly atrophying over time which has resulted in my hand muscles weakening to the point I struggle connecting and disconnecting any type of RF connector; this poses a problem in that I use transverters for my 2m and 70cm systems and constantly need to swap the 28MHz IF lines.



The original station configuration had the disadvantage in that the IF lines between the radio and transverter would need switching between transverters each time I wished change from 2m to 70cm. I also needed to unplug the HF antenna from the IC7300 and instead connect a 20dB attenuator and 3dB splitter to reduce the RF power down to around 0dBm and provide the RX / TX IF lines required by the transverter. Consequently each time I wanted to switch between either the transverters or the HF bands it felt like a major station configuration change.

Another issue with the IC7300 was getting a modulation that was free of spurii, this wasn't an issue when using the transceiver on HF at 100W but when coupled with a transverter and high power amplifier the spurii in the adjacent 3kHz bandwidth or wider could be less than 40dB down in relation to the peak output.

I tried resolving this by using a modification by Kuhne electronics in Germany which places a small circuit board consisting of a couple of relays between the exciter and internal PA to provide TX and RX IF lines however the quality of modulation was still poor.

The main cause seems to be in the way Icom has implemented the ALC and after reading reports from other operators it was apparent that getting the IC7300 to play nicely with any transverter varies from one IC7300 to another.

A station sequencer is employed to ensure any pre-amps are switched out of the TX signal path as well as to ensure hot switching doesn't occur to any relays in said path before any RF is generated by the exciter by placing the exciter as the last event in the sequence.

The current setup used a single sequencer shared between the transverters and other devices though this was perfectly adequate I wanted the new system to use separate sequences where each event could be configured or at least modified at a later date without too much issue.   It would also help if there was any RF leakage between adjacent relay contacts in the proposed IF switch wouldn't be broadcast on the inactive band.

Both of my transverters where built with a single RF port with an internal relay to switch the port between the RX and TX ports on the transverter module however this isn't ideal.

With EME we are dealing with path losses of around 250dB and as such the received signal is extremely weak so much so that you need to gain every decibel of receive performance you can, especially in a small system, even the losses incurred in multiple connectors and relays in the RX path can cause a dB or more of loss, with this in mind I also decided to modify both my transverters to provided a separate RX and TX RF ports.

The station also utilises an Wavenode RF power meter to monitor in real time the VSWR of each antenna. In the event of a high VSWR the Wavenode closes an internal relay this can used to connect to an external monitor to cease operation of the station.

So with the above in mind I decided to completely re-engineer the station to give the best receive and transmit performance I could get on the 2m and 70cm bands, reduce the amount of cable swapping and have separate sequencers for each of the transverters.

To cure the poor quality of the generated signal from the IC7300 I decided to sell it and instead purchase an Elecraft K3S (K3) instead.

The K3 helpfully provides both RX and TX IF ports with the TX port output at 0dBm giving the correct level of drive for the transverters.

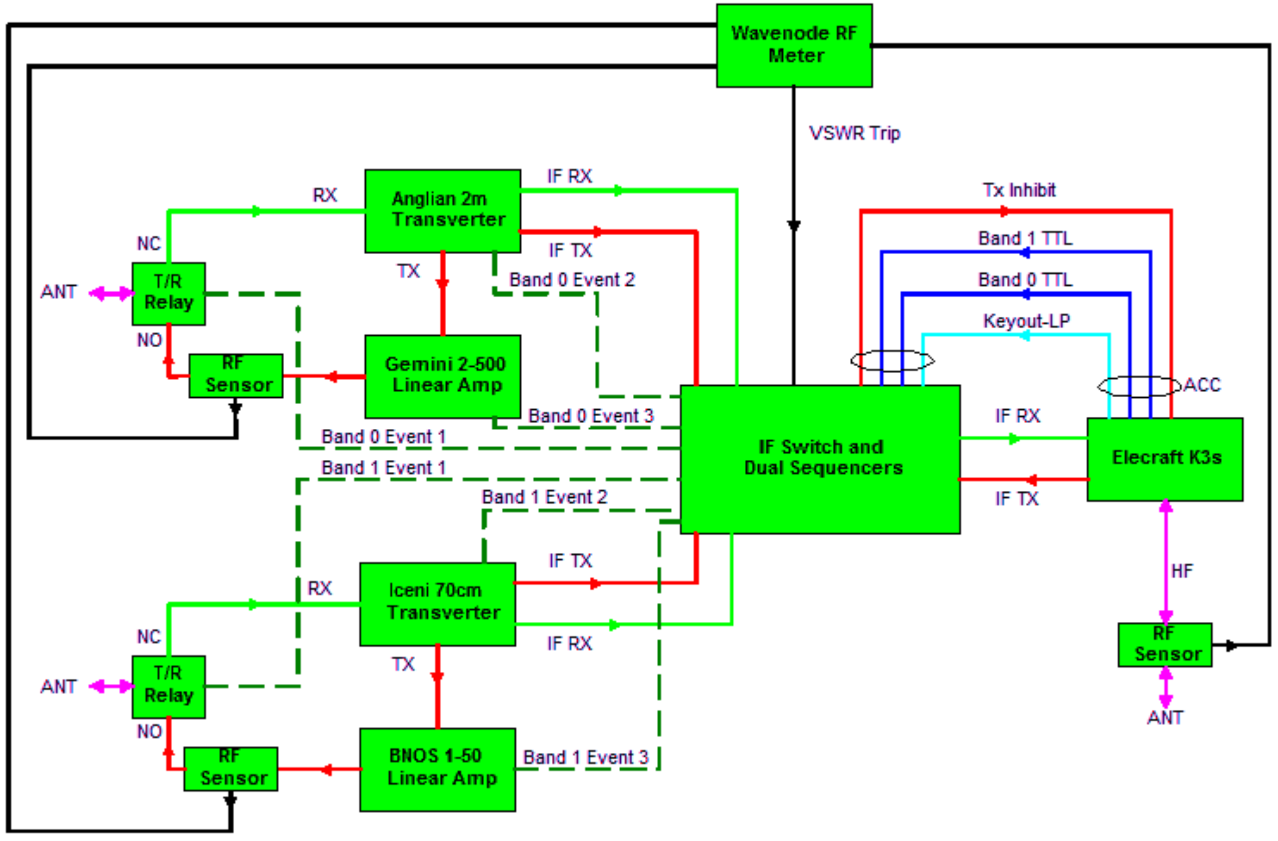
A New Approach.

Upon reading the K3 manual I discovered the ACC port provides four TTL level, low (0V) or high (5V), logic gates where a combination of the four indicates which band is currently selected.

The ACC port can also provide PTT and TX inhibit functions again via two logic gates which would also reduce the amount of cabling required as well as ensure the all devices in the TX chain had switched fully before any RF is generated and vice versa for the RX chain.

The ACC port consists of a DE-15 socket where pins 3, 9, 13 and 14 provide access to the TTL logic gates and pins 7 and 10 provide the TX inhibit (TXINHIB), which can be configured to inhibit the exciter on either a high or low state, and KEYOUT-LP (PTT) which provides a 10mA sink to ground.

This was exactly what I needed to create a single control unit to switch IF lines, provide an independent sequencer to each transverter chain and to ensure that no RF was generated until all events in each sequence was completed all at the press of the band selector button on the K3 after some thought I came up with the following station configuration.



Hardware Configuration.

The K3s band logic gates can be configured in one of three ways NOR for HF bands only, TRN for transverters only and HF-TRN for both the HF bands and transverters.   I selected the TRN setting which made reading the TTL logic simpler as it only requires knowing the current state of two of the logic gates instead of the four gates required for the NOR or HF-TRN configurations.

From here on forward when I reference bands it is taken to also include the transverters.

ADDR	BAND0	BAND1	BAND2	BAND3
TRN1	1	0	0	0
TRN2	0	1	0	0
TRN3	1	1	0	0
TRN4	0	0	1	0

The band switch logic matrix for the K3 is shown left.

As you can see from the table when TRN1 or TRN2 is selected the logic can either be on off for either band but they can neither be off or on together for band 0 and 1 so any other state would be equal to another band.

So with the above in mind I needed firstly someway to read the K3 TTL logic.

Secondly to key the 2m and 70cm transverters as well as associated amplifiers a 10mA sink to ground.

Thirdly the change over relays, Tohtsu CZX-3500, which require 12V and draw around 250mA, which would be held closed for RX operation and released for TX.   The CZX-3500's also have an advantage or providing an extra 20dB of isolation between ports compared to the ones in the transverters which only provided 40dB of isolation.

Fourthly the ability to control two mini relays to switch the IF lines between transverters.

Fifthly It would also be useful to have some indication of which band has been selected as a separate confirmation to the K3, and that events for that band are being sequenced correctly.

Finally it would be nice to have some kind of audible alert in the event of a high VSWR.

Hardware and Software.

Thinking about how to go about accomplishing this I came to the conclusion using an Arduino microcontroller would be the easiest route to getting something working fairly quickly with little expense as well as keeping the electronics side of things simple.

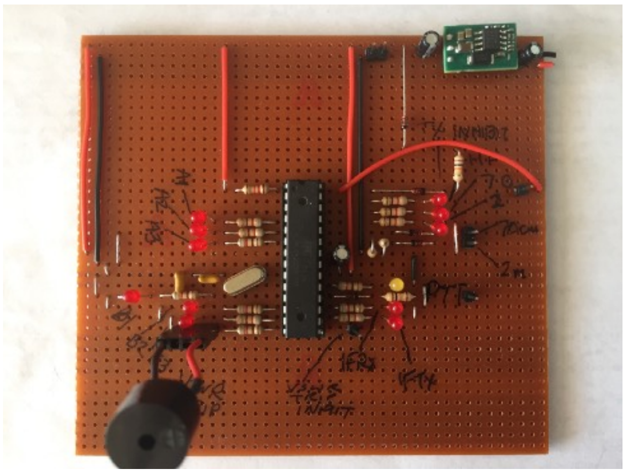
The advantages using an Arduino :

1. Native ability to read the TTL logic from the K3.
2. General purpose I/O pins capable of providing or sinking 5V at 20mA.
3. Each I/O pin is protected from under and over voltage conditions using clamping diodes.

The Arduino UNO uses an ATMega328P which has 23 general purpose digital I/O pins available however two are used for serial coms and one for a reset button, two pins are used for an external 16MHz crystal, so minus those five I/O pins eighteen pins are available to for other purposes.

I decided I need the following pins:

2 pins to read the band switch TTL logic from the K3.
3 pins to light LED's to indicate which band has been selected on the unit.
2 pins for the IF relays.
1 pin to monitor PTT enabled from the K3.
1 pin for a PTT LED indicator.
1 pin to enable and disable TX Inhibit function on the K3.
1 pin to monitor the VSWR trip alert from the Wavenode RF power meter.
1 pin for the VSWR trip enabled warning buzzer.
1 pin for a VSWR trip reset button, for which I'll use the default Arduino reset pin.
6 pins for the two sets of three for the sequencer, one set for each band selected.

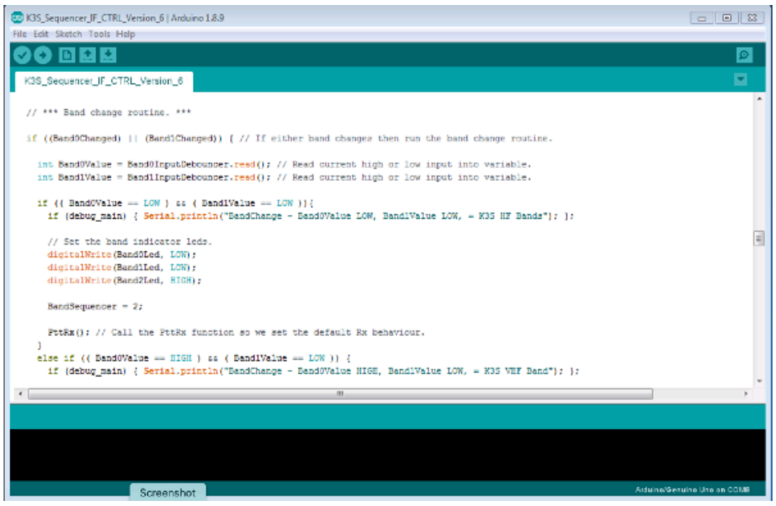


The first thing I did was to create a small prototype board so I could write the software and check everything worked with the K3 before moving on to the proper electronics, the purpose of the prototype board was to read the K3 TTL logic, show the state of the IF relays and sequencers using LED's to indicate there current state as well test the TX inhibit function on the K3.

As is always the way with these things, having a shack that looks like a small nuke has gone off, I couldn't find my breadboard to so had to make a test rig including an ATMega328 using some strip board instead this is also had the advantage that once it was wired up no amount of moving it around on the desk the wiring will stay error free.

Programming the Arduino is fairly straight forwarding using the Arduino IDE from the Arduino website[1]. The language used is a subset of the C/C++ language which is fairly simple to use and there are plenty of resources on the internet to get you started.

The first iteration of the software was a simple loop that read the state of the I/O pins that monitor the band logic gates on the K3 and lit an LED to indicate which transverter / band was selected.


K3S_Sequencer_IF_CTRL_Version_6 | Arduino 1.8.9
File Edit Sketch Tools Help
K3S_Sequencer_IF_CTRL_Version_6
Screenshot
Arduino/Genuino Uno on COM8

```
// *** Band change routine. ***

if ((Band0Changed) || (Band1Changed)) { // If either band changes then run the band change routine.

  int Band0Value = Band0InputDebouncer.read(); // Read current high or low input into variable.
  int Band1Value = Band1InputDebouncer.read(); // Read current high or low input into variable.

  if (( Band0Value == LOW ) && ( Band1Value == LOW )){
    if (debug_main) { Serial.println("BandChange - Band0Value LOW, Band1Value LOW, = K3S HF Bands"); };

    // Set the band indicator leds.
    digitalWrite(Band0Led, LOW);
    digitalWrite(Band1Led, LOW);
    digitalWrite(Band2Led, HIGH);

    BandSequencer = 2;

    PttRx(); // Call the PttRx function so we set the default Rx behaviour.
  }
  else if (( Band0Value == HIGH ) && ( Band1Value == LOW )) {
    if (debug_main) { Serial.println("BandChange - Band0Value HIGH, Band1Value LOW, = K3S VHF Band"); };
```

What became apparent with this first simple loop was despite the fact there isn't any mechanical switches the software was experiencing contact bounce, also known as chatter or button bounce, seen in this instance as the wrong LED being lit for the band selected.

Contact bounce[2] is caused when a switch is pressed or released and for a few milliseconds or more the switch state is unstable to get around this problem it is normal to get the state of the I/O pin and store it in variable and then wait for a defined period and check again when the stored state and current state are equal we can be sure the switch is either high or low.

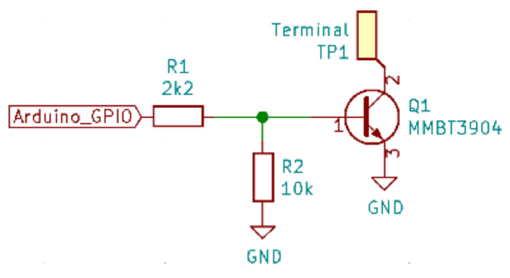
There was two issues caused by contact bounce, one if the Arduino was switched on before the K3 then the band selected on the Arduino may not be in agreement with the band selected on the K3 and secondly when the band was changed on the K3 occasionally the wrong band was detected.

To resolve for these issues I started writing my own routine to take into account the unstable state of the switch while bouncing but in a sudden moment of genius decided not to reinvent the wheel and instead download and installed a library called Bounce2.h[3] which reduced the amount of code I had to write for myself.

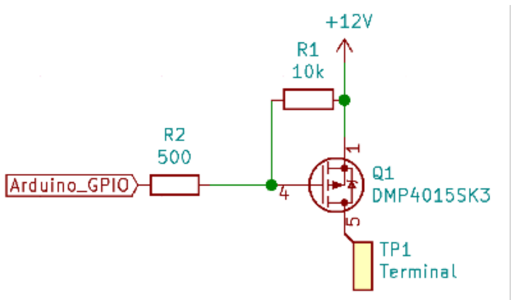
With the contact bounce cured it took an evening to write the software[4] and that included having to remind myself how hardware is referenced on the Arduino, the web is great for these kinds of projects there's always plenty of example code around if you're unsure how to do something.

With the software written the next thing to do was get on with the hardware, now electronics isn't my forte so this wasn't something I was looking forward too.

I knew for sequence events two and three some simple NPN transistors in low side configuration would suffice however for event one I needed the ability to switch between 5V to 48V drawing up to 1A, so that in the event I need to replace the change over relays I wouldn't be restricted in what hardware could be used, so I chose to use a P-Channel MOSFET in a high side configuration.

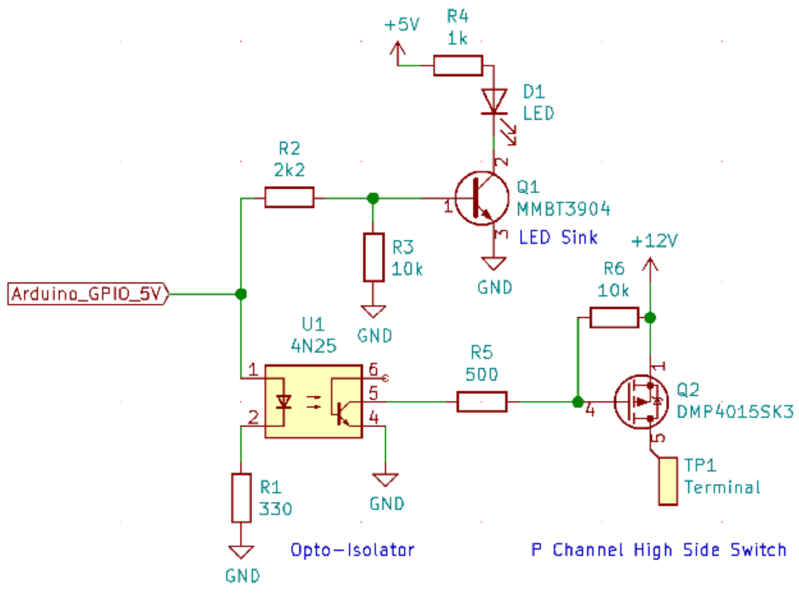


NPN Sink



P Channel High Side Switch

I had discussion with a friend who convinced me that isolating the high voltage side side of the circuit would be prudent, it's not something I would have bothered with however opto-isolators are cheap as chips so I decided to isolate all of the sequencer events this would allow me to change the hardware configuration at a later date if I wanted to.



As I had moved to the opto-isolators I decided instead of turning the sequencer event indicator LED's on direct from the ATMega I/O pins I would instead add an NPN switch in parallel to the opto-isolators to sink the LED cathodes to ground instead.

This seems to be overkill but as MMBT3904 NPN transistors are only five pence each including them wasn't going to cost any significant amount and ensure if I altered the circuit later on I wouldn't run foul of exceeding the 20mA limit on the ATMega I/O pins.

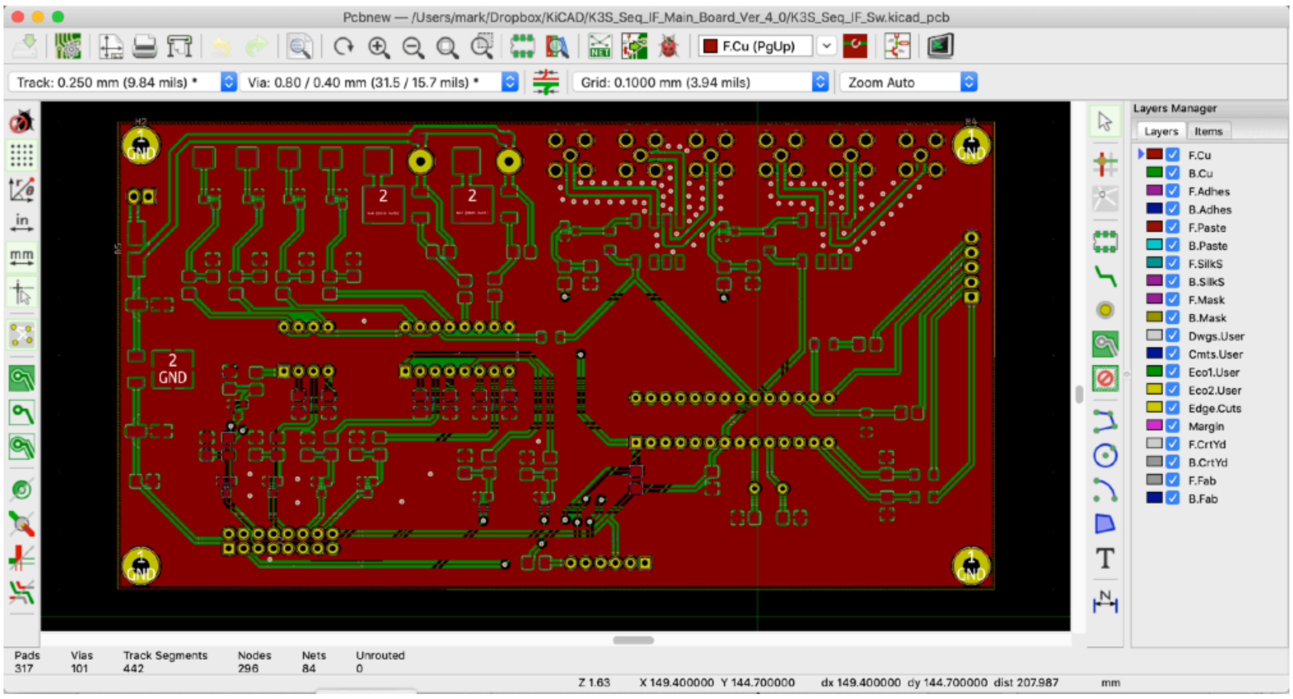
PCB Design and Manufacture.

With a general plan of action I decided instead of building the unit on strip board I would instead have ago at using KiCad[5], to develop both the main PCB and front panel LED PCB.

KiCad is a cross platform open source electronics design automation suite, which allows you create a circuit diagram of your design, perform the PCB layout and display a 3D rendition of the final product.

KiCad can also generate an LTSpice file from the circuit diagram as well which is handy if you wish to run a simulation of design.

KiCad, shown below, is fairly simple to use and there are a huge number of tutorials available on the internet.   I will also be giving talk and demonstration at a CVRS club meeting early 2020.


Pcbnew — /Users/mark/Dropbox/KiCAD/K3S_Seq_IF_Main_Board_Ver_4_0/K3S_Seq_IF_Sw.kicad_pcb

F.Cu (PgUp)

Track: 0.250 mm (9.84 mils) *

Via: 0.80 / 0.40 mm (31.5 / 15.7 mils) *

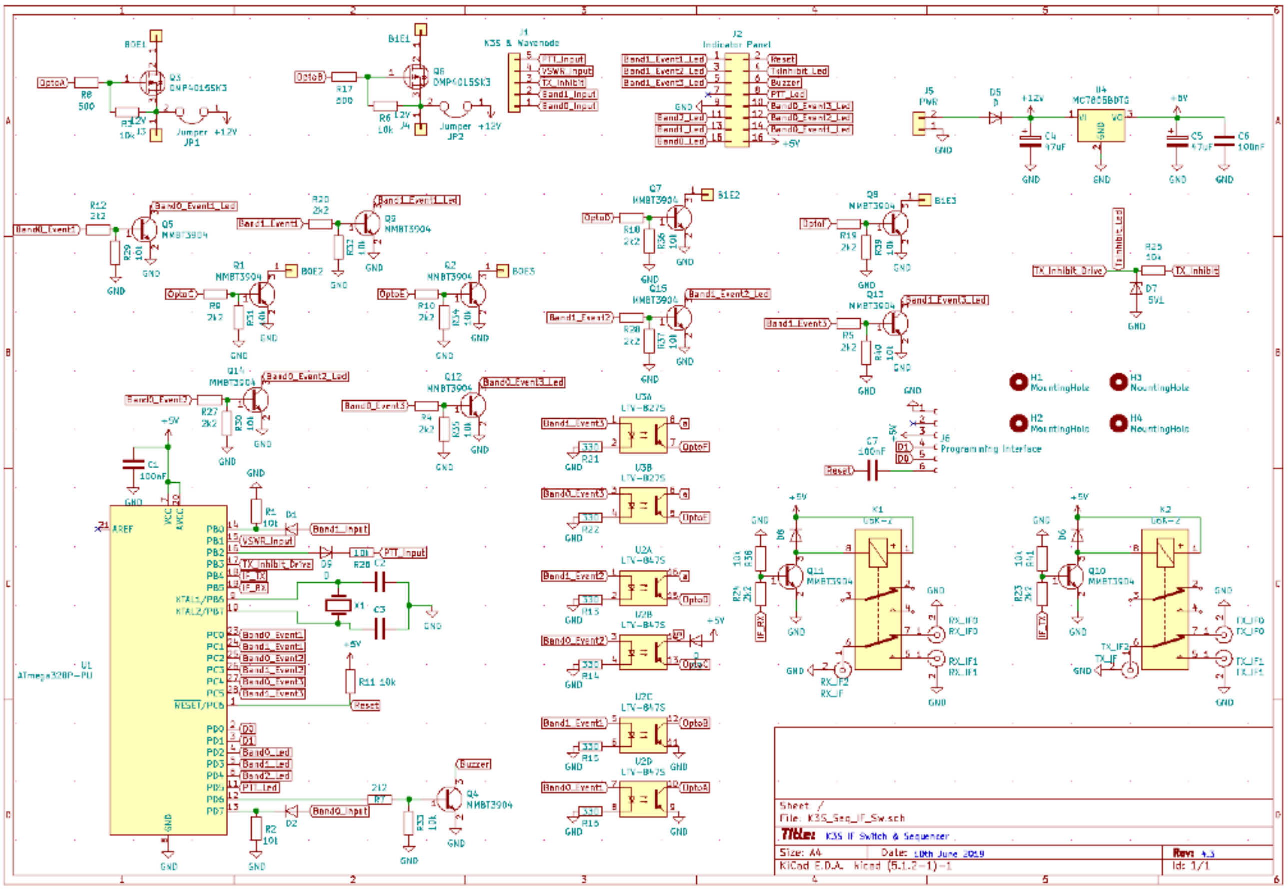
Grid: 0.1000 mm (3.94 mils)

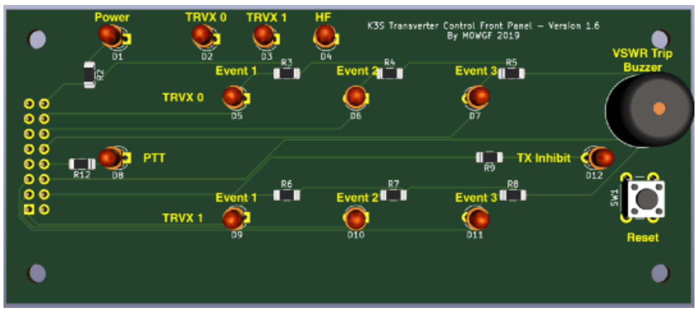
Zoom Auto

```
Pads  Vias  Track Segments  Nodes  Nets  Unrouted
317   101   442             296    84    0
```

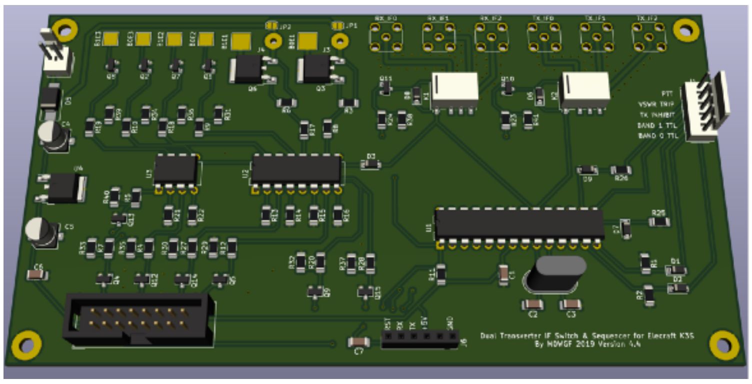
Z 1.63   X 149.400000 Y 144.700000   dx 149.400000 dy 144.700000 dist 207.987   mm

Below is full circuit diagram as created in KiCad.





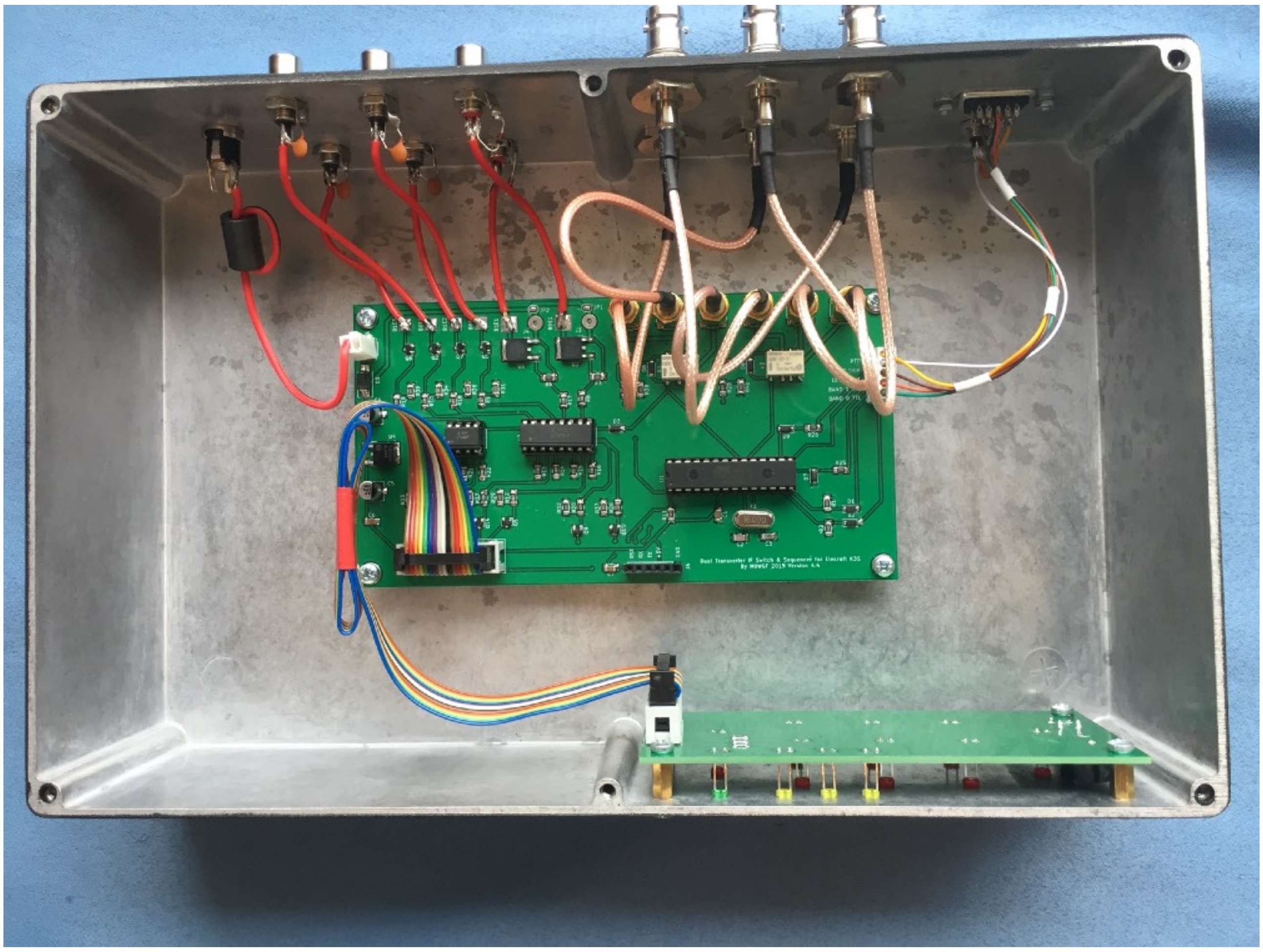
I decided to place all the LED indicators, high VSWR buzzer and reset button on a small PCB, shown left, which could then be mounted to the front of the case this removed a lot of wiring that would have been required if I had mount everything individually. I used IDC connectors to connect the two boards together which again made everything easy to plug and if for any reason there was a fault removing either board would only take a few moments to unplug the cables and pop the screws out. Shown below is the main board.


Dual Transverter IF Switch & Sequencer for Elecraft K3S
By M0WGF 2019 Version 4.4

The PCB's were manufactured by FirstPCB6, a Chinese company, they're cheap to use in fact the postage costs more than the boards. You can however find cheaper manufactures but the time the boards take to come from China can take quite awhile as they're normally sent surface mail instead of air mail. From the date of order they take two weeks to arrive they do give you the option of multiple carriers so you can pick from the cheapest offer available at the time of purchase.

The KiCad project files and Gerber files can be downloaded from Dropbox[7]. I do have a couple of sets of boards spare if anyone wants to build one of these units please drop me an email if interested.

Once the PCB's had the boards had arrived and populated I mount the them in a Hammond '1550' 275 x 175 x 67mm die cast enclosure which matched the transverters.



Hopefully this quick overview of this small project gives you some inspiration to create your own hardware or software and create something that is tailored to your own needs.

1.) https://www.arduino.cc/en/Main/software
2.) https://en.wikipedia.org/wiki/Switch#Contact_bounce
3.) https://github.com/thomasfredericks/Bounce2
4.) https://www.dropbox.com/s/za1mdbixwdgjyzf/K3S_Sequencer_IF_CTRL_Version_6.zip?dl=0
5.) http://www.kicad-pcb.org
6.) https://firstpcb.com
7.) https://www.dropbox.com/s/n1fn1lh4imprqs1/K3s_Sequencer_IF_ProjectFiles.zip?dl=0