A Smart Fool-resistant Conditional Sequencer

Paul Wade, W1GHZ ©2017
www.w1ghz.org

I'm still trying to design a fool-proof sequencer for transverters and amplifiers, but there are still bigger fools. My previous attempts tried to do it all in hardware. I finally decided to break down and include a microprocessor in an attempt to make it smarter. Now the software can be changed as desired to make it more fool-resistant – or less, if you prefer to keep blowing things up. And software is more flexible – more easily modified than hardware. This one is my fourth sequencer design, so I call it the Mark 4. Figure 1 shows one in a simple lunchbox transverter for 3456 MHz.

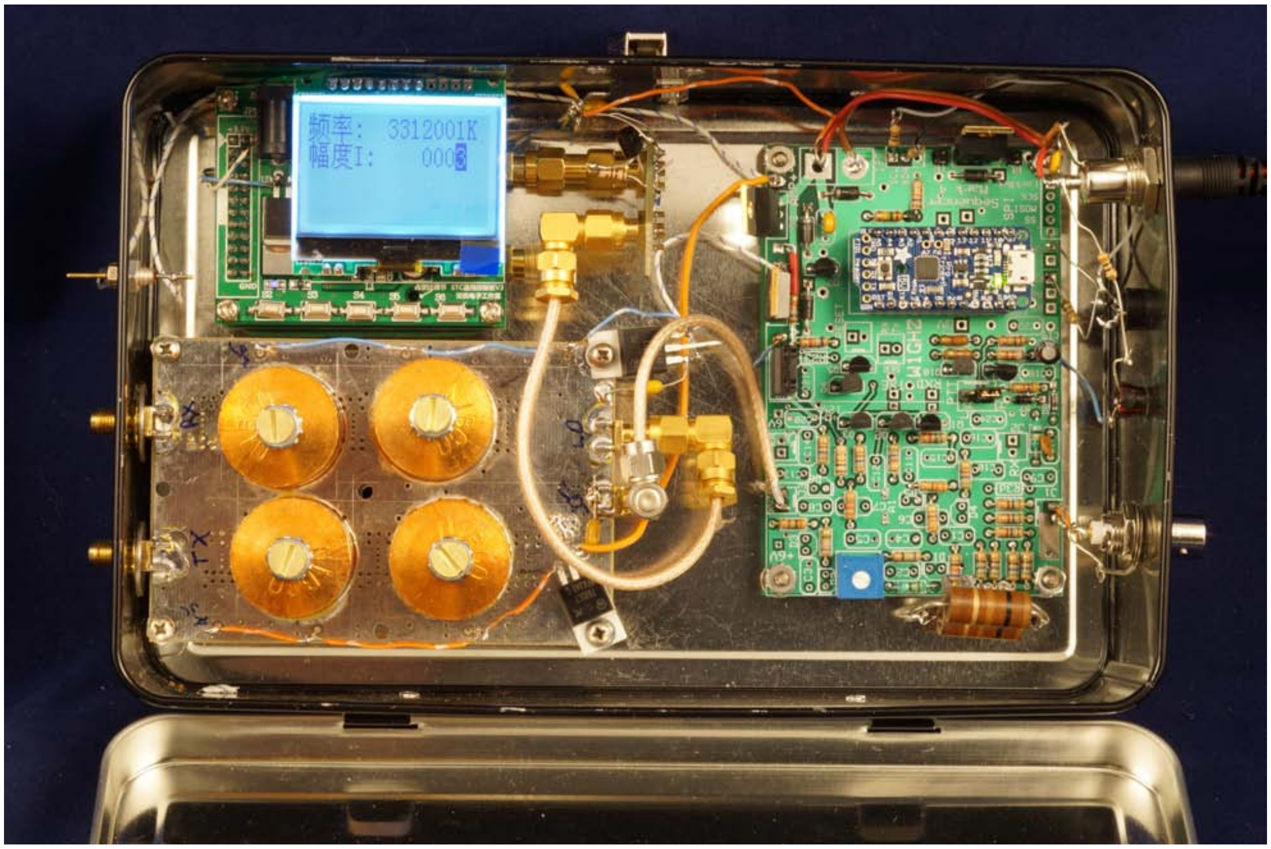


Figure 1- Simple transverter for 3456 MHz with Mark 4 Sequencer at the right

The problem a sequencer attempts to address is that most transceivers transmit power immediately, before anything else can switch from TX to RX. This unwanted power can burn out mixers and preamps, as well as coaxial relays that are not meant to hot-switch RF power – it doesn't take much power to damage a relay contact while it is switching. Another common problem is that many transceivers output a spike of excessively high power until it is reduced by a poorly-designed ALC loop.

Some approaches[1] to dealing with the problem involve trying to delay the transceiver output, either by intercepting the PTT line from the microphone (what about the key) or by controlling an **Inhibit** line to the transceiver. Both of these work well, but need additional connections to the transceiver – rig-specific, and additional points of possible failure.

Another approach is to cross your fingers and hope nothing blows up – strictly for fools!

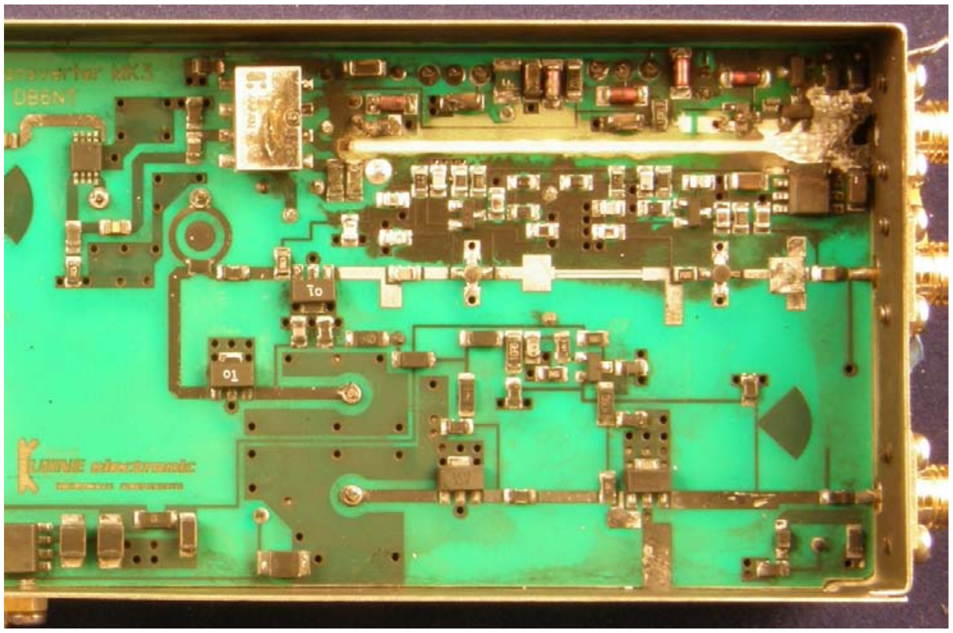


Figure 2- DB6NT transverter with entire input section vaporized by high power

I prefer to have a single connection from transceiver to transverter, with PTT voltage running up the IF coax. To make this fool-resistant, a PIN diode switch controls the RF flowing to the mixer. This PIN switch has three states: RX, TX, and SAFE. An RF sensor detects any transmitted power and switches to the SAFE state to absorb the power until the sequencer finishes all switching. Of course, this is only fool-resistant – a fool could connect a 100-watt transceiver and burn up the whole PIN diode switch. See Figure 2 for the results.

Conditional Sequencer

Most of the published sequencers are just clockwork sequencers – start them and they go through the whole sequencer at fixed times. But if we include some intelligence, we can wait until an operation completes before continuing, and take appropriate action if there is a problem: many coax and waveguide switches have auxiliary contacts to indicate completion, or we could check to make sure an amplifier is ready before applying RF. I call the sequencer that operates intelligently a conditional sequencer.

What I haven't figured out is how to make sure the antenna is connected, except to shut down as quickly as possible if high VSWR is connected. I can assure you that the fastest fuse in the world is an expensive RF transistor. The one possibility might be to measure VSWR with the first RF output and hope it can be turned off before the amplifier is damaged. Newer RF power FETs are more tolerant of VSWR, so this might work.

My previous fool-resistant sequencer[2], the “Even More Fool-resistant Conditional Sequencer,” used a lot of discrete parts to provide sensing and an intelligent state machine. It works really well, but requires a large PC board and a whole lot of assembly and soldering. The parts are all cheap and readily available, but it takes a whole bunch. I recently tried to see if I could shrink it and simplify it to essentials, but without much success.

Microprocessor

One reason I have resisted microprocessors in my designs is that there are too many choices. I want one that is readily available (and will continue to be), cheap, and easy to program. I prefer high-level languages rather than use assembly code or BASIC. Another part of programming is getting the program loaded into the chip – many require special hardware, or a painful process with older microprocessors.

The Arduino[3] seems to be very popular in ham projects. I bought the "Arduino Projects for Amateur Radio" book[4] by Jack Purdum, W8TEE, and Dennis Kidder, W6DQ. It looked pretty straightforward to program in C, with a good IDE (Integrated Development Environment), and the programming connection is ordinary USB. I picked up an Arduino UNO board and wrote a few sketches (that's what Arduino programs are called).

For the sequencer, I did a bit of web research and settled on the Adafruit Pro Trinket from Adafruit[5]. This is an Arduino implementation on a small PCB with integral USB connection for $10. It has adequate IO pins to do a lot more than just a sequencer, and comes in both 3.3 volt and 5 volt versions, which could prove useful for additional options.

The sequencer code started with a simple clockwork sequencer similar to the example in the W6TEE & W6DQ book. Then I expanded it to a state machine with conditional steps and abort mechanism, and debounce on the PTT input to avoid false starts and twitchy thumbs. This only required about 10% of the available internal memory on the Arduino chip, so there is lots of room for other options.

Since the Arduino has some analog inputs as well as digital, I added a voltage monitor for the 12 volt input, driving a bicolor LED to indicate good, marginal, and low, and to prevent transmitting if the voltage is too low.

Other Options

The Arduino Pro Trinket has some standard interfaces defined, like SPI and I2C, that are used by many other chips. In particular, some microwave synthesizer chips from Analog Devices have SPI interface for setting frequency, so the Sequencer could also initialize the LO frequency on power-up.

Some other possible additions to the sequencer:

- VSWR monitor and shutdown (analog inputs)
- Amplifier voltage
- Temperature monitor (external sensor)
- TR relay auxiliary contacts
- Latching relay driver
- Dash sender
- CW ID
- Call and grid string
- CW keyboard
- Rotator control
- Grid calculator from GPS output
- Multiband transmit protection
-

Signal Names

To avoid confusion, we try to use the same signal names in hardware and software, and to give them logical significance: a signal name with suffix “_H”, like signalname_H, is asserted high, so that a high voltage state is a logical 1 or TRUE, and a low voltage state is a logical 0 or FALSE. Conversely, a signal name with suffix “_L”, like signalname_L, is asserted low, so that a low voltage state is a logical 1 or TRUE, and a high voltage state is a logical 0 or FALSE. Signal names with no suffix are assumed to be asserted high, but often we use both versions of a signal by inverting it, so the suffixes are important.

Since we are coding in **C**, a logical 1 is a TRUE state for comparisons, and a logical 0 is FALSE. We also use Boolean variables, which have two values: TRUE or FALSE.

Another possible confusion is between software variables and hardware signals which are connected to the chip. In the code, I try to include “_Pin” for any hardware signal that is driven or detected by the chip. For example, the PC board hardware signal PTT_L goes to a pin called PTT_L_Pin, which then might be logically inverted in software and become a Boolean called PTT.

Hardware

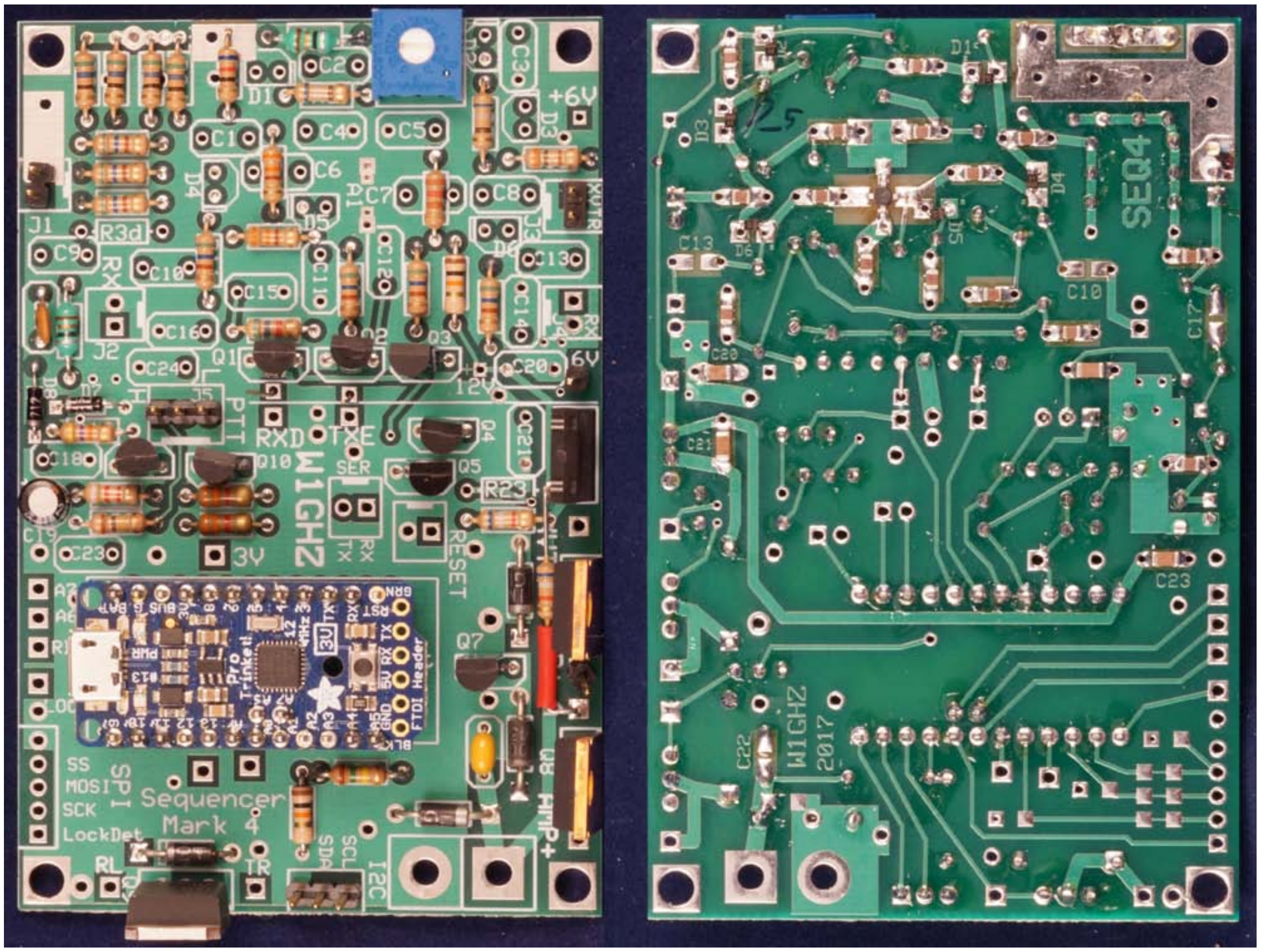


Figure 3 – Sequencer Mark4 Top and Bottom

The Arduino provides the brains but does not have the brawn to drive the external parts. A PC board provides FET drivers for relays and 12 volts at sufficient current for amplifiers. The board also includes the IF interface with PIN diode switch. I've tried to keep the board split into two halves for the two functional areas. The two halves are visible in the photo, Figure 3, and the schematic, Figure 4. In both figures, the top half, above the **TXE** and **RXD** terminals, is the IF interface and PIN diode switch, while the bottom half is the Arduino and control circuits with associated driver FETs.

The PC board layout is for both surface-mount capacitors, on the bottom, and thru-hole components, on the top. I chose to use surface-mount chip capacitors and PIN diodes, to keep most of the RF on the bottom, and ordinary leaded ¼ watt resistors on the top. You could use disc capacitors if you choose, and PIN diodes with leads if you can find any.

IF Interface

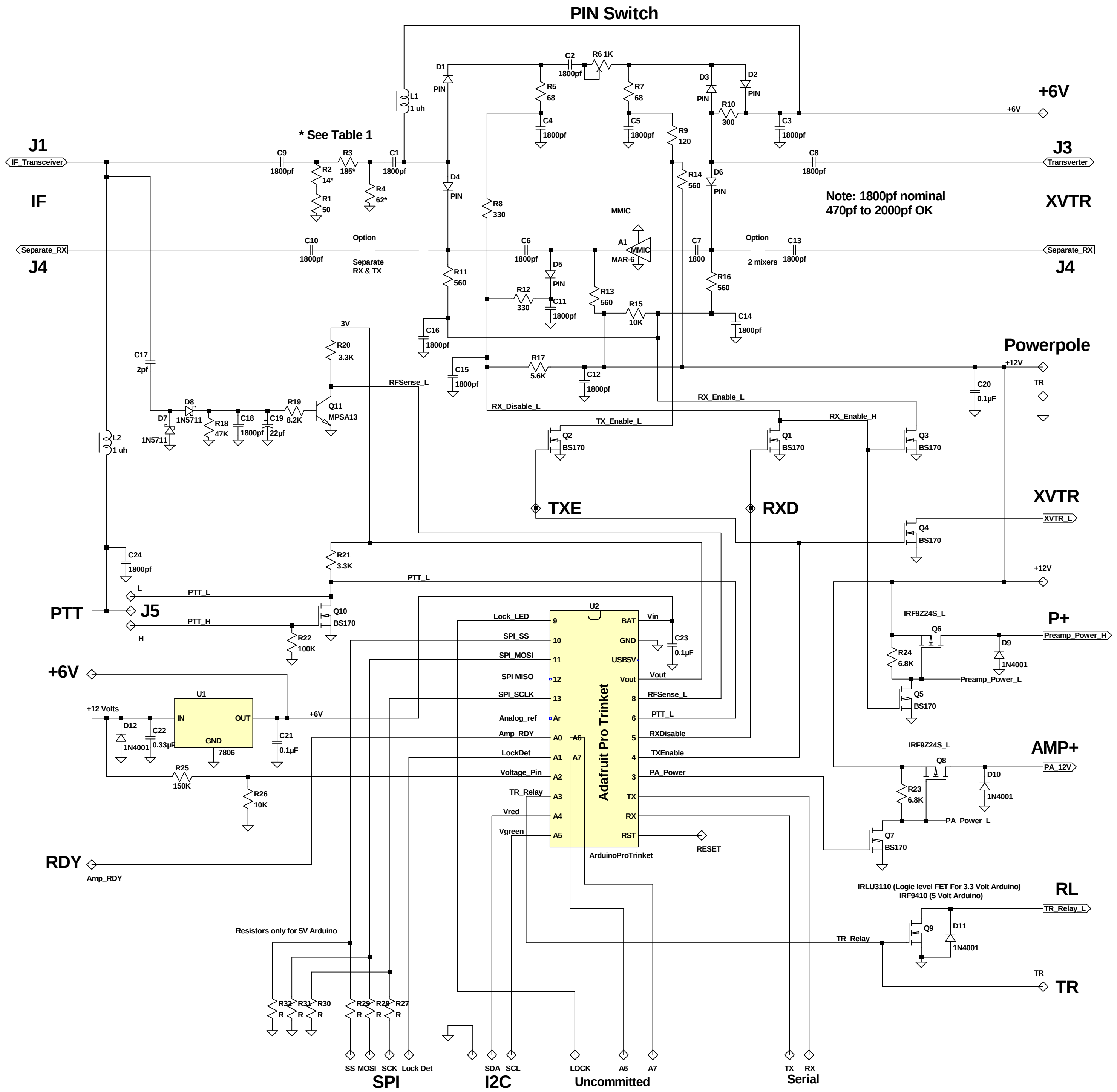
The IF interface starts with a power attenuator R1, R2, R3, and R4 to reduce the transceiver output power to perhaps 50 mW. The resistor values are chosen accordingly to the expected power level – see Table 1 for typical values, with most of the power dissipated in R1, a 50 ohm resistor which can be flange-mounted to the chassis if needed. I use an FT-817 set for 2.5 Watts output, since the ALC oscillates at lower power settings and creates spurs across the band with keying. The interface could handle a transceiver with as much as 25 watts, but that seems like a waste of power.

Attenuator table

The input attenuator may be configured for different power levels and attenuations. This is done in two parts: a fixed input attenuator to absorb most of the power and keep power levels in the PIN diodes to less than 50 mW (+17 dBm) to minimize IMD, and an adjustable attenuator which is switched out on receive. Thus, a 1-watt input attenuator would need to have at least 13 dB of attenuation, while a 10-watt one would require at least 23 dB of attenuation. Then the PIN diode switch includes an adjustable attenuator with about 20 dB of adjustment to set the desired output level for the mixer, usually less than one milliwatt. The limiting factor for the input attenuator is that it is also in the receive path (if all else fails, it limits power going this way to 50 mW also). To recover the loss in the receive path, there is a MMIC amplifier which provides about 20 dB of gain to compensate for the fixed attenuator, and to act as a cheap fuse if some fool were to set his IF transceiver to 100 watts.

So why not just use a low power transceiver for the IF? In most cases, this would require modifying the transceiver to reduce output power, so that it would be unique to the transverter. This limits interchangeability, and things do fail – but only at inopportune times! Another strategy would be to use a low power setting on a transceiver like the FT-817, which has provision for different power levels. However, Leif, SM5BSZ, has tested many rigs for output cleanliness[6], and found that the FT-817 emits spurious garbage when operated at less than 2.5 watts, particularly on CW. N1EKV investigated further and found that the ALC circuit, used to set the power level, oscillates at low power settings. He also found a simple fix, by changing one resistor, but complete disassembly is required to get at the small surface-mount resistor on the bottom of the board. I decided that it is preferable to operate at 2.5 watts and be compatible with all my other IF rigs.

Since the input attenuator must absorb nearly all of the IF power, it is designed so that a large proportion of that power is dissipated in a hefty 50-ohm power resistor – some resistor types may be



W1GHZ 2017

SEQUENCER Mark4

Figure 4

bolted to a heat sink. The rest of the attenuator consists of parallel ¼-watt resistors to make up the desired attenuation while absorbing some portion of the power. The resistor values for the various power levels are shown in Table 1:

IF POWER	**10 watts**	**6 watts**	**3 watts** 1/2 to 3	**100 mw** <250 mw
First attenuator	23 dB	20 dB	18 dB	6 dB
R1	50 @ 8W	50 @ 4W	50 @ 2W	50
R2	27 * 4	43 * 4	56 * 4	100
R3	1.5K * 4	1K * 4	560 * 3 750 * 4	39
R4	56	62	62	150

NOTE: 56 * 4 means four 56 ohm resistor in parallel

Table 1

PIN Diode Switch

The power attenuator is followed by the PIN diode switch, which switches between the MMIC amplifier on receive (RX), a variable attenuator used to set the transmit power to the proper level for the transverter mixer on transmit (TX), or a SAFE state where all power is absorbed while sequencing and switching is taking place, thus protecting the PA, the TR relay, and the preamp from stray RF while the relay is switching. This third state is not possible with a single relay, which is why I have chosen the added complexity of the PIN diode switch. The controlling signals are **RXD** (RX Disable) and **TXE** (TX Enable), both asserted High. To receive, both are deasserted. RXD is asserted, but not TXE, for the SAFE state, and then TXE is asserted along with RXD to transmit. TXE should never be asserted without RXD, a combination that would enable both transmitter and receiver simultaneously.

A PIN diode has an intrinsic (I) layer, neither P nor N, between the normal P and N layers. This makes it operate very slowly, so that it will rectify only at very low frequencies. At high frequencies, the electrons can’t get across the intrinsic layer fast enough, so it looks like an open circuit, with some capacitance, of course. However, if we put some DC current (electrons) through the diode, then the high frequency electrons can travel with the DC electrons, just modulating the DC current. So to make an RF switch, we apply more DC current than the peak RF current to turn it on, and remove the current to turn it off. Adding reverse bias in the off condition reduces the capacitance and improves the isolation. In the PIN diode switch, each diode has one end at 6 volts and the other end is switched between 12 volts and ground to turn on and off, through resistors to set the DC current

In the receive state, PIN diode D6 is on and D1 is off, connecting the mixer to the MMIC amplifier. At the other end, D1 and D5 are off, and D4 is on, connecting the output of the MMIC to the IF through a fixed attenuator. There is a small net gain between the MMIC amplifier and the attenuator.

In the transmit state, the PIN diode switch must attenuate the RF output from the IF transceiver down to the level required by the mixer, usually less than one milliwatt. PIN diode D1 is on and D4 is off, connecting the IF input to the attenuator. At the other end, D3 is on and D6 is off, connecting the output of the attenuator to the transmit mixer. Finally, D5 is on to protect the receive MMIC amplifier from any transmit leakage.

For the SAFE state, D1 is on and D4 is off, connecting the IF input to the attenuator, but D3 and D6 are both off, so neither side is connected to a mixer. Instead, D2 is on, shorting out the output of the attenuator to ground. This reflects the signal back into the attenuator. However, the total attenuation is more than 25 dB, so the return loss from this reflection is more than 50 dB – absorbing all the power for practical purposes.

The IF interface as shown is for a common TX and RX at both the IF and transverter sides. However, either may be split for separate transmit and receive:

> **Separate mixers for Transmit and Receive –** the TX mixer connects to J3 and the RX mixer to J4. D6 and R16 must be removed, and C13 added.
>
> **Separate IF ports for Transmit and Receive –** the IF transmitter connects to J1 and the IF receiver to J2. D4 and R11 must be removed, and C10 added. Since the receive path does not go through the input attenuator, the MMIC gain may be too high. Consider a MMIC with lower gain.

My sequencers are setup for 2.5 watts in from the FT-817, with about 17 dB of power attenuation capable of dissipating about 5 watts. With the variable attenuator adjusted for about 1 milliwatt out , or about 34 dB attenuation on TX, the loss is greater than 60 dB in the SAFE state, which should prevent any significant unwanted output until the sequencer is finished.

Arduino and Control

The bottom half of the PC board contains the Arduino and control circuits. The inputs are the PTT line, either L (GND to transmit) or H (+V to transmit), an input from the RF sense, and an Amp_RDY pin. I prefer to run the PTT through the IF coax – FT-817 is modified to provide ~+8V on transmit [], and the IC-202 does the same. A jumper on J5 selects which polarity of PTT is coming up the IF coax. The RF sense is used to switch to the SAFE state if RF arrives before the PTT signal. In some simple loaner transverters that might be used with other rigs or even an HT, the RF sense can be used to transmit – just a simple software change in the Arduino.

Outputs from the Arduino drive the TXE and RXD signals that control the PIN diode switch. Other outputs drive FET switches that can handle more voltage and current: P-channel power FET switches to provide +12 volts at the appropriate time, to a preamp and to the receive side of the transverter, or to amplifiers and the transmit side of the transverter. An N-channel FET provides a GND signal to switch a commercial transverter, and an N-channel power FET grounds the bottom end of a coax relay or waveguide switch. The Arduino may be programmed to activate the TR switch on either transmit or receive – your choice. Note that the TR power FET, Q9, is driven directly by the Arduino; if it is a 3.3 volt version, Q9 must be a logic-level FET which can switch at a lower gate voltage.

Those are the basic functions hard-wired on the PC board, but the Arduino has at least a dozen more pins available for additional functions. I am currently using three of them to monitor the 12 volt battery for portable operation. An analog input compares the voltage (reduced by a resistive voltage divider) to an internal 1.1 volt reference. The software calculates the supply voltage and drives a bicolor LED with two pins, displaying Green for good voltage, Yellow for marginal, and Red for low voltage. There is also the option of refusing to transmit if the voltage is too low.

The pinout of the Adafruit Pro Trinket is shown in Figure 5. The pin numbers indicated in blue are usable as inputs or outputs, as defined by the software. The pin numbers may seem arbitrary, but I think they refer to the pins on the actual Arduino chip in the middle of the small PC board. When used as inputs, they are CMOS inputs, so make sure they are not left floating or you may get unexpected results. It is possible to program a weak pull-up resistor. The outputs drive in both directions, so they can sink current to ground, perhaps 20 ma max, or supply enough current to drive an LED through a resistor. The pins that also have an orange indicator, A0 to A5, can also be used as analog inputs, to measure voltage and compare it to an internal reference voltage or an external reference voltage supplied to the ANALOG REF pin. Pins A6 and A7 are unique, and off the standard inline pinout – they may only be used as analog inputs.

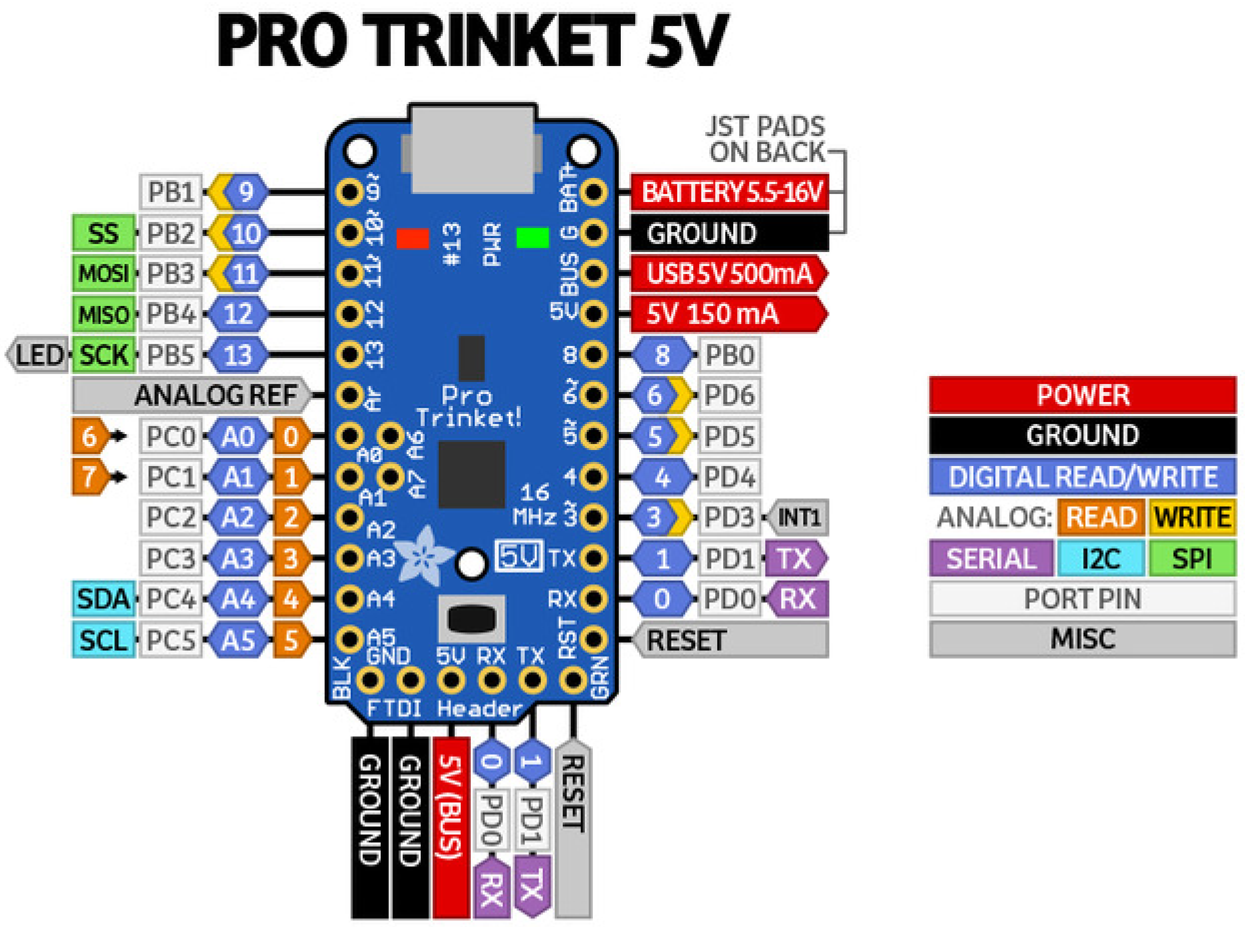


Figure 5 – Pin definitions and locations for the Adafruit Pro Trinket

The 3 volt version of the Adafruit Pro Trinket is identical to the 5 volt one in Figure 4, except the 5V pin provides a 3 volt output and the inputs and outputs are limited to 3 volts. The advantage of the 3 volt version is that it can drive other 3 volt chips, like synthesizers directly. Otherwise, additional resistors, R27 thru R32, are required to limit the voltage – the PCB has pads for these resistors, but the pads for the series resistors have shorting bars which must be cut.

The Arduino provides the smarts for the sequencer – or as much as the programmer provides. It comes in many versions. After some research, I settled on the Adafruit Pro Trinket as providing a good combination of capability, price, and easy programming. Just a matter of installing the IDE on your PC, adding drivers for the Adafruit Pro Trinket, adding the code, and connecting a USB cable. Unless you've done it before, read the book[4] by W8TEE & W6DQ and try a few examples.

NOTE: One gotcha with the Adafruit Pro Trinket that differs from some other Arduinos: you must press the RESET button before programming, and upload the code within 10 seconds. Sometimes compiling takes more than ten seconds, so just press the RESET button again.

Once the code is loaded, it is stored and starts again on power-up.

I solder pins to the Adafruit Pro Trinket and socket pins on the PC board, so I can take one out and swap one in with a different program, or test a modified program before putting it in a transverter.

Program = Arduino Sketch

Rather than printing the Arduino Sketch for the sequencer here, I will make several versions available on my website, www.w1ghz.org, adding more with different options when they are working. Of course, you are welcome to make improvements and urged to do so and to share them! The base sequencer takes about 2K of onboard memory out of around 28K available, so there is plenty of room for options and experiments.

The sequencer is a state machine – it moves sequentially thru a series of states, moving when specified conditions are satisfied, and taking specified actions in each state. A state diagram of the basic sequencer is shown in Figure 6. The diagram is not necessary, but it helps to visualize what is planned and to see that the coding is doing what is planned.

The state machine can be expanded as necessary. For instance, in a version that operates with RF_sensing, I add an extra state that waits long enough for CW characters to be sent without doing any switching or state change. This makes changeover a bit slower, but I never operate full breakin anyway.

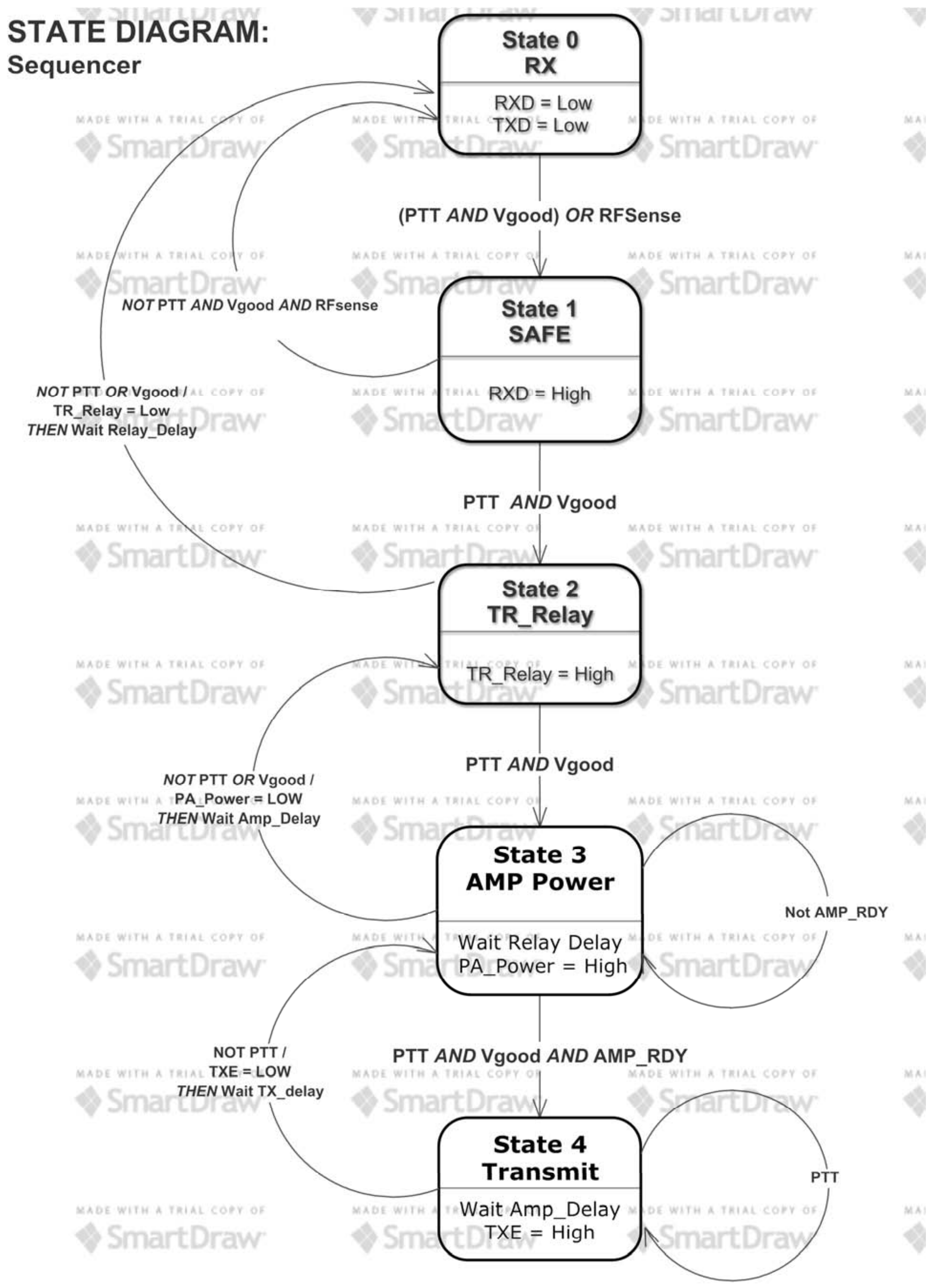


Figure 6 – State Diagram for Sequencer Mark4 without options

Options

Some of the other options suggested above can take advantage of the unused Arduino pins. A real advantage is that new ideas can be added easily without having to modify hardware. Here is how some options might be implemented:

- **TR relay auxiliary contacts –** some coax and waveguide switches have auxiliary contacts so that successful switching may be detected. By connecting one contact to a voltage through a resistor and the other contact to ground, contact closure can be detected as a change in input logic state from High to Low.
- **Latching relay driver –** this might take adding an additional power FET to drive the latching relay in the other direction, but some latching relays have internal drivers and only require TTL levels (5 volt logic). Some code will be needed to do the switching correctly, and to make sure the relay initializes in the right direction.
- **Amplifier voltage –** higher power solid-state amplifiers often operate at 28 or 48 volts. A voltage divider can reduce the voltage to be measured by an analog input, to be sure that the amplifier is ready.
- **VSWR monitor and shutdown –** a directional coupler with an IC power detector on each leg can monitor both forward and reflected power. The detector outputs would go to analog inputs and shut down the RF power if VSWR is high. Many of the detectors have outputs that are linear in dB, making Return Loss calculation easy and constant with power, so problems can be detected at lower power. My plan is to use the analog-only inputs A6 and A7 for VSWR monitoring.
- **Temperature monitor** (external sensor) – an IC temperature sensor, like an LM34, has an output voltage directly proportional to temperature, and could be put near the final transistors to monitor amplifier temperature. The voltage would go to an analog input.
- **Transceiver inhibit –** some transceivers, like the FT-817, have an inhibit pin which prevents RF ouput. In the FT-817, the output is inhibited if the inhibit pin on the ACC connector is at less than 1.0 volts. While this pin could be driven directly from the Arduino, I would use an external FET for isolation. For fool-resistance, the SAFE state provides protection if this connection should come loose.
- **Bidirectional Amplifier –** some simple microwave transverters use a single amplifier on both transmit and receive, with the direction reversed by a 4-port transfer relay. We would like the amplifier powered for both transmit and receive, but not powered while the relay is switching. A simple matter of software.
- **Remote PIN switch –** if, for some reason, you wish to separate the PIN switch from the control section, the PC board may be cut along the silk-screen line next to the RXD and TXE pads. A few wires are needed to make the needed connections: **RXD**, **TXE**, **+12V**, **+6V**, and possibly **PTT**, if it is coming up the IF coax. The only problem is that this breaks up the RF sensing circuit.
- **Multiband transmit protection –** some compact antenna stacks, particularly in rover installations, have significant power getting into antennas for different bands. One solution is to switch the TR switch on all bands to transmit when any band is transmitting. This can be done with just one wire connecting all rigs together, as long as all use similar sequencers, using two Arduino pins, one output pin to drive a FET to pull the wire Low when this band is transmitting, and one input pin to detect when any band is transmitting. A bit of code can sort out the difference and prevent conflicts. The wiring is sketched in Figure 7.

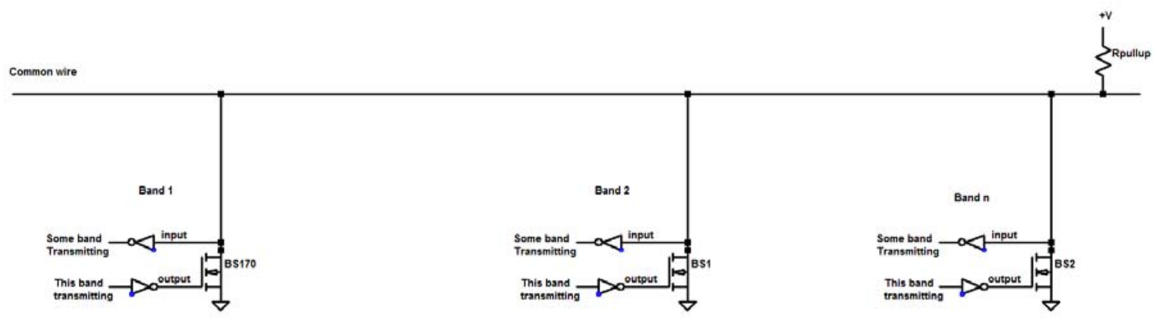


Figure 7 – Wiring multiple sequencers for Multiband Transmit Protection

Construction

I start by wiring and testing the PIN diode switch, starting with the surface-mount components on the bottom of the board. To keep things simple, all the capacitor footprints with no reference designator (Cxx) are the nominal 1800 pf – I have a reel of these, so I was using them. I since found a reel of 1000 pf and started using them, since the value isn't critical – anything between perhaps 470 pf and 2000 pf is fine. Using a small temperature-controlled iron, I solder in these chip capacitors, then go on to the others, which may be a different value or an optional location, not always used. Next are the tiny PIN diodes, aligning the end with the stripe to the PCB pad with the dot, followed by the MMIC, A1. Finally, the resistors are added on the top side. The voltage regulator, U1, is installed to provide +6 volts.

To test the PIN diode switch, apply +12 volts (nominal) to the Powerpole pads and connect clip leads to the TXE and RXD pads. TXE and RXD must be connected to either GND (Low) or +6V (High), not left floating. With both connected to +6V, apply some 144 MHz (or other band) RF at the IF connection J1 and look for output at the XVTR connection, J4. The output at J4 should be around 30 dB down, and adjustable with the trimpot, R6. Now connect TXE to GND, and the output should drop to about 60 dB down. If OK so far, connect both TXE and RXD to GND, and reverse the RF connections to test the receive path. Limit power to less than 1 milliwatt (0 dBM) to avoid overdriving the MMIC.

Troubleshooting: two of the seven sequencers I've built to date did not work as described. One had a PIN diode installed backward, and the other had a chip capacitor with one end that had not been soldered. I've also installed MMICs backwards, but not on this project. Work carefully and inspect with a magnifier, and you shouldn't have many problems.

The rest of the construction is big parts with leads, easily soldered. The pins on the Adafruit Pro Trinket can be aligned for soldering by putting them through the holes in the PC board, then the sockets aligned with the Pro Trinket pins.

Any wires from the sequencer which leave the enclosure or travel near RF should have feedthrough capacitors and Ferrite beads.

Arduino Again

Everything else depends on the sketch code loaded into the Arduino. If you've played with Arduino or PIC chips before, this is trivial. Otherwise, the first time brings some trepidation. Attach a USB cable and give it a go – you can't hurt anything. If you have a local Makerspace or teenage geek, they can probably help if necessary.

To get started, try the Blink sketch found in the Arduino IDE (File -> Examples -> Basics -> Blink). This sketch just turns the red light on the Arduino on and off repeatedly. Upload, and if it works, modify the sketch to blink a Morse character like "V" and upload again. If that works, you are good to go.

A Note on Preamps

For some time, the conventional wisdom for preamplifiers and LNAs is that leaving them powered at all times makes them less susceptible to failure. However, at EME2016, G4DDK[7] presented results of real testing he has done. His conclusion is that "keeping power on during transmit does not confer any advantages." His suspicion is failure is due to relay isolation. I have seen that isolation in some relays and waveguide switches is much lower while switching, a problem addressed by proper sequencing.

Sequencer for Power Amplifiers

Another use for sequencers is for high power amplifiers, like the VHF and UHF LDMOS amplifiers. These operate at higher input powers than the PIN diode switch can handle, but still need sequencing and control for successful operation. Figure 8 shows a prototype of a sequencer board without the PIN diode switch section.

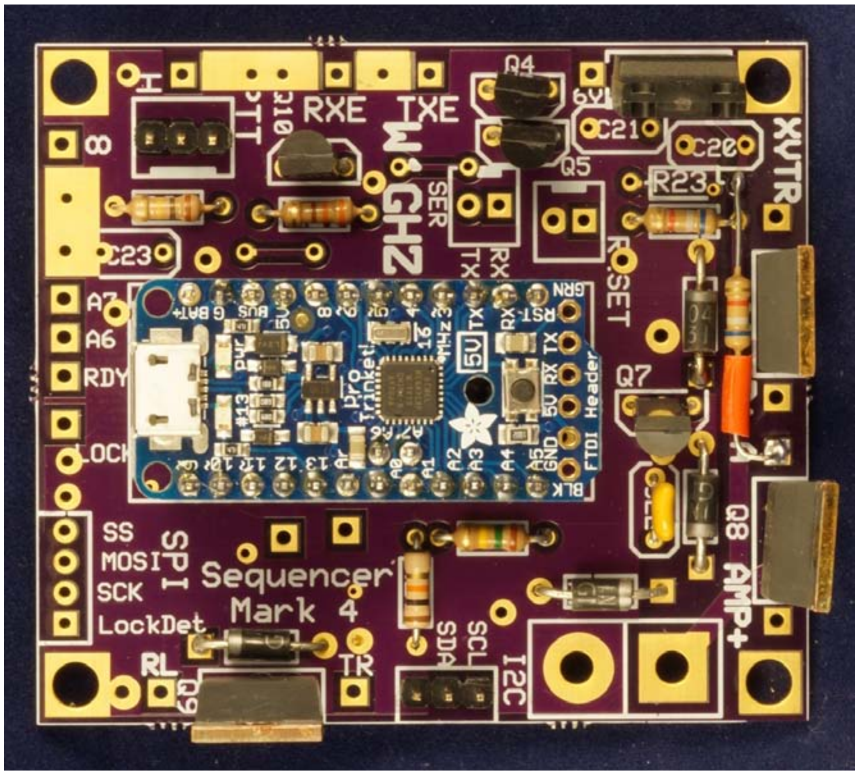


Figure 8 – Sequencer Mark4 for Amplifier Control

Summary

This sequencer provides a combination of fool-resistance, flexibility, and reasonably easy construction. PC boards are available, and all the other parts are readily available. I hope that it prevents a few disasters and enables more contacts.

Putting the smarts in an Arduino makes it easy to program and modify as needed. This does add the possibility that the fool may be the programmer rather than the operator. For programming, in addition to the book[4] by Jack Purdum, W8TEE, and Dennis Kidder, W6DQ, extensive documentation and numerous examples are available online.

References

1. Marshall Williams, K5QE, "The Dark Side of Sequencing," *Proceedings of the 2016 VHF Super Conference,* ARRL, 2016.
2. Paul Wade, W1GHZ, "Even More Fool-resistant Conditional Sequencer," [MUD2009?]. also www.w1ghz.org
3. www.arduino.cc
4. Jack Purdum, W8TEE, and Dennis Kidder, W6DQ, *Arduino Projects for Amateur Radio*, McGraw-Hill, 2015, ISBN 978-0-07-183405-6.
5. www.adafruit.com
6. http://www.sm5bsz.com/dynrange/eme2004/ft817_1e270433_144cwk500mw.gifor http://www.sm5bsz.com/dynrange/dubus205/dubus205.htm
7. Sam Jewell, G4DDK, "*Failure Levels in LNAs*," EME2016 Conference Proceedings, Venice, 2016, pp. 19-26, ISBN 979-12-200-1222-5.

Parts

Component values and part numbers are shown in the schematic, Figure 3. Most of the parts are ordinary and should be readily available, but a few might need better definition:

- Arduino – Adafruit Pro Trinket from www.adafruit.com, either 5V or 3.3V version.
- Socket for Adafruit Pro Trinket – Feather Stacking Header Kit P2830
- PIN diodes – I have used BAP50-03 (Digikey 568-1916-1-ND), BAR63-03W (Digikey BAR63-03WE6327INCT-ND), and BAP63-02 (Digikey 568-1926-1-ND). All are inexpensive and stocked in quantity. I'm sure many others will work fine, even the old MPN-3401 if you can find them.
- P-channel power FET (Q6, Q8) – the IRF9Z34 shown is good, any other that handles amplifer current is fine.
- N-channel power FET (Q9) – the IRF9410 shown is good, or any other that can handle relay voltage. For 3V Adafruit Pro Trinket, a logic-level FET is required; the IRLU3110 works.

Errata

There is one error in the intial lot of PC boards, marked "W1GHZ 2017" on the bottom side. The bottom end of R23 should not be connected to the hole near Q5, but rather to the pad marked "A" near Q8, so that R23 is connected between the +12V rail and pad "A". Modified placement can be seen in Figure 2.