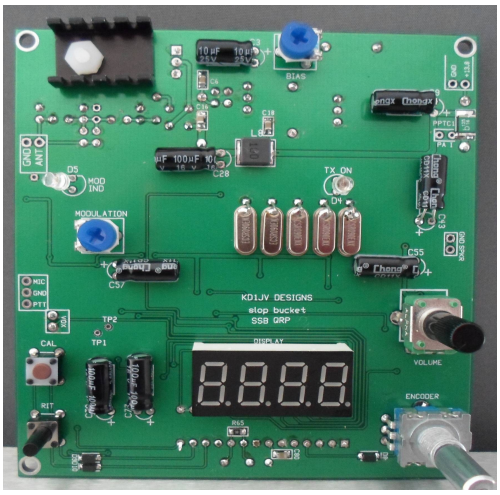
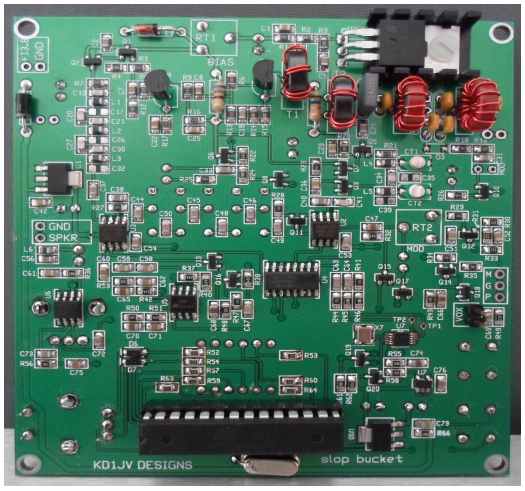
THE SLOP BUCKET

A 7 WATT QRP SSB/CW/Digital TRANSCEIVER on a 4" x 4" board

By KD1JV Designs





Operation:

Controls:

Volume : Controls the speaker volume. There is sufficient power to drive a small speaker to a reasonable volume. AGC is applied before the volume control.

Tuning : Tune the frequency at the selected tuning rate. Default – 100 Hz.

Tuning Range: Tuning outside the band is not limited. When the frequency is tuned outside the band, <14.000 MHz, >14.350 MHz, the display will shift to indicate the MHz digit(s) and the transmitter keying is disabled. This allows listening to SWBC stations. If you try to transmit while out of band, the display will show four "dashes" indicating transmission is disabled.

Encoder Push button : Press to increment tuning rate, 100 Hz, 500 Hz, 1 kHz, 10 kHz and 100 kHz rate. Selected decade digit will blink off when selected. 100's digit blinks when rolling over from 100 kHz to 100 Hz rate, then again when incrementing from 100 to 500 Hz rate.

Mode: Timed Push button.

Short push: Toggle RIT on and off. When enabled, display will show the difference between your transmit frequency which is now fixed and the receive frequency, +/- 09.99 kHz.

Long push : Toggle CW mode on and off. Display will indicate [CodE] when the switch has timed out. Release the switch. Display will read [C on] or [C of] to indicate CW mode status.

Microphone and push to talk:

These connections are done with a 3.5mm phone jack. I wire tip as PTT so that a straight key with mono plug can be used to key the rig in CW mode. The microphone input circuit is active during CW transmit, so grounding the mic input with the mono plug is recommended in stead of letting it float.

The rig is designed to be used with an Electret type microphone. Power is supplied via a 22K resistor to 5 volts and should power most mic elements. www.qrpkits.com sells a little microphone kit which works nicely with this rig.

CW mode:

CW shifts the BFO frequency by 600 Hz during transmit to move it into the pass band of the IF crystal filter and the LO frequency is shifte by 600 Hz to compensate and ensure you are transmitting and receiving on the same frequency. The product detector is unbalanced by connecting a resistor to one of the inputs to ground to allow the BFO signal to be outputted by the mixer instead of being suppressed. This also matches the transmit frequency to that of the other station. The CW waveform has a decent rise/fall time to eliminate over the air key clicks.

Modulation indicator LED:

A red lead is used to indicate modulation. In CW mode it should shin brightly during transmit. In SSB mode, it should flicker on as you talk. Power output has to exceed about 3 watts before the LED starts to come on.

VOX:

A simple VOX circuit can be enabled with a jumper. It takes a somewhat higher audio level to trigger the VOX then a normal speaking voice level would be. Therefore, if VOX is used, the modulation control needs to be reduced to prevent over modulation and resulting problems. The VOX mode is added to facilitate connecting to a sound card for digital modes without the need for some T/R signal from the computer, which most lack these days.

Setting modulation level for digital modes.

1. Turn the modulation level control to minimum. Fully CCW if mounted on bottom of board.
2. Place the modem into transmit mode and adjust the output sound level until the VOX kicks in and the transmit mode is enabled, as indicated by the TX on LED.
3. If the idle tone is a two tone, adjust the Modulation control until the Modulation LED starts to flash on brightly.

Power supply and output power:

The Slop bucket is designed to run at about 13.8 Volts. At this voltage power output will be about 7 watts PEP.

Additional heat sinking of the PA:

Just enough heat sinking is suppled for low duty cycle operation of the PA. Use of digital modes will likely cause excessive heating of the PA, which causes a reduction in power output and possible failure. Casual use with short transmissions and long receive pauses would not be a problem. If digital modes are to be extensively used or if your a long winded rag chewer, additional heat sinking is recommended.

Specifications:

Receiver current: ~54 ma at 13.8V
Sensitivity: ~ 0.3 uV.
Audio frequency response 300 Hz to 2600 Hz. (kind of narrow)

Transmit current ~1.2A at 13.8V, 7.3 watts out.
Power output ~6.4 to 7.4 watts at 13.8V Power peaks at low end of band.
Operating Voltage 11 to 14 volts. Below 11V, the PA bias starts to go out of regulation. The rig will still operate down to about 10 volts with a few watts output, but CW operation only is recommended at this low voltage.

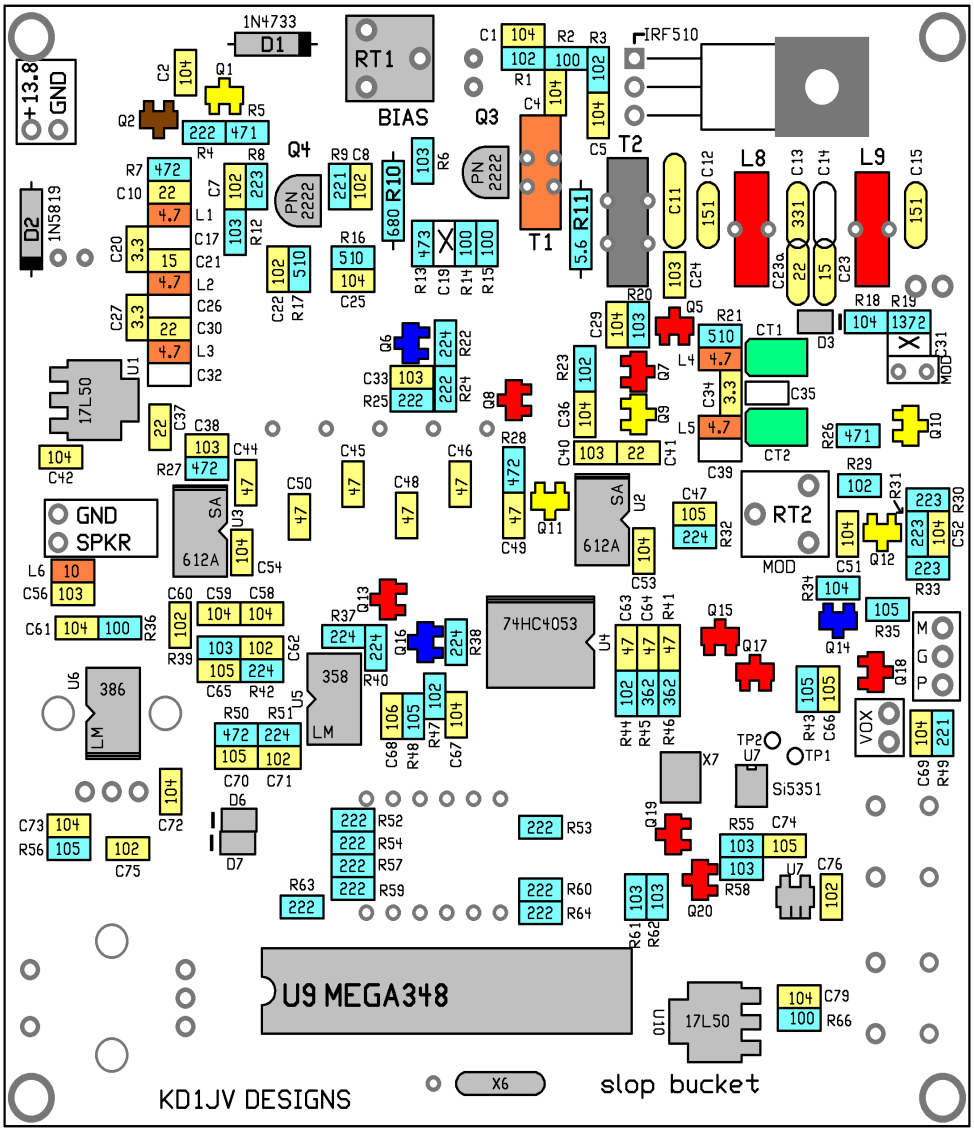
With A-B comparisons with the Ten-Tec Argo V, the “Slop bucket” isn't quite as sensitive, but is pretty close hears everything the Argo does. Internal noise is much less on the Slop Bucket then the Argo, and that helps to copy the weaker stations.

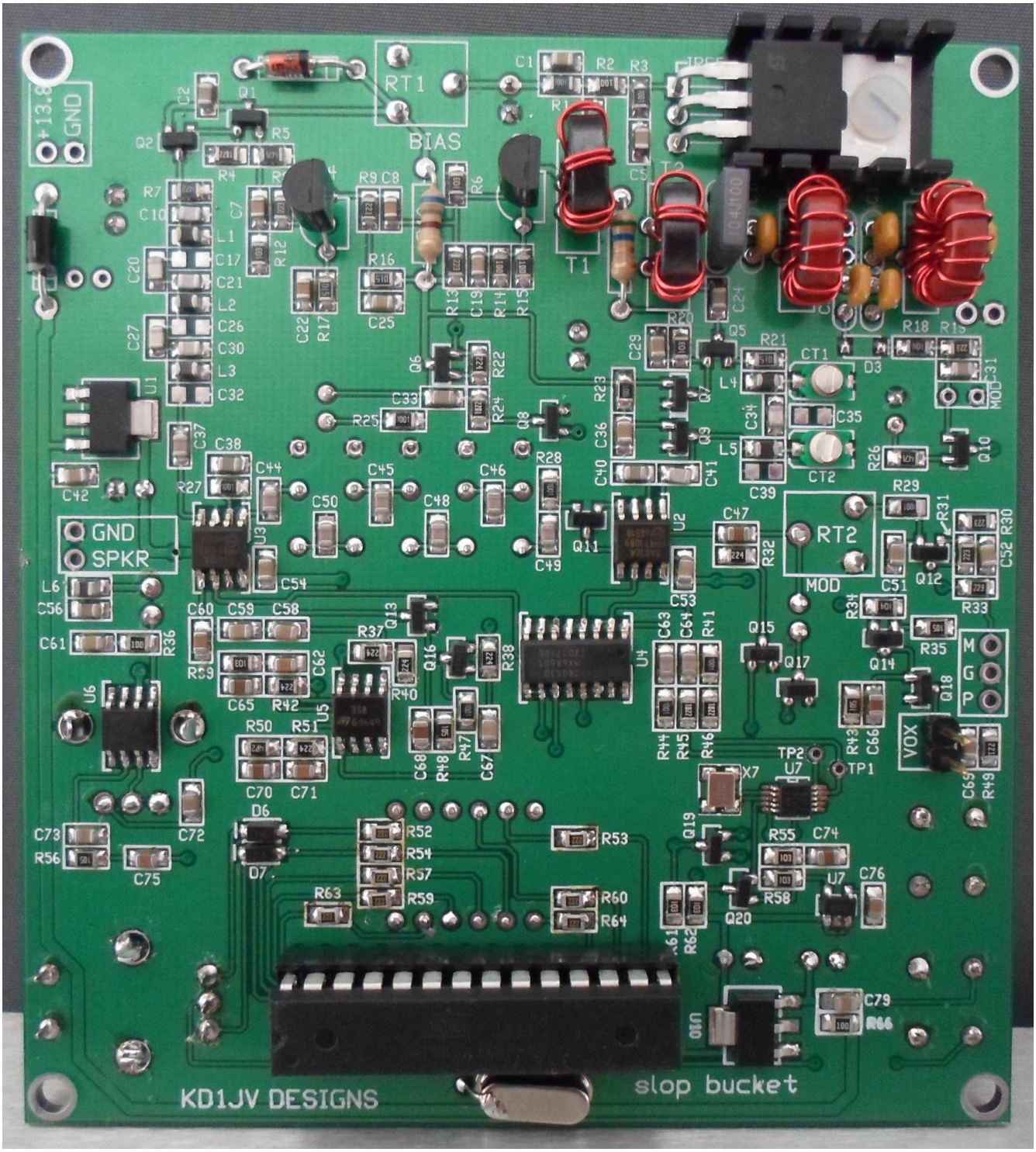
PARTS LIST

	qty	Resistors 0805		qty	Caps 0805		qty	semiconductors
	5	10 ohms [100]		3	3.3 pfd [ORG/BLK]		2	PN2222A NPN TO-92
	3	51 ohms [510]		1	15 pfd [BRN/GRN]		5	MMBT3904 [YEL] NPN SOT-23 [1AM]
	2	220 ohms [221]		4	22 pfd [RED/RED]		3	MMBT3906 [BLUE] PNP SOT-23 [K3N]
	2	470 ohms [471]		9	47 pfd [YEL/VOL]		9	2N7002 [RED] MOSFET SOT-23 [K7K]
	6	1K [1001 or 102]		8	1 nf 102 [BRN/BLK/RED]		1	MMBF4416 [BRN] JEFT SOT-23 [6A]
	11	2.2K [2201 or 222]		5	10 nf 103 [BRN/BLK/ORG]		6	1N4148 SS DIODE
	2	3.6K [362]						
	4	4.7K [472]		24	100 nf 104 [BRN/BLK/YEL]		1	1N4733 1W 5.1V ZENER AXIAL
	8	10K [103]		5	1 ufd 105 [BRN/BLK/GRN]		1	1N5819 1A/40V SHOTKEY AXIAL
	1	13.7K [1372]		1	10 ufd 106 [BRN/BLK/BLUE]			
	4	22K [223]					1	LM358 DUAL OP AMP SO-8
	1	47K [473]		1	10 ufd/25V electrolytic		1	LM386 AUDIO AMP SO-8
	2	100K [104]		7	100 ufd/16V electrolytic		1	74HC4053 ANALOG SWITCH SO-16
	7	220K [224]		2	150 pfd [151] C0G LEADED		1	Si5351A CLOCK GENERATOR
	6	1Meg [105]		1	330 pfd [331] C0G LEADED		1	MEGA328 PROGRAMED DIP
	1	5.6 OHMS 1/4W		1	15 pfd [15] C0G LEADED		2	NCP1117LPST50T3G (5V REG)
	1	680 OHMS 1/4W		1	22 pfd [22] C0G LEADED		1	TC1014-3.3VCT713 (3.3V REG)
	2	2K TRIMMERS			CRYSTALS		1	IRF510B TO-220 MOSFET, PWR
	1	50K 9mm POT		1	25.000 MHz SMT		1	4 DIGIT SEVEN SEG LED DISPLAY
				1	16.00 MHz HC-49U			
		INDUCTORS		5	9.00 MHZ HC-49U		1	0.001 uf [102] leaded disk cap
	5	4.7 uhy 0805			MECHANICAL			
	1	10 uhy 0805		1	6mm / 12 mm TACT SW			
	1	10 uhy 6028 PWR		1	12mm ENCODER			
	2	T37-2 TOROID		1	28 PIN DIP SOCKET			
	2	FT37-43 TOROID		2	3.5mm PHONE JACK			
				1	2.1mm POWER JACK			
				1	BNC PANEL JACK			
				2	HEAT SINK			
				1	NYLON SCREW+NUT			
				1	5 FEET #28 MAGNET WIRE			
				1	Red filter			

Overall parts location map. Print this page for reference while placing parts.

Capacitors – yellow
Resistors – blue
inductors – orange
all others – gray
SOT-23 package location matches the part color coding strip.




+13.8
GND
RT1
BIAS
T1
T2
GND
SPKR
RT2
MOD
VOX
M
G
P
KD1JV DESIGNS
slop bucket

Assembly:

If you bought this kit, you should already be experienced with using solder paste and have the supplies and tools needed. However, if you haven't used solder paste before, there are hours of “how to” videos on YouTube which you can watch and learn. It's really not that complicated. Some very thin wire solder is supplied for touch up and the few top side SMT parts which are easier done by hand soldering.

So,

1. Place small dabs of paste on all the pads.
2. Place the parts.
3. Melt the solder with the hot air embossing gun.
4. Check for touch up work - missing connections, shorts between IC pins.
5. Place the through hole parts.

It really doesn't matter what order you place the SMT parts. I generally place all the semiconductors first, then the resistors, then the caps, then what ever is left over. So, that's the order the assembly diagrams are presented. If there is more then one of a specific value or type o part, place all of that same value in one pass. This is especially important for capacitors since their value isn't marked on the part. Once out of the carrier, you can't tell them apart without a meter.

Semiconductors

SOT-23 parts are color coded with a sticker which matches the highlighted color in the diagram. Most of the ICs have a dot in the pin 1 corner. The SA612A parts have the Phillips logo in the pin 1 corner. try to get the pins on the Si4351 chip lined up with the pads on the board as close to possible (SEE CAUTION BELOW). U7 is the only part with 5 leads in the SOT-23 type package, so it should be easy to identify.

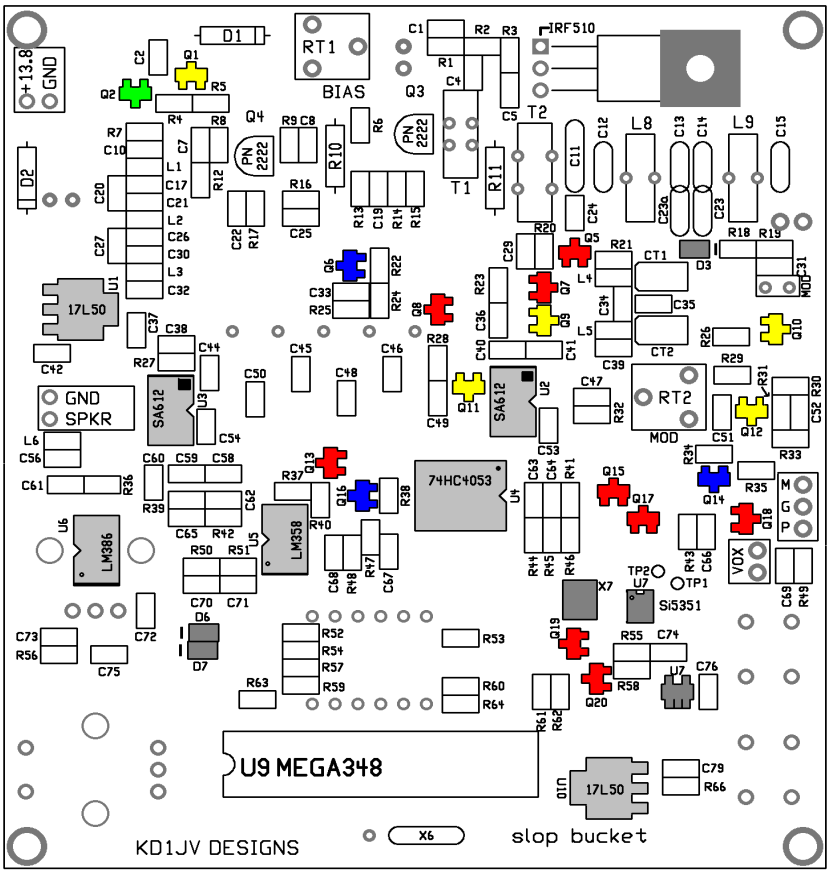
The line marking the cathode end of the D3, D6 and D7 diodes is hard to see. Shinning a LED flashlight at an angle on the part is the best way to see the markings on these parts and the IC's. The cathode end of the diode faces the tractor feed side of the part carrier, so if you carefully peel back the covering, you can pick out the part with your tweezers and keep it facing in the proper direction. Note that only two (2) diode go on this side of the board, so don't open all the slots, just the two you need.

Cautions:

The Si5351 chip mounting needs special attention. Very little solder paste is needed, but it's difficult not to over do it. And you need to see the pads to line up the pins on the chip. You need to get these pretty close for surface tension to do the final alignment. In any event, you will have to clean off the leads with the solder wick suppled to ensure there are no shorts between leads. Place the edge of the solder wick over the leads and lightly press down with the tip of your iron for a second. Avoid any sideways motion while during this.

X7, the Si5351 crystal:

This is the little slightly rectangular silver bit. The connections are symmetrical, active on opposite corners, so the 180 degree orientation doesn't matter. What matters is that the long side is parallel to the Si53451 chip.



Green – Q2 – [6A] jfet MMBF4416

Yellow – Q1/9/10/11/12 – MMBT3904 [1AM]

Blue – Q6/14/16 – MMBT3906 [K3N]

Red – Q5/7/8/13/15/17/18/19/20 [K7K] 2N7002

U1, U10 [17L50] SOP-223 regulator

U2, U3 [SA612A] SO-8 mixer

U4 [74HC4053] SO-16 switch

U5 [LM358] SO-8 op amp

U6 [LM386] SO-8 audio amp

U7 [Si5351] TSOP-8 pll clock

Second U7 – SOT-23-5 [A5CE] 3.3V reg

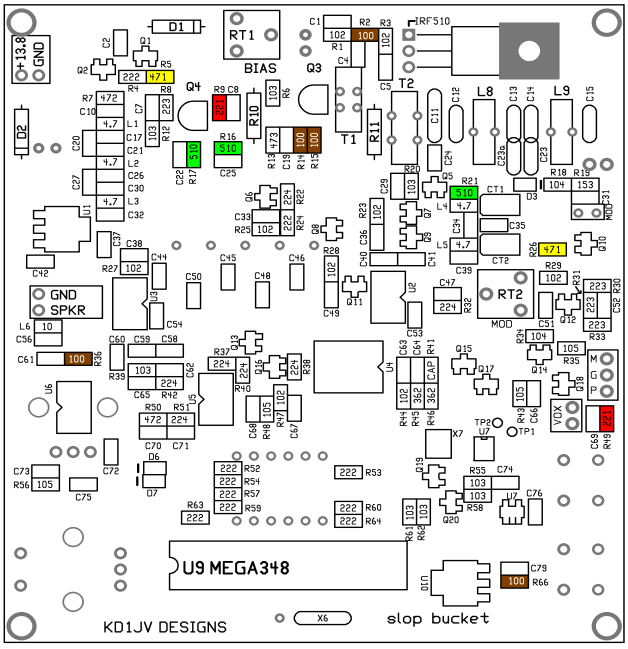
X7 – 25 MHz crystal.

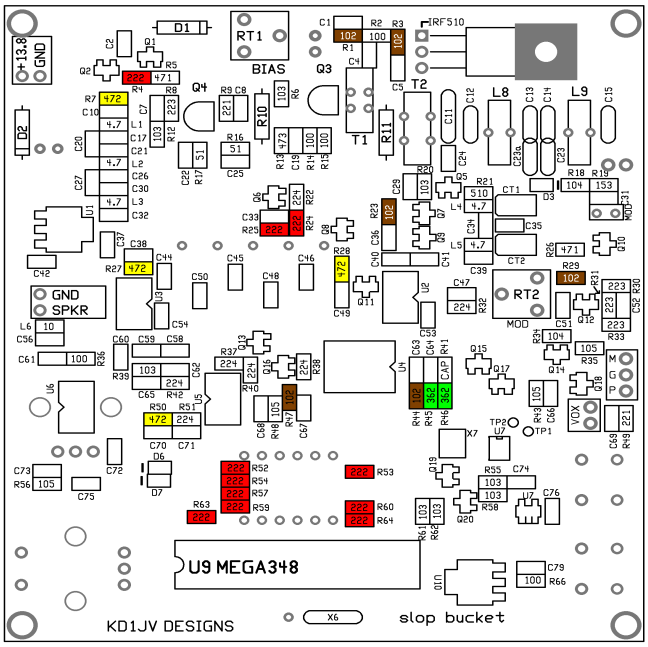
D3, D6, D7 Diodes – look carefully for the line marking the cathode on the part and on the board. The cathode side of the package faces the sprocket end of the carrier, so if your careful, you can go by that. Shining a light at an angle helps to make the markings more visible.

You will have three (3) diodes left over, these go on the top side of the board. Place these aside in a safe location for now.

Resistor locations by values shown by decade

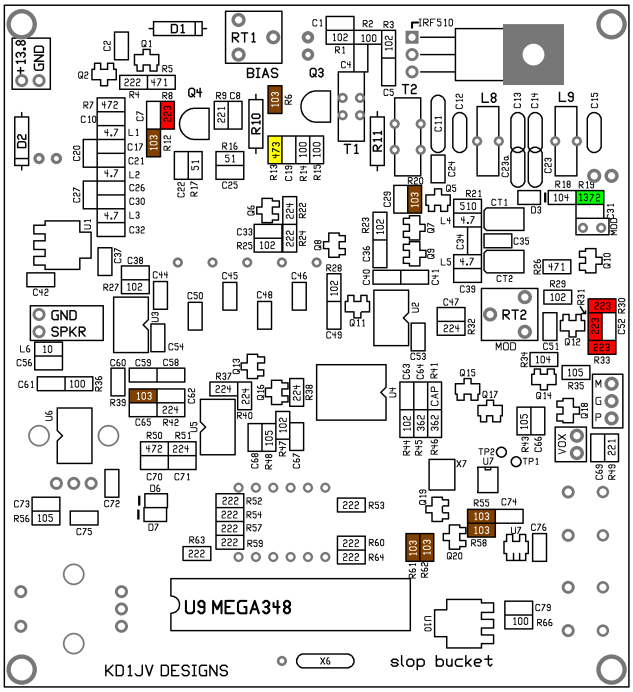
R19 and R25 had last minute value change so the new value isn't shown on layouts which don't have those parts highlighted.

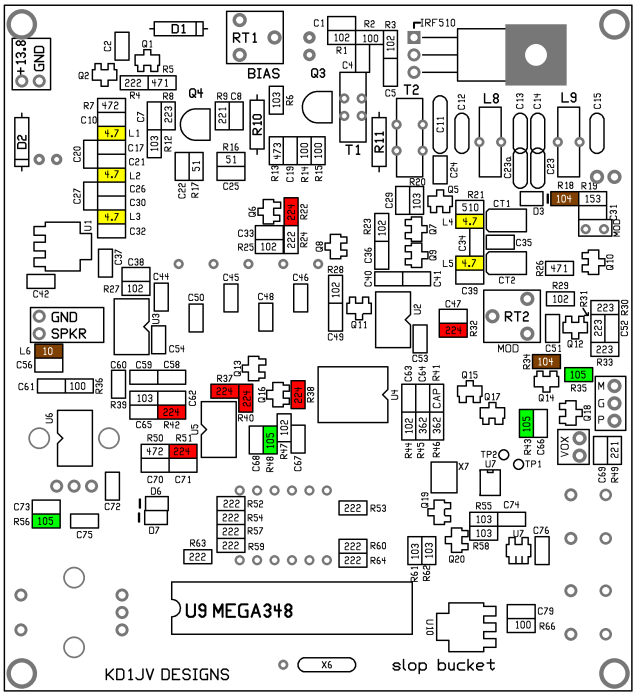




Brown [100] R2,14,15,36,66 Green [510] R16, R17, R21
Red [221] R9, R49 Yellow [471] R5, R26

Brown [102 or 1001] R1,3,25,29,44,47
Red [222 or 2201] R4,23,24,52,53,54,57,59,60,63,64
Green [362] R45, R46 Yellow [472] R7,27,28,50



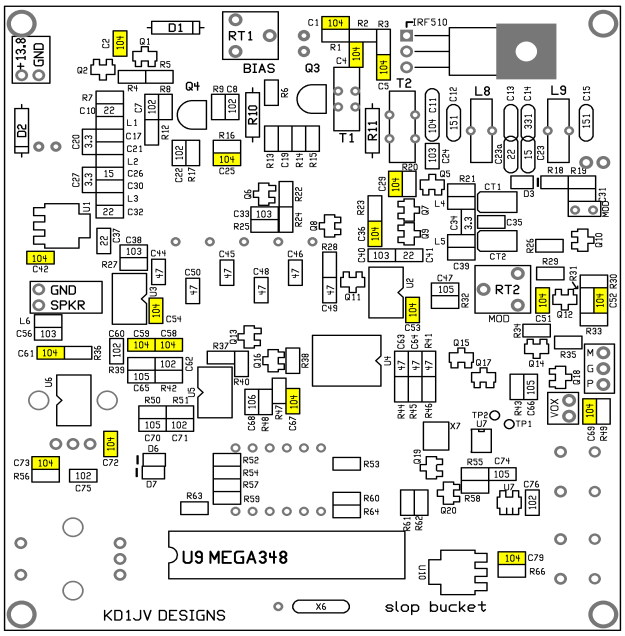


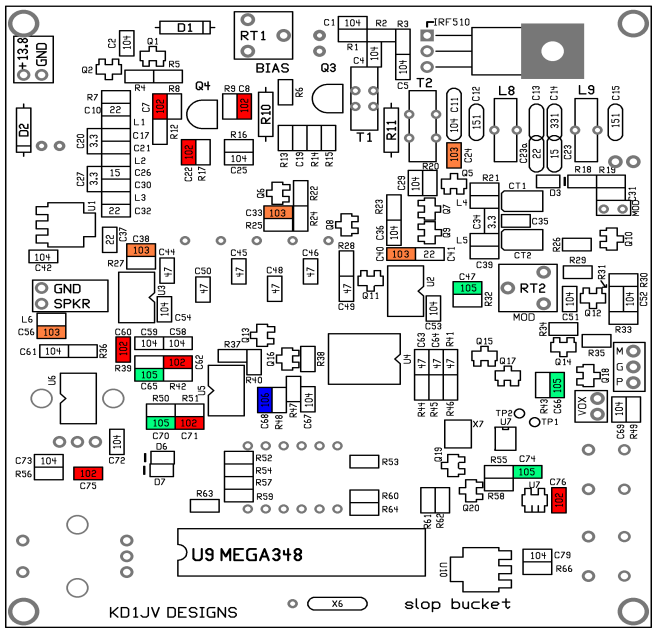
Brown [103] R6,10,20,39,55,58,61,62 Red [223] R8, 30, 31, 33
Green [1372] R19 Yellow [473] R13

Brown [104] R18,34 Red [224] R22,32,37,38,40,42,51
Green [105] R35,43,48,56 Yellow [Inductor] Brown [Inductor]

- The inductors are the small black cubes which will be the only parts from the resistor bag left unused when all the resistors values have been placed.
- There will be one 105 resistor left over, this goes on the top side of the board. Put it in a safe place for now so it doesn't get lost.

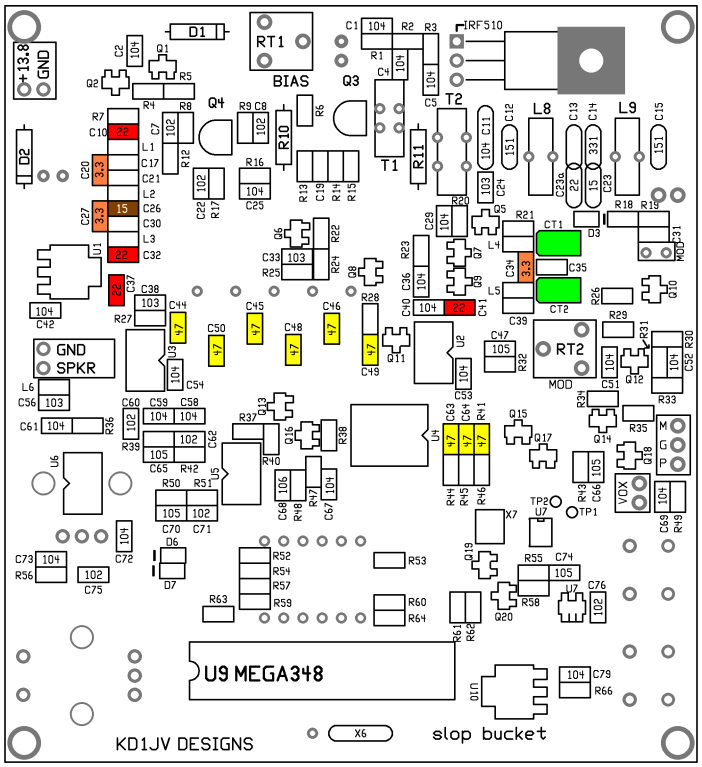
Capacitors location by values.
Capacitors carriers are color coded with the part value, using resistor color coding. Therefor a 104 (0.1 ufd) cap is coded BRN/BLK/YEL The color highlighted in the diagrams represent the multiplier band for values 102 to 107 and the most significant digit for 3.3 to 47 pfd caps.





Yellow [104] C1,2,4,5,25,29,36,42,51,52,53,54,58,59,61, C67,69,72,73,79
There will be four (4) 104 caps left over, these go on the top side later.

Red [102] C7,8,22,60,62,71,73,76 (BRN/BLK/RED)
Orange [103] C24,33,38,40,56 (BRN/BLK/ORG)
Green [105] C47,65,66,70,74 Blue [106] C68



Orange [3.3 – ORG/BLK] C20,27,34 Brown [15 – BRN/GRN] C26
Red [22 – RED/RED] C10,32,37,41
Yellow [47 – YEL/VIO] C44,45,46,48,49,50,63,64 and **R41**
Place the two (2) trimmer caps CT1 and CT2. Note the cut corner end.

The C19, C17, C21, C30, C31 locations are not used.

- **All the bottom side SMT parts should now be placed. Heat the board and melt the solder paste.**

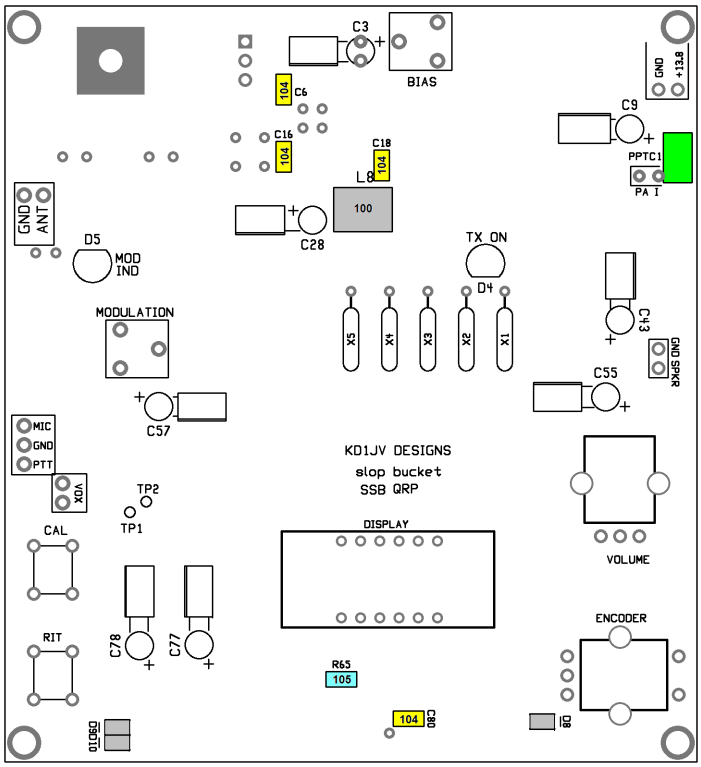
Top side SMT parts:

There are only a few of these parts on this side of the board, so it's easiest just to hand solder them in place.

104 caps – 4 places
105 resistor -1 place
diodes – 3 places

The power inductor and PPTC fuse are packaged with the through hole parts.

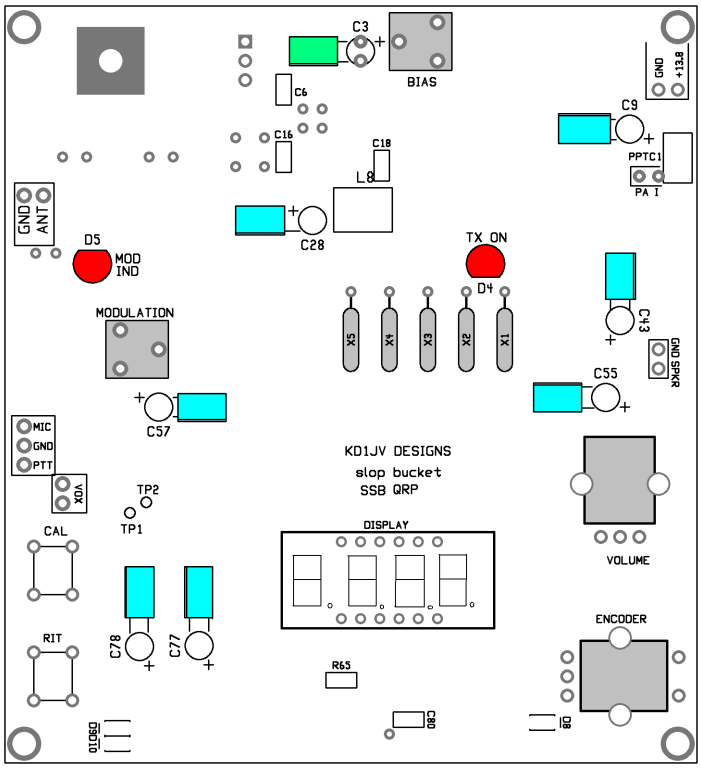
Inductor – L8 one place.
PPTC fuse – Markings [b 125 over 16]



Through hole parts:

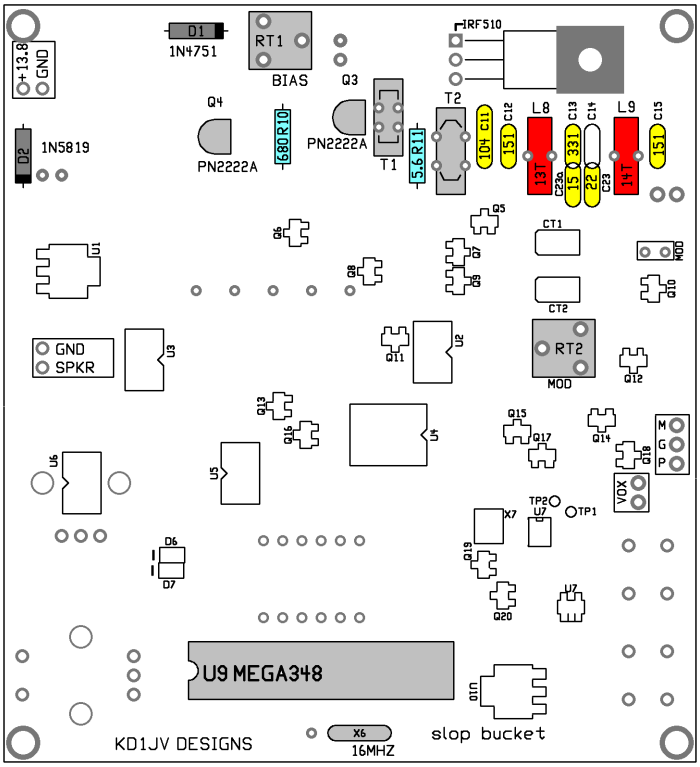
Top side:

- C3 is a 10 ufd cap, all other electrolytics are 100 ufd. Long lead is plus. Lay caps down flat to board as shown.
- The bias and modulation trimmers can go on either the top or bottom side of the board. Which side you put them on really depends on how you will package the board and if it's easier to access the top or the bottom of the board. The trimmers will increase in the CCW direction if mounted on the top and CW if mounted on the bottom.
- You'll want to space the height of the LED's so they stick out a little from the top of the enclosure. You might want to hold off soldering these in place until you decide on a box.
- LED short leg is cathode and goes into pad on flat side of outline.
- Ground the edge of the crystal cases to the solder pad located above it, using lead clippings.
- It's easier to solder the DIP IC socket on the bottom side in place first, then do the display on the top side.
- You also might want to hold off installing the volume control and encoder until after the bottom side TH parts are mounted. That would keep the board from rocking as you work on the bottom.



Bottom side through hole:

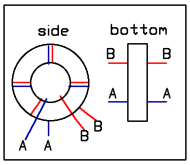
- D1 – 1N4733 NOTE: this diode is missing it's ground connection. Scrape a little of the solder mask off the board and bend the ground side lead over, snip and solder to the board.
- D2 – 1N5819
- R10 – 680 ohm Blue/Gray/Brown
- R11 – 5.6 ohm Green/Blue/Gold/Gold
- Q4 and Q4 – PN2222A
- C12, C15 – 151 MLCC
- C13 – 331 MLCC
- C23 – 22 MLCC
- C32a – 15 MLCC
- C11 – 104 box
- X6 – 16 MHZ crystal – ground case to pad to left.
- 28 pin DIP socket
- RT1 and RT2 trimmers if you want them on this side of the board.



Toroids:

L8 – 13 turns on T37-2 core (red)
L9 – 14 turns on T37-2 core (red)

T1, T2 – 5 turns bifiller. Make a hairpin with 6" of magnet wire and wind five turns. Snip the top of the hairpin off to separate the wire. Tin and use ohm meter to find the common ends of the wires. Arrange wires so common ends are opposite each other.



PA, IRF-510 mounting:

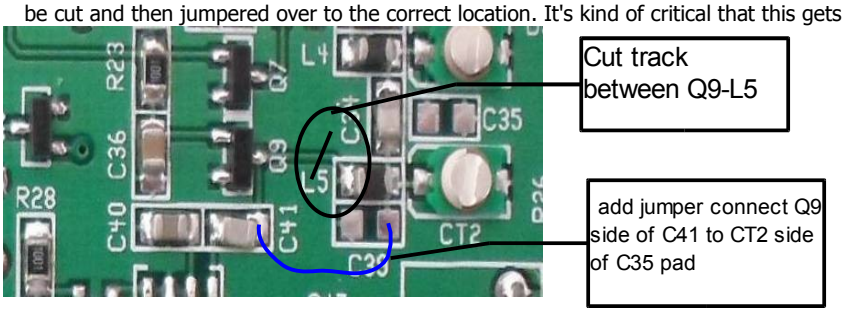
- Put a little dab of the heat sink compound on the big pad, bottom side
- Place heat sink over hole
- more compound, then mica insulator
- More compound, the the IRF-510 MOSFET
- Line up the holes and insert nylon screw
- Put a dab of compound on the pad on the top side of the board
- Place the second heat sink over the screw
- Secure with the nylon nut.

A 2 pin SIP header can be soldered into the VOX enable pads so that a shunt jumper can be used to enable or disable VOX. It can go on either side of the board, but choose the side which will be easier to access when you package up the board.

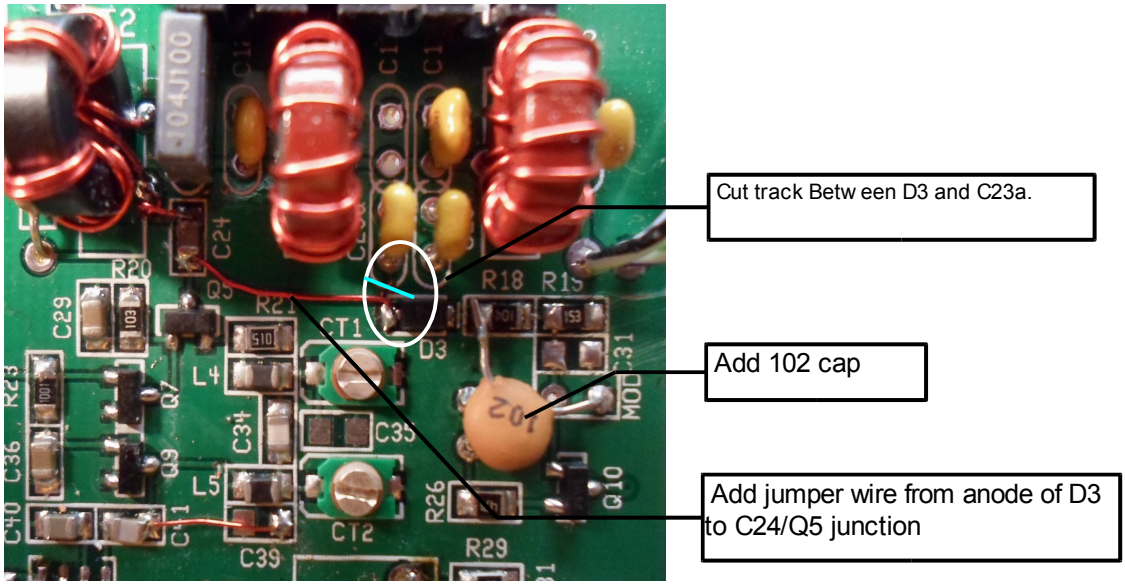
REQUIRED MODS:

Few layout errors on the board need to be corrected.

1) The input to the 1st mixer accidentally got connected to the ground side of L5 instead of the hot side. Therefore the track needs to be cut and then jumpered over to the correct location. It's kind of critical that this gets fixed or your not going to hear anything!



2) In order for the modulation LED to work, a cap needs to be connected between the junction of D3-R18 and ground. The easiest way to fix this is to tack in a leaded part. Also, the detector diode input has to be moved. It worked fine where it is when connected to a 50 ohm dummy load, but not when connected to an antenna or tuner. So, it has to be moved to the junction of C24 and Q5. The connection from the diode to C23a has to be cut.



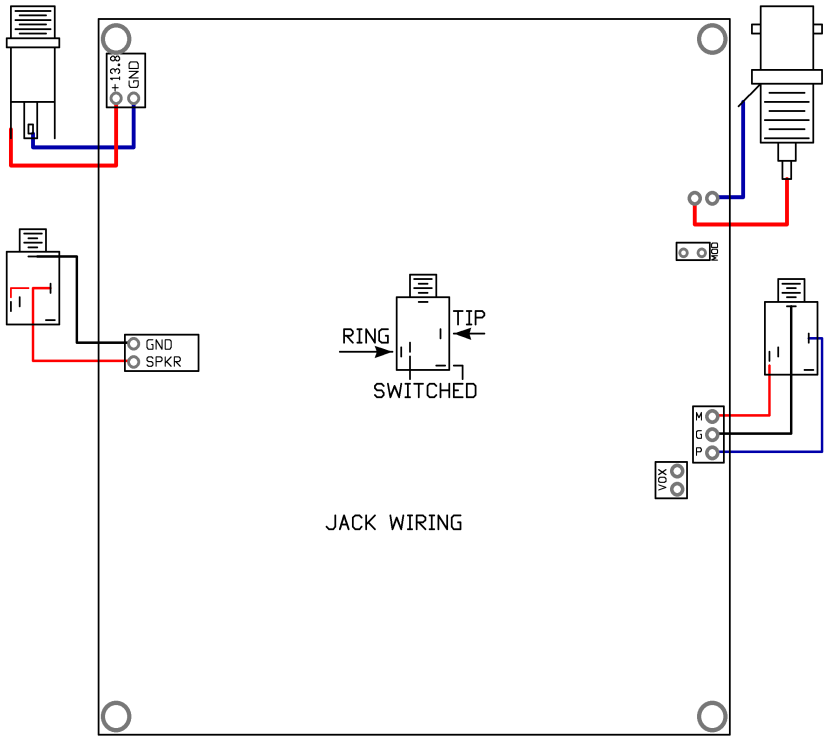
3) Make sure you grounded the end of D1 in the PA bias supply.

Power up and testing:

Before you do anything else, time to give the board a close inspection. The usual cause of any initial trouble is due to missing solder connections or shorts between IC leads. If you can find any now, it saves time later trying to figure out why something doesn't work.

Make sure there aren't any solder shorts on the Si5351 clock chip leads. Use the supplied solder wick the remove any excess solder. Press the solder wick down on the leads with the tip of your soldering iron, but only in a vertical direction. Avoid any side to side motion which might move the leads. Also inspect the 3.3V regulator U7 (lower) for solder shorts. These are generally the only two IC's which you might have this trouble with. Also look for any connections which might be missing solder. You might have missed putting some paste down on a pad or not quite enough.

Now, before you can power up the board and start to test, you need to wire up the various jacks. Until you get the board mounted in a case, you should take a few minutes to make a bracket to mount the jacks on out of a piece of scrap aluminum or sheet metal so their not flopping around and potentially short out to something. Or at least tape up the exposed terminals. Sorry, you need to supply your own wire. If possible, use shielded wire for the Microphone connections.



Initial power up:

Power to the circuits!

Okay, well not quit yet. First make sure the two trimmer resistors are set to min. That would be fully counter clock wise if mounted on the bottom of the board or fully clock wise if on the top side. They come from the factory set fully clock wise. Now you can apply power to the circuits, 12 to 14 volts will do.

- The display should come on and read [bn20] for a second, then [160.0]
- Test the operation of the controls. Tune up and down, change tuning rate, active RIT and CW mode.
- Check output of clock chip. The frequency on TP1 should ideally be 9.000.900 MHz and 23.160.00 MHz on TP2. These frequencies will likely be a little high before calibration.
- Connect speaker or headphones, connect antenna or signal generator.
- Turn up the volume. A small amount of background hiss should be heard.
- Turn the two trimmer caps, CT1/CT2 about a ¼ turn.
- Tune around until you hear a signal.
- Fine tune CT1/CT2 for best signal strength.

You'll probably notice that sideband stations don't tune in on exact 1Khz dial increments. Calibration fixes this and is highly recommended. Without calibration, not only will the frequency read out be off, the BFO frequency will not be quite right.

Calibration:

It is important to verify the proper outputs from the clock chip prior the transmitting. If there is a bad connection to the reference crystal, the VCOs drift to one end of their range or the other, producing a frequency which can burn up the driver if transmit is keyed for more then a few seconds. If you don't have a frequency counter use a general coverage receiver to verify the 9 MHz BFO frequency. If this is okay, the LO should be too.

There is some warm up drift so allow some warm up time. If you don't have a frequency counter, you could use a receiver with good calibration and zero beat the signal. Since the receiver won't have a very low frequency response, you'll have trouble finding the exact zero beat. One way around this to look at the audio output of the receiver with a digital mode program and read the frequency of the beat note off the waterfall. Say you see the signal at 300 Hz. Knowing the encoder tunes in 1 Hz steps, count 300 clicks of the encoder. You might want to make a tick mark on a piece of paper every 10 clicks so you don't lose count!

1. Press the "CAL" button above the MODE switch for at least one second, until [CAL] is displayed.
2. A 10 MHz signal will be outputted at TP1.
3. Using a frequency counter, read the frequency
4. Using the tuning encoder, adjust the frequency to exactly 10,000,000 Hz. Tuning is in 1 Hz steps.
5. Click the MODE switch to store and reboot.

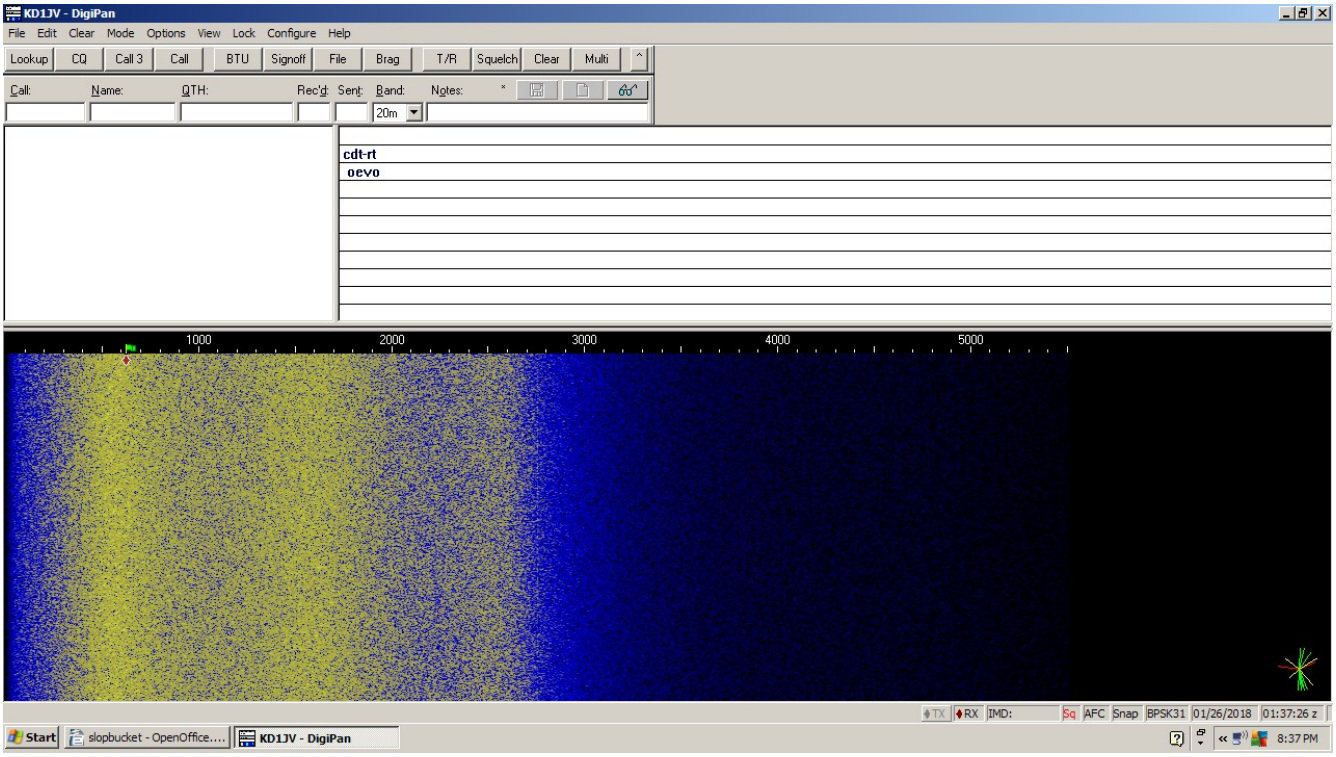
BFO adjust:

Just in case the default BFO frequency doesn't quite line up with the filter due to parts tolerances or your are unable to calibrate the Master clock, the BFO frequency can be tweaked. For best results, look at the audio output with a digital mode program like DigiPan. Ideally, use a wide band white noise source at the antenna to see the full response. Atmospheric noise might be enough for this.

Since only the BFO frequency is changed in this mode, it is possible to use an off air signal and adjust for best sounding voice.

- Press and hold closed the CAL button for longer then 1 second. [bF0] will be displayed.
- Tune up or down with the encoder. Tuning is fixed to 10 Hz steps. The display still shows the Rx frequency, so there is no visual indication something is happening.
- Push the encoder switch to store the new frequency and rest.

Here's a screen shot of the audio output with a noise generator as a signal source at the antenna jack. The audio sounds pretty good with this setting of the BFO.



Transmitter check out:

- Connect an amp meter in series with your supply to the board. Most DVMs have a 10A unfused input, use that.
- Connect up a watt meter and dummy load

Set PA bias current

- Activate PTT The TX on LED should light.
- Verify there is 5.1V or so on the end of D1 to make sure you grounded the anode.
- Note supply current. It should be about 150 ma
- Adjust the PA [BIAS] control while watching the supply current. Set the control to increase the current by about 20 -30 ma. That should be at about the 1 o'clock position. Higher idle currents would improve IMD, but run the risk of instability and larger heat dissipation.

CW test:

- Activate CW mode by holding closed the MODE button for longer then one second until you see [CodE] then [C on] on switch release.
- Activate PTT
- Tx on and Mod LEDs should come on
- Power output should be between 7 and 8 watts with a 13.8V supply.
- Check power output level at the band edges. Power output will likely peak at the lower end of the band.

Voice modulation:

- Exit CW mode by holding the MODE button closed until [CodE] is displayed, then [C of] when switch is released.
- Active PTT and start talking.
- Adjust the MOD trimmer until the MOD LED starts to flicker on. Continue until the LED flashes brightly on voice peaks. If the LED stays bright and doesn't flicker much as you talk, your probably over modulating. For me the setting is about 12 o'clock. Of course, the best way to set the modulation is to look at the RF waveform with a Scope.

Problems?

If something doesn't work, you can be sure it's due to soldering. It always is. There probably aren't that many bad connections. It's just a matter of narrowing down the area to look. Once you know where to look, it's usually pretty obvious and you say “how did I miss that before?”

It is helpful to have a 20 MHz 'Scope and signal generator to trace signals through the circuits. With just a DVM, there isn't much you can do other then check to see if voltages seem reasonable. The schematics and circuit descriptions should help.

It took me a couple of tries to get the connections to the Si5351 to work on one of my builds. If calibration worked, then this is okay. If calibration didn't work, well this is the area to look it.

So, what works and what doesn't?

Getting power? Checking the input and output of the three regulators is always the first step.

One or more switches or encoder don't work? Most likely a backwards diode.

No audio, not even hiss? Check around U6 and U5.

Just some hiss can't hear any stations or atmospheric noise? You did make the input mod, right? If that's not it, the problem could be anywhere in the receiver signal flow. Make sure the mixers U2 and U3 are actually getting the BFO and LO signals.

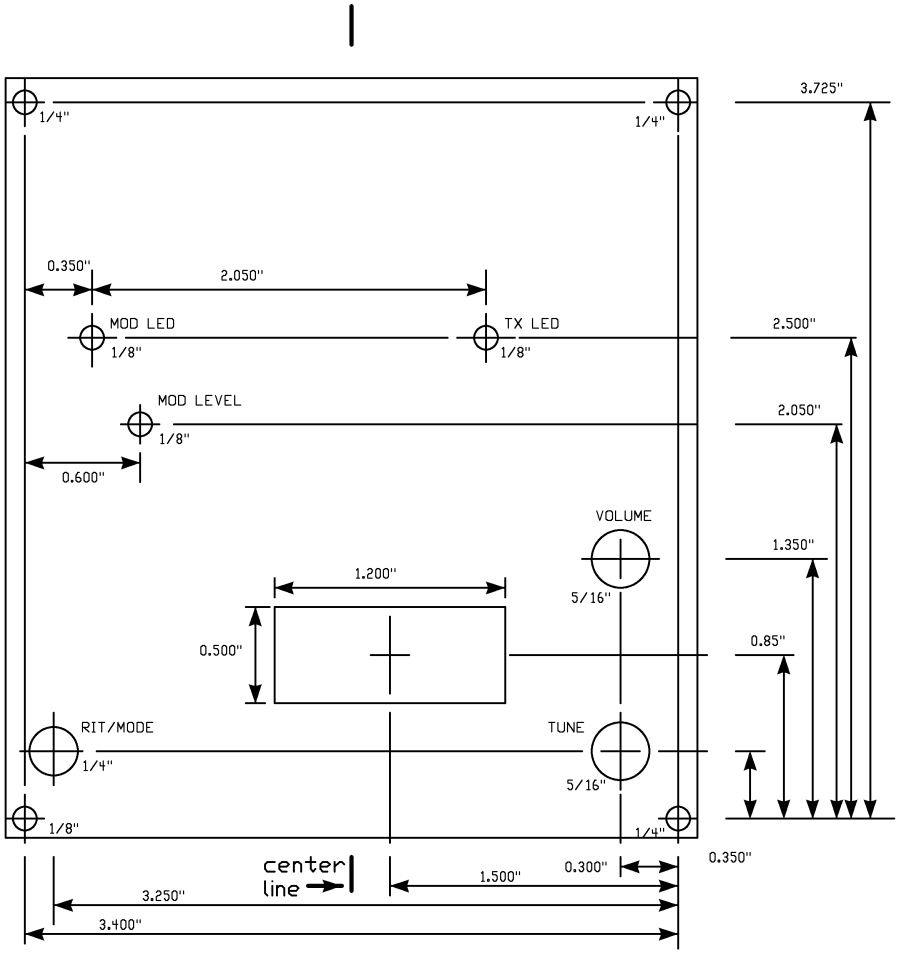
No transmit?
Is Tx supply switching on? Easy to tell, the Tx LED should come on.

Is a signal getting through the Band pass filter?

Are T1 and T2 wired right? There should be conductivity between all four leads when soldered to the board. If there continuity only between the pads on each side of the toroid, then the wires are not positioned correctly.

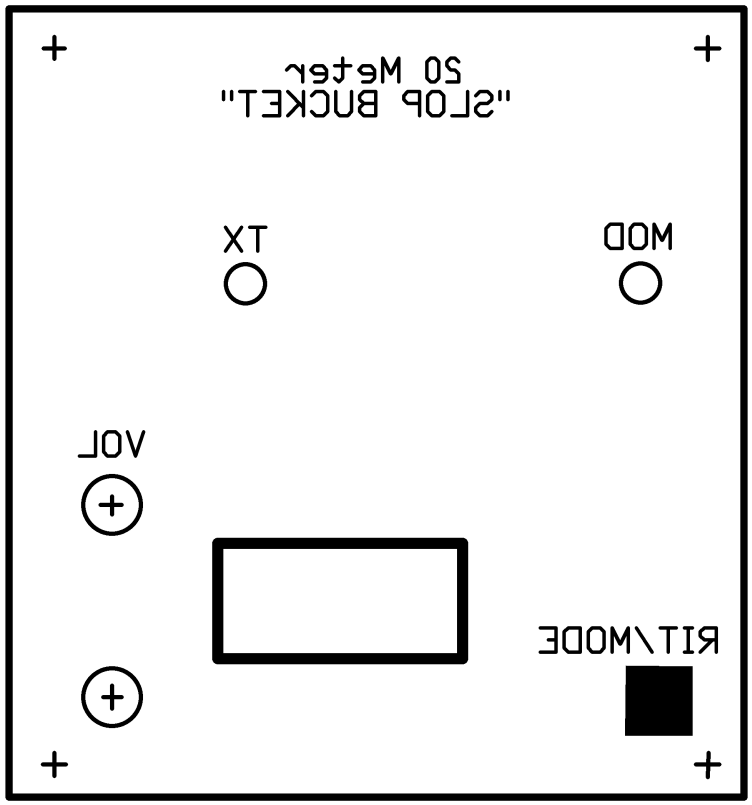
Output on Q4 and Q3?

Actual size drill template. Be sure your print options are set to “print 100%” and not “fit to page”.



Front panel overlay.

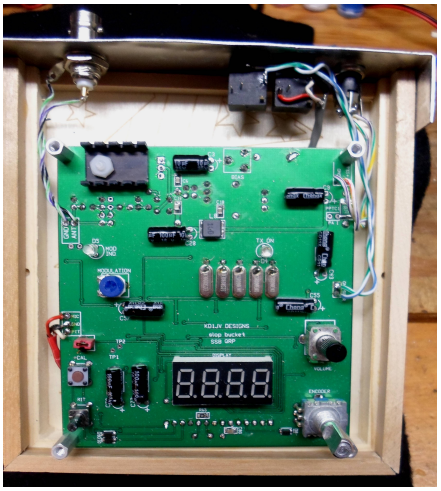
Print this to a piece of mylar transparency film to make a front panel overlay. The image is reversed so that the printing will be on the back side of the film and therefore protected from rubbing off.



Packaging the board.

This is where you get to be creative. It's unlikely you'll find a suitable commercially available box, so you'll have to make something custom. I like to check out the craft isle at Walmart to see if I can find anything to use to start with there. Those of you who live in a less rural area then I do may have a local hobby store to find material in.

I spotted this "Framed Paint Kit" in the craft isle which looked like it would be a good starting point. I brought along the board so I could check sizes. The board fit inside the frame with a little wiggle room to spare. I bought two of them figuring I'd use one for the top and one for the bottom. Near the frames you will also find a package of assorted sizes of square dowels. Get a package, you will need them to increase the height of the sides.



The frame is a bit longer then needed, so I trimmed 2.5" off the end, making a box 5" X 5.35" Looks like it could have been made 5" square.

I stacked the two frames and secured them together with some tape. A 1/8" hole is then drilled on each side over the center of the frame and a long 4-40 screw inserted to hold to two pieces together in alignment. I then used a small Dremel table saw to slice the ends off both pieces in one pass, ensuring they would be exactly the same length. Otherwise, you'd want a fine pitched tooth hand saw.

I decided to secure both the top and bottom pieces to the board via threaded spacers. Therefore, the mounting holes need to line up between the top and bottom pieces. First print out the drill template and trim. Center the template on the top piece, with the bottom mounting screw locations positioned 5/8" up from the bottom edge of the frame. Tape down the template and mark the drill holes with a sharp, needle point tool. Trace the outline of the display cut out with an Xacto knife, then finish the cut with the knife later. (Back up the bottom with a block of wood so you don't crack the thin wood)

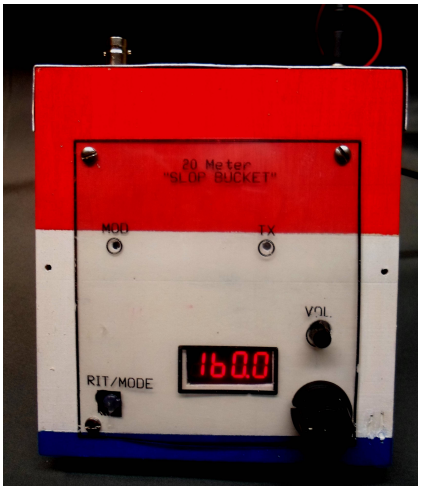
With the two frame pieces stacked and secured together, drill the four mounting holes straight down and through the bottom piece. Ideally with a drill press and even then they may not exactly line up and will need to be pulled a little.

Spacer rails have to be added around the sides of the frame to increase the hight. This is were the square dowels come in. The 6mm dowel turned out to be the exact right thickness to work with my spacers. I got fancy and mitered the corners. The spacers are glued to the bottom section with white wood glue. To sure up the edge of the frame which is now open, I used the 5mm dowels glued to the edge.

To mount the board, 3/4", 4-40 screws are inserted from the bottom, then a #4 nut. 3/8" long spacers are then added. 1/2" spacer could be used too and would eliminate the nut. 3/8" spacers screw on the ends of the 3/4" 4-40 screw the board sits over.

Finally you need an end panel to mount the jacks on. I folded over the ends of the panel and secured the panel to the frame with a couple of 1/2" 19 gauge wire nails. Two nails on each side to keep the panel from rocking.

When wiring the microphone/PTT jack, it is recommended you use shielded wire and route it away from the antenna jack.



Here's the finished product.

The RIT/MODE switch lever barely reaches the top of the panel. I opened up the hole for the switch some and put a blob of hot glue on the transparency over the switch location. This provides enough of a bump to push on to active the switch.

In a rush to get this done, the transparency is only held on by the four mounting screws. I'll leave it up to you to find a good way to make the screw holes. I just poked a hole through the transparency and that sort of worked. I need to get some spray glue to fix the transparency firmly to the panel.

Since red, white and blue paint came supplied with the frame, I decided to go with a patriotic paint scheme.

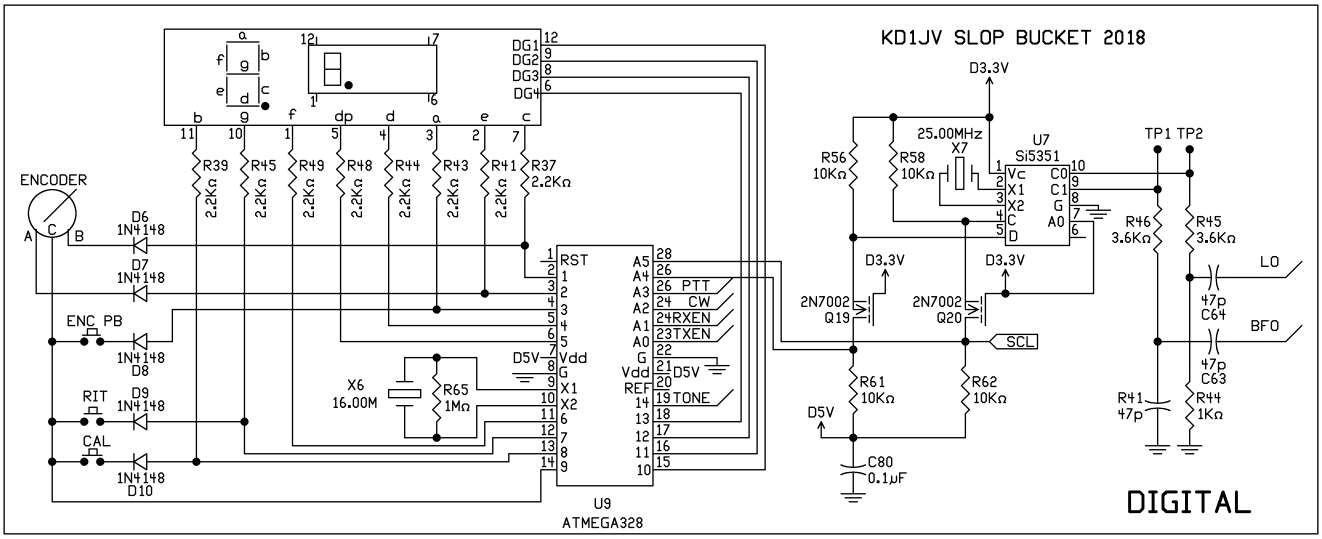
Oh, and drilling a few ventilation holes around the PA heat sinks would be a good idea too.

One thing about this packaging, it's pretty light. Don't want to sit on it though.

Reference – schematics, board tracks and Arduino program listing.

Schematics:

Digital circuits – processor and clock

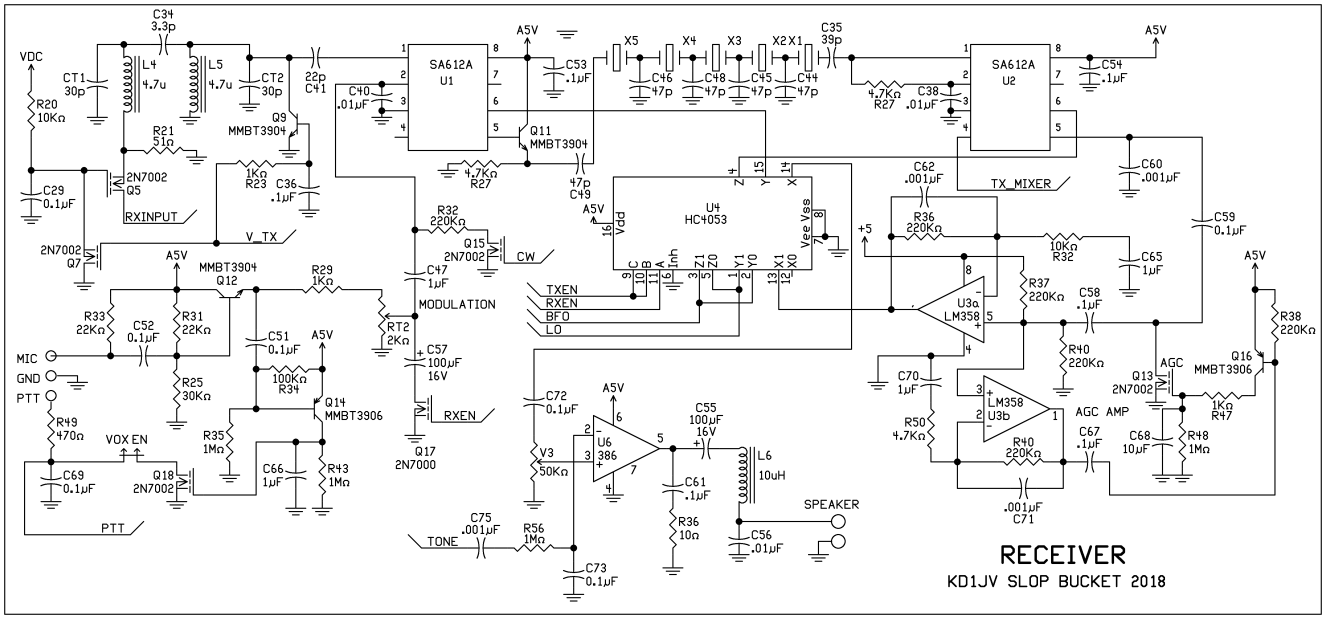


An Atmel MEGA328 processor is used for the control functions of the rig. It is programmed with the bootloader and is compatible with the Arduino IDE. Port numbering is consistent with the Arduino standard. The tuning encoder and switches are multiplexed to the LED display data bus. The switches are isolated with diodes which allow the switches to be connected to the bus only when required. Without the diodes, pushing a switch would affect the display.

A Silicon Labs Si5351 programmable clock chip is used for the Local Oscillator and BFO signals. Since processor runs on 5 volts and the clock chip on 3.3 volts, a level shifting circuit is used between the two. The inherent Source – Drain diode in the 2n7002 MOSFET pulls the clock input low when the processor port goes low and blocks the 5V with it is high.

The 3.3 volt square wave from the clock chip has to be reduced to suitable level for the mixers. A resistor divider is used to reduce the level for the Local Oscillator, while the BFO uses a resistor/capacitor divider. Using a cap instead of a resistor reduced the number of spurs coming out of the mixer and cleaned up the transmitter output.

The receiver circuits:



U4, 74HC4053, a tripple DPDT analog switch routes the LO and BFO signals to the appropriate mixer for receive or transmit. In receive mode, the LO is routed to U1 which is the 1st mixer and the BFO to U2, which the product detector. In transmit mode, the BFO is now routed to U1, which now works as the balanced modulator. The LO is routed to U2, which is now the transmit mixer. This allows the crystal filter to stay connected between the output of U1 and the input of U2.

Receive mode:

The input signal from the antenna first travels through the transmitter low pass filter, then is cap coupled the drain of Q5 which is the QSK switch. During receive, the gate is pulled high via R20 turning Q5 on. This connects the input signal to R21, a 51 ohm resistor. This couples the signal into the bottom of L4 and eliminates the need for a transformer winding. L4 and L5 tune the input to the frequency of interest, in this case 14 MHz. The output of the input filter is coupled to the 1st mixer, U1 a SA612A Gilbert cell mixer.

A 9 MHz, 5 crystal LSB filter is used for selectivity. The 4.7K resistors on the input and output terminate the filter and sooth out the response. The filter has about a 2500 Hz band width.

U2, another SA612A mixer brings the 9 MHz signal from the IF filter to audio. C60 removes some of the RF component of the mixer output.

The audio output from U2 is amplified by U3a and U3b. These are configured as non-inverting amplifies, biased at 2.5V by R37 and R40. This way both amplifiers get the same input signal with a minimum of parts. U3a has a gain of 22. C62 rolls off the high frequency response.

AGC is applied by shunting the signal at the junction of C59 and C58 to ground via Q13. AGC action is applied here where then is no DC bias to affect, which can cause thumps if it changes.

The level is detected by Q16, a PNP transistor. When the signal to the base starts to exceed the turn of voltage of the B-E junciton (about 500 mV) Q13 starts to turn on, pulling the gate of Q13 high. Once Q13 starts to turn on it shunts the signal at the C59/C58 junction towards ground, reducing the input signal to the amplifiers. This reduces the signal at the base of Q16, so it starts to turn off.

R48 and C68 set the AGC time constant. R47 slows down the response a little to help prevent overshooting. A sudden, real strong signal can effectively mute the audio and take a second or two to recover.

To make the AGC circuit more sensitive, U3b operates at twice the gain of U3a. This limits the output of U3a to 500 mv P-P maximum, otherwise it would be 1 V P-P which is too much, as it will drive the final audio amp well into clipping if the volume is turned up. 500 mV is still a lot, but it's a compromise between keeping the audio level down to a reasonable level without it kicking in too often. The output of the first stage audio amp is routed through one the 74HC4053 analog switches for muting and then to the audio power amp.

A classic LM386 audio amp provides the final audio gain and can drive a small speaker. L6 and C56 keeps out RF which might be picked up by the speaker or headphone leads. C61 and R36 are required for stability. In CW mode, side tone is injected into the second input via a low pass filter, R56/C73, to provide volume control independent side tone level.

Transmit mode:

Microphone input:

Bias for an Electrete Mic element is supplied via R33. Very little signal is needed to modulate the mixer, so the mic signal is only buffered by Q12, not amplified. The signal is further reduced by R29 in series with the modulation level control. The buffer transistor is on all the time so that it can drive the VOX circuit. In order to keep any sound picked up by the mic out of the mixer during receive, the audio input to the mixer is shunted to ground via C57/Q17. During transmit, Q17 is turned off allowing the audio to modulate the mixer.

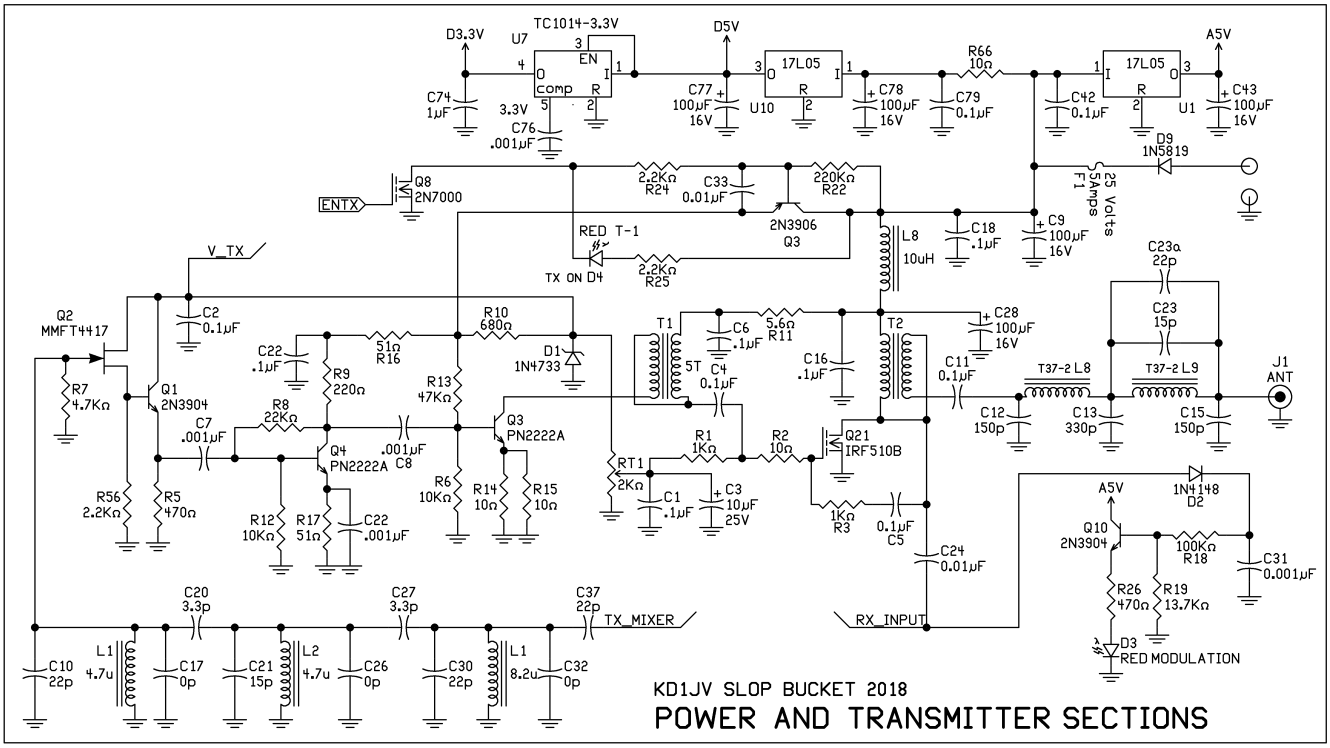
In CW mode, the mixer is unbalanced by connecting a 220K resistor from the modulation input of the mixer to ground via Q15. The BFO frequency is shifted 600 Hz when transmitting CW. This puts the carrier in the pass band of the IF filter and produces an output frequency which matches another station, putting you both on the same frequency. The mic input also becomes active in CW mode as there is no way to keep it turned off independently based on mode. If a mic is connected in CW mode, there could be some audio modulation.

The QSK switch Q5 removes most the transmitters output signal from getting into the receiver input. Any signal which leaks through Q5 due to capacitive coupling is removed by using Q8 to shunt the RF input side of the mixer to ground.

The output of U1 is now filtered by the IF filer, the output of which is then mixed with the LO in U2, which feeds the transmitter band pass filter.

A triple tuned band pass filter is used to clean up the output of the Tx mixer, U2. There are a couple of unused cap locations which were added in case odd cap values had to be made up.

The output of the band pass filter is terminated by a 4.7K resistor and buffered by Q2/Q1. The buffer insures the filter isn't affected by the power gain stage, Q4.



Transmit circuit power, V_TX, is switched on and off by Q3. C33 provides some rise/fall time to the switched voltage. V_TX is semi-regulated with a 5.1V zener diode and powers the filter input buffer, the Q4 stage and the base bias for Q3. V_TX also activates the QSK switching and the bias voltage for the PA MOSFET.

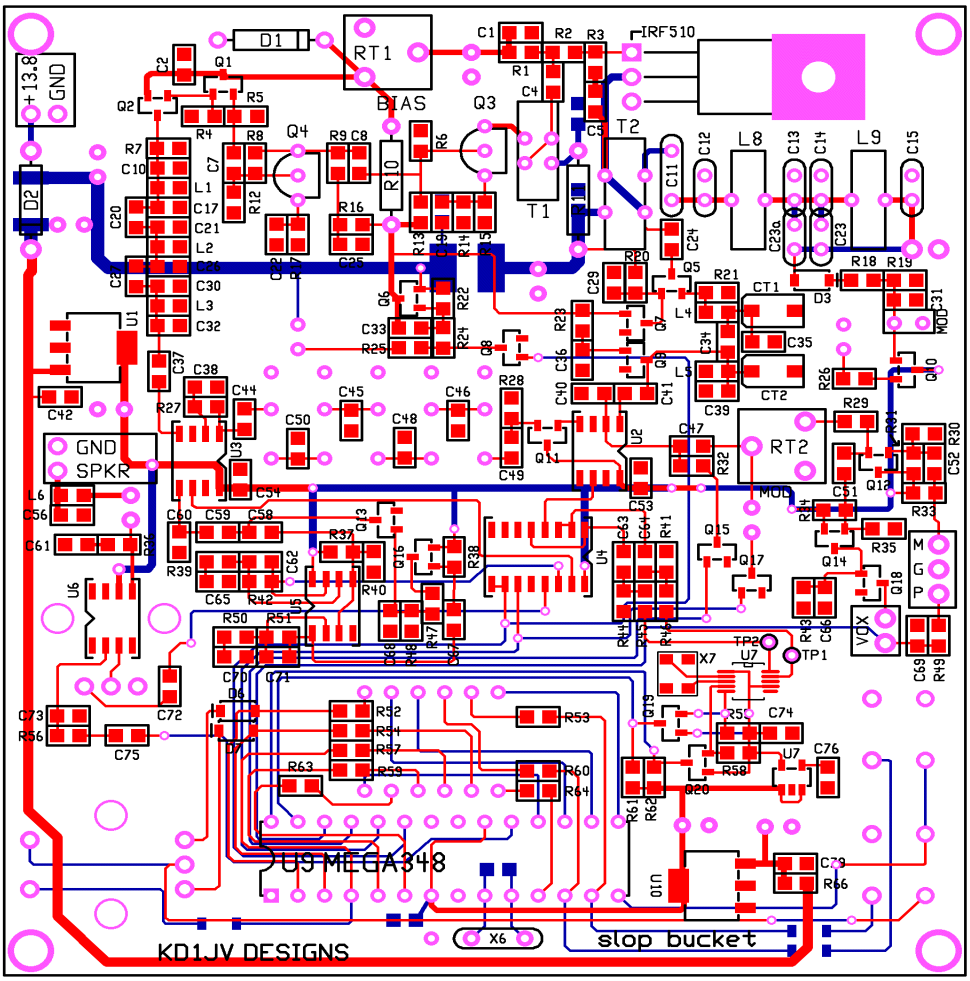
Supply voltage is applied to the PA and driver through L8. Without this, the transmitter liked to oscillate with just a hint of bias on the PA. Some negative feedback is applied to the PA which helps with linearity and stability.

The output low pass filter with 2nd harmonic trap keeps harmonic spurs suppressed to -50 dBc or better.

The modulation indicator uses a voltage divider from the rectified RF output to bias a NPN transistor. With the LED in series with the emitter, it takes a minimum of about 2V to get the LED to start coming on. This way the LED only gets bright near the typical peak power output.

Separate analog and digital 5V regulators, along with liberal use of 100 ufd caps keeps digital noise out of the receiver. D9 provides the reverse polarity protection and F1, a solid state resetable fuse should keep anything from catching fire if there is an over current problem.

Board track layout to help you trace out the connections if needed.



Arduino Firmware Sketch listing:

```
/*Frimware for "The Slop bucket" by KD1JV
 *


*/
#include <EEPROM.h>
#include "Wire.h"
#include "si5351.h"

Si5351 si5351;
#define SLED4 9
#define SLED3 10
#define SLED2 11
#define SLED1 12

//red leds

#define LED_N_0 0x14
#define LED_N_1 0xf6
#define LED_N_2 0x4c
#define LED_N_3 0x64
#define LED_N_4 0xa6
#define LED_N_5 0x25
#define LED_N_6 0x85
#define LED_N_7 0x76
#define LED_N_8 0x04
#define LED_N_9 0x26
#define LED_r 0xcf
#define LED_neg 0xef
#define LED_C 0x1d
#define LED_n 0xc7
#define LED_E 0x0d
#define LED_t 0x8d
#define LED_o 0xc5
#define LED_d 0xc4
#define LED_A 0x06
#define LED_L 0x9d
#define LED_P 0x0e
#define LED_F 0x0f

//flag definitions

#define RIT_ON B00000001
#define UPflag 0x01
#define DWNflag 0x02

#define E_sw 2
#define C_sw 6
#define R_sw 7
#define cw_mode 1
#define rit_on 1

// register names
byte ritflag = 0;
byte state;
unsigned long tcount;
long duration=0;

int Eadr;

byte BANDpointer = 1;
byte EncoderFlag = 0;
byte OUT_OF_BAND = 0;
byte sw_inputs ;
byte digit1 = 0xff;
byte digit2 = 0xff;
byte digit3 = 0xff;
byte digit4 = 0xff;
byte digitX = 0xff;

byte ten_mhz;
byte one_mhz;
byte hundred_khz;
byte ten_khz;
byte one_khz;
byte hundred_hz;
byte d1temp = 0x00;
byte Tx_mode = 0;
```

```
volatile int        digit_counter = 1;

const int      MUTE =    A1;
const int      TXEN =    A0; // production A0
const int      CWTX =    A2;
const int      PTT    =    A3;

   //frequency tuning
int                 stepSize;      // tuning rate pointer
long int        stepK;           // temp storage of freq tuning step
long int        Fstep100    =    10000;                          // 100 Hz step
long int        Fstep500 =      50000;
long int        Fstep1K = 100000;
long int        Fstep10K    =1000000;
//encoder

volatile int c = HIGH;               // init state of pin A
volatile int cLast = HIGH;        // init last val the same, low
volatile int d = HIGH;               // make data val low

unsigned long      frequency;
unsigned long      freq_result;
unsigned long      VFOfreq;
unsigned long      TXfreq;
unsigned long      OPfreq;
unsigned long      CW_TX_freq;
unsigned long      OPfreq_temp;
unsigned long      RITtemp1;
unsigned long      RITtemp;
unsigned long      RITresult;
unsigned long      time1;
unsigned long      time0;
unsigned long      CW_BFO;
unsigned long    IFfreq_SSB;
unsigned long    IFfreq_CW;
unsigned long    SSB_BFO;
unsigned long      IFfreq;
int                     REG = 0;
int                     temp;
signed long          cal_value = 0;

// registers for limit testing

unsigned long    low_band_limit;    //low limit, band tuning
unsigned long    high_band_limit; //high limit, band tuning

ISR (TIMER1_COMPA_vect) {TIMER1_SERVICE_ROUTINE();}

void setup() {
  //switch inputs

DDRB = 0x3f;
DDRD = 0xff;

pinMode(A0, OUTPUT);
pinMode(A1, OUTPUT);
pinMode(A2, OUTPUT);
pinMode(A3, INPUT_PULLUP);

digitalWrite(TXEN, LOW);
digitalWrite(CWTX, LOW);
digitalWrite(MUTE, LOW);

  si5351.init(SI5351_CRYSTAL_LOAD_6PF, 0); //set PLL xtal load

noInterrupts();   //this sets up the timer interup for display multiplexing and switch read.
TCCR1A = 0;
TCCR1B = 0;
TCNT1 = 0;

OCR1A = 238;
TCCR1B = 0x0b;
TIMSK1 |= (1 << OCIE1A);
interrupts();   //turn on the interputs
delay(100); //let things settle down
cal_data();       //load calibration data
    si5351.set_correction(cal_value); //correct the clock chip error
    stepK = Fstep100 ; //default tuning rate
    stepSize = 1;   //set the tuning step pointer
```

```
BANDpointer = 5;   //select 20 meters

   get_band();
   digit4 = LED_N_6;
   digit3 = LED_n;
   si5351.set_freq(SSB_BFO, 0ULL, SI5351_CLK1);
   IFfreq = SSB_BFO;
   IFfreq_CW = SSB_BFO - 60000;
   delay(500);
   displayMHZ();
   PLLwrite();
   digitalWrite(MUTE, HIGH);
}

void loop() {
   // test for switch closed

if (digitalRead(PTT) == LOW) {transmit();}

state = bitRead(sw_inputs,E_sw);     //read change tuning step switch
if (state == LOW) {nextFstep();} // debounce, wait for switch release
while (state == LOW) {
   delay(10) ;
   state = bitRead(sw_inputs,E_sw);
}


if (bitRead(sw_inputs, R_sw) == LOW) {timebutton();}
while (state == LOW) {
   delay(10) ;
   state = bitRead(sw_inputs,R_sw);
}

    if (bitRead(sw_inputs,C_sw)== LOW) {timebutton2();} // debounce, wait for switch release
while (state == LOW) {
   delay(10) ;
   state = bitRead(sw_inputs,C_sw);
}

    if (EncoderFlag == 2) {Tune_UP();} //test tune up flag
    if (EncoderFlag == 1) {Tune_DWN();} //test tunr down flag
}
//*****************************************************************
//end of switch polling loop
//*****************************************************************
void timebutton() {

duration = 0;   //clear button duration counter
time0 = tcount; //set basetime to current counter value

do {time1 = tcount; duration = time1- time0; //calculate how long the button has been pushed
//if (duration > 50)    {digit4 = LED_r; digit3 = LED_N_1; digit2 = LED_t; digit1 = 0xff;} //short push message
if (duration > 1000) {digit4 = LED_C; digit3 = LED_N_0; digit2 = LED_d; digit1 = LED_E;} //1 second push message
//if (duration > 5000) {;digit4=LED_C; digit3 =LED_A; digit2 = LED_L; digit1 = 0x00;}   //2 second push message
delay(1); //for some reason a delay call has to be done when doing bit read flag tests or it locks up
//this doesn't seem to be a problem when doing digital reads of a port pin instead.
}

while (bitRead(sw_inputs,R_sw) == 0); // wait until the bit goes high.

duration = time1- time0; //the duration result isn't saved when exiting the do/while loop, so has to be calculated again
//if (duration >5000) {calibration(); duration = 0;} //test duration to jump to desired function
if (duration >1000){int_cw_mode(); duration = 0;}
if (duration >50){RIT(); duration = 0;}
}

//frequency or code speed adjust test

void   Tune_UP() {
        EncoderFlag = 0;   //clear the encoder flag
        FREQ_incerment();
}

void Tune_DWN() {
        EncoderFlag = 0;
        FREQ_decerment();
}

// adjust the operating frequency
```

```
void FREQ_incerment( ) {
OPfreq   = OPfreq + stepK;   //add frequenc tuning step to frequency word
limitTest( );
 if (ritflag &= RIT_ON == 1){RITdisplay( );} //test for RIT mode
   else   displayMHZ( );
         PLLwrite( );
}

void FREQ_decerment( ) {
OPfreq   = OPfreq - stepK;
limitTest( );
if (ritflag &= RIT_ON == 1){RITdisplay( );}
else   displayMHZ( );
         PLLwrite( );
}
void limitTest( ) {

OUT_OF_BAND = 0;
if (OPfreq < low_band_limit) {OUT_OF_BAND = 1;}
if (OPfreq > high_band_limit) {OUT_OF_BAND =1;} //band tuning limits
}
//toggle tuning step rate
void   nextFstep ( ) {

++ stepSize ;
if (stepSize == 5) {(stepSize = 1);}

 switch(stepSize) {
 case 1:
         stepK = Fstep100;
         d1temp = digit1;
         digit1 = 0xff;   // 0x00 CC
         delay(500);
         digit1 = d1temp;
         break;
 case 2:
         stepK = Fstep500;
         d1temp = digit1;
         digit1 = 0xff; //0x00 CC
         delay(500);
         digit1 = d1temp;
         break;
case 3:
         stepK = Fstep1K;
         d1temp = digit2;
         digit2 = 0xff;   //0x00 CC
         delay(500);
         digit2 = d1temp;
         break;
case 4:
         stepK = Fstep10K;
         d1temp = digit3;
         digit3 = 0xff;   //0x00 CC
         delay(500);
         digit3 = d1temp;
         break;
}
}

void int_cw_mode( ) {
if (Tx_mode == 0){Tx_mode = 1; digit4 = LED_C; digit3 = 0xff; digit2 = LED_N_0; digit1 = LED_n; delay(1000); }
else {Tx_mode = 0; digit4 = LED_C; digit3 = 0xff; digit2 = LED_N_0; digit1 = LED_F; delay(1000); }
displayMHZ( );
}

/*
 * RIT mode stuff here
 */

void RIT( ) {

if (ritflag & RIT_ON == 1){RIText( );}
else RITenable( );
}

void RITenable( ){
ritflag |= RIT_ON;
RITtemp1 = VFOfreq;
RITtemp = OPfreq;
RITdisplay( );
```

```
}

void RIText() {

ritflag =0;
OPfreq = RITtemp;
PLLwrite();
displayMHZ();
}

void RITdisplay() {
if (RITtemp >= OPfreq)
{
   RITresult = RITtemp - OPfreq;
   digit3 = LED_neg;
   }
else
   {
      RITresult = OPfreq - RITtemp;
      digit3 = 0xff;
      }

digit4 = LED_r;
frequency = RITresult;
frequency = frequency/100;
freq_result = frequency%10000;
freq_result = freq_result/1000;
hex2seg();
digit2 = digitX;
freq_result = frequency%1000;
freq_result = freq_result/100;
hex2seg();
digit1 = digitX;
}

////////////////////////////////////////////////
//This breaks up the frequency value into decades and then converts the result
//hex value into the LED 7 seg map.
////////////////////////////////////////////////

void displayMHZ(){
   frequency = OPfreq/100;                    //first divide by 100 to remove the fractional Hz digits

freq_result = frequency%100000000;        //get the 10 MHz digit by first getting the remainder
freq_result = freq_result / 10000000;   //divide the remainder by 10,000,000 to get the MSD
hex2seg();                                    //convert the result to the 7 segment code
ten_mhz = digitX;

freq_result = frequency%10000000;        //get the 100,000 kHz digit by first getting the remainder
freq_result = freq_result / 1000000;   //divide the remainder by 100,000 to get the MSD
hex2seg();                                    //convert the result to the 7 segment code
one_mhz = digitX;

freq_result = frequency%1000000;
freq_result = freq_result / 100000;
hex2seg();
hundred_khz = digitX;

freq_result = frequency%100000;
freq_result = freq_result/10000;
hex2seg();
ten_khz = digitX;

freq_result = frequency%10000;
freq_result = freq_result/1000;
hex2seg();
one_khz = digitX;

freq_result = frequency%1000;
freq_result = freq_result/100;
hex2seg();
hundred_hz = digitX;

if (OUT_OF_BAND == 0) {digit1=hundred_hz; digit2 = one_khz; digit3 = ten_khz; digit4 = hundred_khz;}
else {digit1=ten_khz; digit2 = hundred_khz; digit3 = one_mhz; digit4 = ten_mhz;}
}

void   hex2seg()
{
   if (freq_result == 0) {digitX = LED_N_0;} //this is the conversion table
```

```
   if (freq_result == 1) {digitX = LED_N_1;}
   if (freq_result == 2) {digitX = LED_N_2;}
   if (freq_result == 3) {digitX = LED_N_3;}
   if (freq_result == 4) {digitX = LED_N_4;}
   if (freq_result == 5) {digitX = LED_N_5;}
   if (freq_result == 6) {digitX = LED_N_6;}
   if (freq_result == 7) {digitX = LED_N_7;}
   if (freq_result == 8) {digitX = LED_N_8;}
   if (freq_result == 9) {digitX = LED_N_9;}
}

/*
 *
 * timer outside of the normal Ardinu timers
 * does keyer timing and port D mulitplexing for display and
 * switch inputs.
 */

void TIMER1_SERVICE_ROUTINE()
{
   ++tcount;
   ++ digit_counter;
   if (digit_counter == 6 ) {(digit_counter = 1);}
   switch(digit_counter)

   {
      case 1:
      digitalWrite(8, HIGH);
      DDRD = 0xff;
      PORTD = digit1;
      digitalWrite(SLED1, HIGH);
      break;

   case 2:
     digitalWrite(SLED1, LOW);
      PORTD = digit2;
      digitalWrite(SLED2, HIGH);
      if (OUT_OF_BAND == 0) {digitalWrite(2, LOW);}
      break;

 case 3:
      digitalWrite(SLED2, LOW);
      PORTD = digit3;
      digitalWrite(SLED3, HIGH);
      if (OUT_OF_BAND == 1) {digitalWrite(2, LOW);}
      break;

   case 4:
      digitalWrite(SLED3, LOW);
      if (digit4 == 0x14){digit4 = 0xff;}
      PORTD = digit4;
      digitalWrite(SLED4, HIGH);
      break;
 case 5:
      digitalWrite(SLED4, LOW);
      DDRD = 0x00;
      PORTD= 0Xff;
      digitalWrite(8, LOW);
   for (int i = 0; i < 10; i++) {sw_inputs = PIND;} //read the switches 4 times to debounce
      digitalWrite(8, HIGH);
   c = bitRead(sw_inputs,1);  //read encoder clock bit
   if (c !=cLast){encoder();} //call if changed
 break;
}
}

/*
 * encoder, test for direction only on 0 to 1 clock state change
 */

void encoder()         { if (cLast ==0){
      d = bitRead(sw_inputs,0);
//      d = c^d;                        //test data for up/down
      if ( d == LOW ) {
          EncoderFlag = 2 ;               //if low
      }
      else {
          EncoderFlag = 1;                //if high
      }
}
```

```
        cLast = c;                    //store new state of clock
}

/*
 * output the frequency data to the clock chip.
 */

void PLLwrite() {
   if (BANDpointer >=5) {VFOfreq = OPfreq + IFfreq;}
 else VFOfreq = IFfreq - OPfreq;

if (Tx_mode == 1){VFOfreq = VFOfreq - 60000;}

 si5351.set_freq(VFOfreq, 0ULL, SI5351_CLK0);
}

void transmit(){
if (OUT_OF_BAND == 1) {freezeTX();}
if (Tx_mode == 1) {CW_transmit();}
else SSBxmit();
}

void freezeTX() {
digit1 = LED_neg; digit2 = LED_neg; digit3 = LED_neg; digit4 = LED_neg;
do delay(2);
while (digitalRead(PTT)== LOW);
displayMHZ();

}


void SSBxmit(){
digitalWrite(MUTE, LOW);
if (ritflag == 1){si5351.set_freq(RITtemp1, 0ULL, SI5351_CLK0);}
digitalWrite(TXEN, HIGH);
do delay(2);
while (digitalRead(PTT)== LOW);
digitalWrite(TXEN, LOW);

delay(5);
if (ritflag == 1){si5351.set_freq(VFOfreq, 0ULL, SI5351_CLK0);}
digitalWrite(MUTE, HIGH);
}

void CW_transmit(){
digitalWrite(MUTE, LOW);
si5351.set_freq(IFfreq_CW, 0ULL, SI5351_CLK1);
if (ritflag == 1){si5351.set_freq(RITtemp1, 0ULL, SI5351_CLK0);}
digitalWrite(TXEN, HIGH);
digitalWrite(CWTX, HIGH);
tone(13, 600);
do delay(2);
while (digitalRead(PTT)== LOW);

digitalWrite(CWTX, LOW);
delay(9);
digitalWrite(TXEN,LOW);
noTone(13);
if (ritflag == 1){si5351.set_freq(VFOfreq, 0ULL, SI5351_CLK0);}
si5351.set_freq(SSB_BFO, 0ULL, SI5351_CLK1);
delay(8);
digitalWrite(MUTE, HIGH);
}

/*
 * Ref oscillator frequency is calibrated first
 * 10MHz signal outputted on CLOCK 0
 * tune to equal exactly 10.000,000 MHz
 * adjusted in 1 Hz steps.
 * display shows corectional factor in 0.01Hz. 1Hz is 100 on display.
 * Push keyer PB to advance to next IF offset cal
 * Push keyer PB to advance to band select
 * Push keyer PB again to store and exit
 */

void timebutton2() {

duration = 0;   //clear button duration counter
time0 = tcount; //set basetime to current counter value
```

```
do {time1 = tcount; duration = time1- time0; //calculate how long the button has been pushed
if (duration > 50)    {digit4 = LED_C; digit3 = LED_A; digit2 = LED_L; digit1 = 0xff;} //short push message
if (duration > 1000) {digit4 = LED_N_6; digit3 = LED_F; digit2 = LED_N_0; digit1 = LED_E;} //1 second push message
delay(1); //for some reason a delay call has to be done when doing bit read flag tests or it locks up
//this doesn't seem to be a problem when doing digital reads of a port pin instead.
}

while (bitRead(sw_inputs,C_sw) == 0); // wait until the bit goes high.

duration = time1- time0; //the duration result isn't saved when exiting the do/while loop, so has to be calculated again
if (duration >1000){BFO_ADJ(); duration = 0;}
if (duration >50){calibration(); duration = 0;}
}

void calibration(){
unsigned long temp = 0;
calwrite();
delay(500);
state = bitRead (sw_inputs,E_sw);
while (state == HIGH) {if  (EncoderFlag == 1) {ADJ_UP();}
if  (EncoderFlag == 2) {ADJ_DWN();}
state = bitRead (sw_inputs,E_sw);}
temp = cal_value;  //store the cal value
EEPROM.write(4, temp);
temp = cal_value >>8;
EEPROM.write(5, temp);
temp = cal_value >>16;
EEPROM.write(7, temp);
temp  = cal_value >>24;
EEPROM.write(8,temp);

while (state == LOW) {delay(100); state = bitRead (sw_inputs,E_sw);}

//changeBand();

si5351.set_freq(SSB_BFO, 0ULL, SI5351_CLK1);
PLLwrite();
displayMHZ();
}

void ADJ_UP(){
   temp = temp + 100;
   EncoderFlag = 0;
   calwrite();
}
void ADJ_DWN() {
   temp = temp - 100;
   EncoderFlag = 0;
   calwrite();
 }

void calwrite() {
cal_value = temp;
si5351.set_correction(cal_value);
si5351.set_freq(1000000000, 0ULL, SI5351_CLK1);
//displayfreq();
}

void BFO_ADJ() {
state = bitRead (sw_inputs,E_sw);
while (state == HIGH) {if  (EncoderFlag == 1) {ADJ_UP_B();}
if  (EncoderFlag == 2) {ADJ_DWN_B();}
state = bitRead (sw_inputs,E_sw);}

temp = SSB_BFO;  //store the cal value
EEPROM.write(9, temp);
temp = SSB_BFO >>8;
EEPROM.write(10, temp);
temp = SSB_BFO >>16;
EEPROM.write(11, temp);
temp = SSB_BFO >>24;
EEPROM.write(12,temp);

while (state == LOW) {delay(100); state = bitRead (sw_inputs,E_sw);}
}

void ADJ_UP_B(){
        EncoderFlag =0;
        SSB_BFO = SSB_BFO + 1000;
        si5351.set_freq(SSB_BFO, 0ULL, SI5351_CLK1);
```

```
}

void ADJ_DWN_B ( ) {
          EncoderFlag = 0;
          SSB_BFO = SSB_BFO - 1000;
          si5351.set_freq(SSB_BFO, 0ULL, SI5351_CLK1);
}

void cal_data( ){

temp =0;
temp = EEPROM.read(8);
cal_value = temp;
cal_value = cal_value << 8;
temp = EEPROM.read(7);
cal_value = cal_value + temp;
cal_value = cal_value << 8;
temp = EEPROM.read(5);
cal_value = cal_value + temp;
cal_value = cal_value << 8;
temp = EEPROM.read(4);
cal_value = cal_value + temp;

if (cal_value >= 0xff00) {cal_value = 00000;}

}

void changeBand( ){
   digit4 = LED_N_6;
   digit3 = LED_n;
   get_band( );

while (state == LOW) {delay(100); state = bitRead (sw_inputs,E_sw);}

do {
   if (EncoderFlag ==1) {nextband( );}
}
while (bitRead(sw_inputs,E_sw) !=0);
displayMHZ( );
PLLwrite( );
EEPROM.write(6,BANDpointer);

do {delay(50);}
while (bitRead(sw_inputs,E_sw) !=1);

}

void nextband ( ) {
   EncoderFlag =0;
++BANDpointer ;
if (BANDpointer == 6) {(BANDpointer = 1);}
get_band( );
do  {delay(50);}
while (bitRead(sw_inputs,E_sw) !=1);
}

void get_band( ) {
switch(BANDpointer) {
   case 1:
   BAND16( );
   break;
   case 2:
   BAND80( );
   break;
   case 3:
   BAND60( ) ;
   break ;
   case 4:
   BAND40( ) ;
   break ;
   case 5:
   BAND20( ) ;
   break ;

}

}

void BAND16( ) {
      digit2 = LED_N_1;
```

```
        digit1  =  LED_N_6;
        low_band_limit  =  180000000;
        high_band_limit  =  182000000;
        OPfreq  =  180200000;
        SSB_BFO  =        900090000;
}

void  BAND80( )  {
        digit2  =  LED_N_8;
        digit1  =  LED_N_0;
        low_band_limit  =  350000000;
        high_band_limit  =  400000000;
        OPfreq  =  375000000;
        SSB_BFO  =        900090000;
}

void  BAND60( )  {
        digit2  =  LED_N_6;
        digit1  =  LED_N_0;
        low_band_limit  =  535150000;
        high_band_limit  =  536650000;
        OPfreq  =  535150000;
        SSB_BFO  =        900090000;
}

void  BAND40( )  {
        digit2  =  LED_N_4;
        digit1  =  LED_N_0;
        low_band_limit  =  700000000;
        high_band_limit  =  730000000;
        OPfreq  =  715000000;
        SSB_BFO  =        1106040000;
}

void  BAND20( )  {
        digit2  =  LED_N_2;
        digit1  =  LED_N_0;
        low_band_limit  =  1400000000;
        high_band_limit  =  1435000000;
        OPfreq  =  1416000000;
        SSB_BFO  =        900090000;
}
```