A Modular 40 Meter CW Transceiver with VFO

Add a digital display and frequency control to a popular QRP rig.

Dr Jack Purdum, W8TEE; Farrukh Zia, K2ZIA, and Dennis Kidder, W6DQ

Way back in my Novice class license days, you had two choices for getting a rig: You either built it from scratch, or from a kit. There's a tremendous feeling of accomplishment and pride when you make contacts on something that you've built.

Today, we can build a VFO-controlled 40 meter band CW transceiver with an LCD display for less than $50. We snap prefabricated modules together in LEGO® fashion, and end up with a viable and fun transceiver. You can easily integrate the VFO/display board presented here into virtually any other rig and cover 160 to 10 meters.

At the Milford Amateur Radio Club Field Day location, we always have a GOTA (Get On The Air) station set up, where members of the public can make a contact. I overheard a mother talking to her enthusiastic young son: "Yes, it looks like a fun hobby, but where are you going to get the thousands of dollars it takes to buy the radio?" That's how our hobby is perceived. So, at next year's Field Day I'm attaching a sign to this rig: "Build this station for under $50." I selected a clear acrylic case (see Figure 1) so that passers-by can see how simple it is.

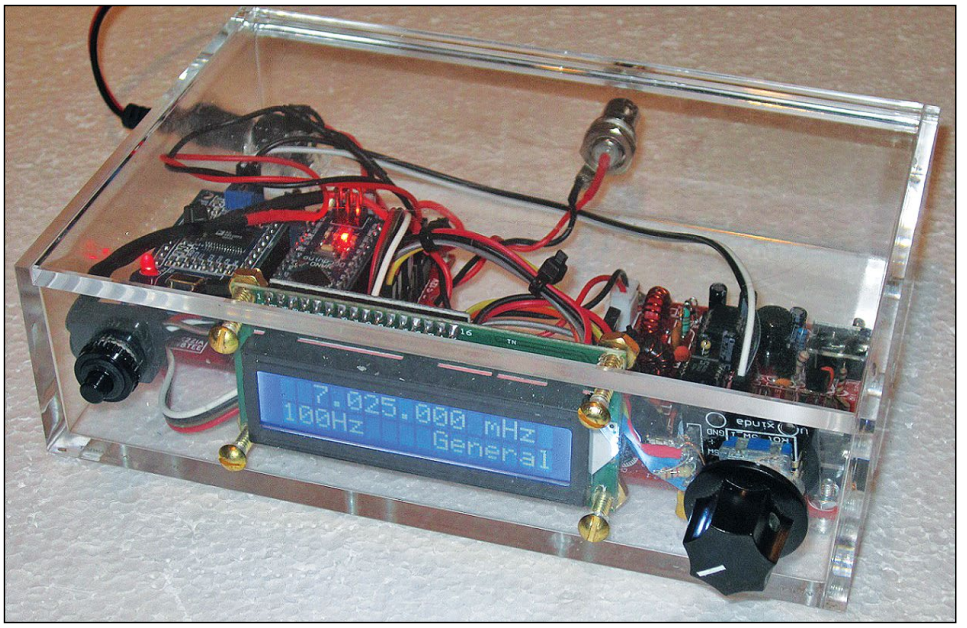

Figure 1 — The modular transceiver in a clear acrylic housing.

The Components

Table 1 shows the five major project components. We use the Arduino Nano microcontroller to control the rig's features. Make sure you get the Nano V3.0 that has a USB connector on the board. You can download the Arduino development environment free of charge.[1] Our program source code and assembly manual are available from the *QST* in Depth web page.[2] You could also use an Arduino Uno, Mega 1280, 2560, or Teensy, if you already own one.

The Forty-9er transceiver is sold as a 3 W crystal-controlled kit that operates from a 12 V dc power source. There have been some design changes to the transceiver since it was first introduced as the NorCal QRP Club's Forty-9er kit by Wayne Burdick, N6KR (of Elecraft fame), and Doug Hendricks, KI6DS (of Youkits fame).

Table 1
Major Construction Components

Component	Cost	Description
Forty-9er transceiver kit	$11.00	Crystal-controlled 3 W transceiver kit for 7023 MHz (**www.ebay.com** — search for "forty-9er"). Also available on Amazon as the SainSmart Forty-9er).
AD9850 DDS module	$8.00	Amazon and eBay
16 × 2 I2C LCD display	$6.00	Blue 1602 IIC with I2C interface SKU: EA-010204 (**www.yourduino.com/**)
Arduino Nano V 3.0	$3.50	Make sure it has the USB connector
Rotary encoder	$1.00	Purchased in a lot of 10 for $10
Total cost:	**$29.50**	**This leaves room for an enclosure, antenna and power connectors, and miscellaneous parts.**

Figure 2 shows the parts contained in the transceiver kit. We inventory parts by poking the components into a large foam sheet (don't poke the static-sensitive components, like ICs and transistors, into the sheet).

The instructions enclosed with the kit are very sparse, and omit construction details. To help with construction, we wrote an assembly instruction manual that you can download from the *QST* in Depth web page. Our manual includes schematics, and illustrates modifications needed to

accommodate our VFO and LCD display additions.

The small (1.675 × 1 inch) AD9850 DDS signal generator board forms the heart of the VFO. There are several variants of the board available, and not all are pin-for-pin compatible. It's best to select one that looks like the one pictured in Figure 3 if you plan to use our PCB for the VFO/Nano. The AD9850 chip uses a 125 MHz reference oscillator and operates from a 5 V source, which ties in nicely with the Arduino family of microcontrollers. We noticed that the chip ran a bit warm, so we dropped the supply voltage to 3.4 V, using a couple of diodes in series.

We chose the Nano microcontroller because of its low cost and small footprint. The downside is that the Nano doesn't accept a standard Arduino plug-in shield. Therefore, you must use prototype construction with non-pluggable shields, perf board, or the PCB we designed for the project.[3] The VFO presented here is a very stable, simplified version of the multiband VFO that Dennis Kidder, W6DQ, designed.[4]

Modifying the Forty-9er

The original Forty-9er transceiver is crystal controlled. It transmits and receives on a fixed frequency separated by a small offset. Figure 4 shows a partial schematic of the original Forty-9er as distributed with the kit. The receiver filter section (green boxed area) has a fixed frequency crystal filter (Y1) with a very narrow receiver bandwidth centered around 7.023 MHz. The red boxed area shows circuit elements that relate to the oscillator.

The transmit frequency is controlled by a second 7.023 MHz crystal (Y2) connected to the NE602 or NE612 internal oscillator circuit. We replaced the two crystal-controlled circuits with a VFO that covers the entire 40 meter band. Download our assembly instruction manual and follow the schematic for the following discussion.

The first modification changes the receiver input filter circuit to increase its bandwidth so it spans the entire 40 meter band. Replace the receiver narrow bandwidth crystal filter (green box, see Figure 4) consisting of C2, Y1, and C21, with an LC band-pass filter that covers the entire 40 meter band. Diodes D1 and D5 prevent

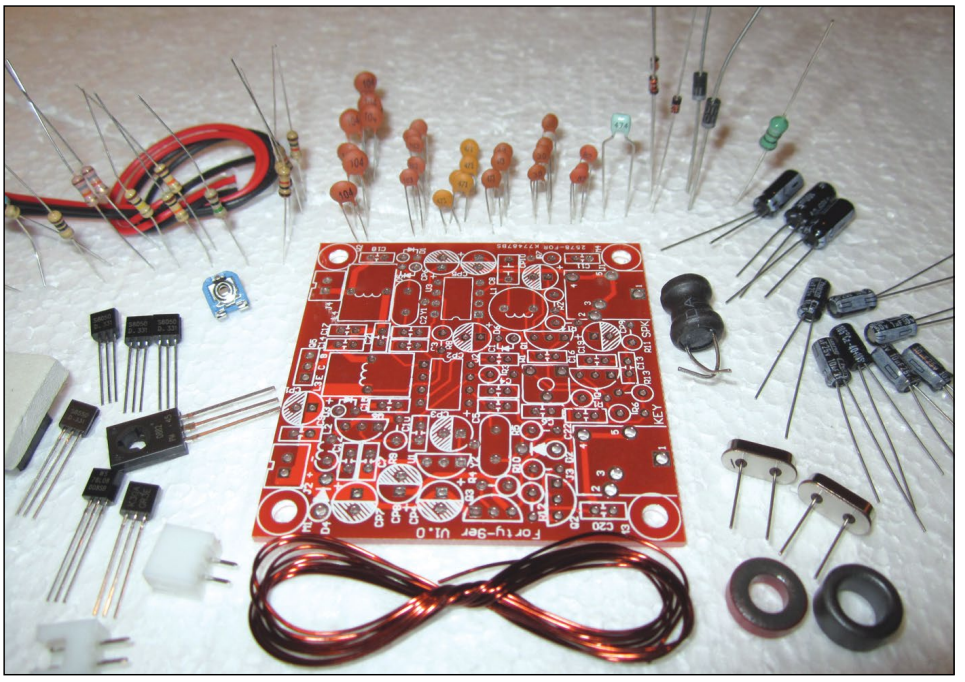

Figure 2 — The Forty-9er PCB and parts.

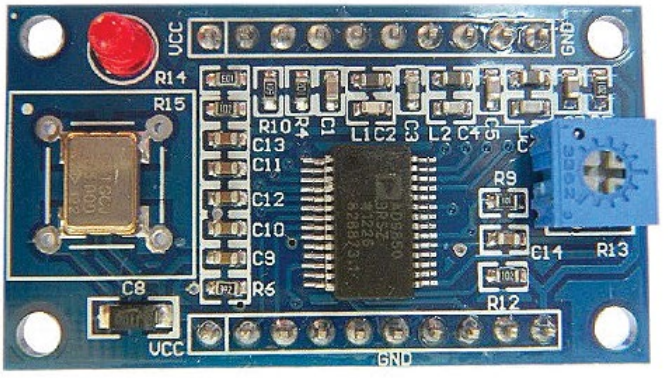

Figure 3 — The AD9850 Direct Digital Synthesizer (DDS) board.

Table 2
Components Affected by the Modifications

Component	Value	Note
C2*	30 pF or 33 pF	Replace with 82 pF capacitor
C4, C5	82 pF	Do not install
C6	0.01 µF	Do not install
C21*	30 pF or 33 pF	Replace with 0.01 µF capacitor
D2	1N4001	Replace with 4.7 V or 5.1 V Zener (install in same orientation as D2)
R3	10 kΩ	Do not install
R5	100 kΩ	Replace with 10 kΩ
W1	50 kΩ trimmer pot	Replace with jumper (see schematic diagram)
Y1	7.023 MHz	Replace with LC filter (see schematic diagram and assembly manual)
Y2	7.023 MHz	Replace with three-pin header (see schematic diagram and assembly manual)

*Note that original schematic shows C2 and C21 values as 33 pF, but the actual kit may contain 30 pF capacitors, as shown in modification schematic.

the transmitter output signal from damaging the NE602 mixer input. The LC bandpass receiver filter must prevent a dc path between diodes D1, D5, and NE602 mixer input INA (Pin 1), as stated in the NE602 data sheet.

A digitally controlled DDS VFO provides the transmit oscillator signal over the 40 meter band, so the internal oscillator in the NE602 chip and its associated external components (C4 – C6, D2, R3, R5, W1, Y2) are no longer needed. The KEY UP or DOWN (RX/TX) position of switch transistor Q2 provides the signal to shift the DDS VFO microcontroller transmit frequency over a small offset. Because the Nano digital input can withstand a maximum voltage of only 5.5 V, we use a 5.1 V Zener diode to limit the 12 V dc signal from Q2 to 5.1 V. A lower voltage Zener diode can be used as long as the voltage presents a logic-level high (about 3.5 V) to the Nano keying the transmitter.

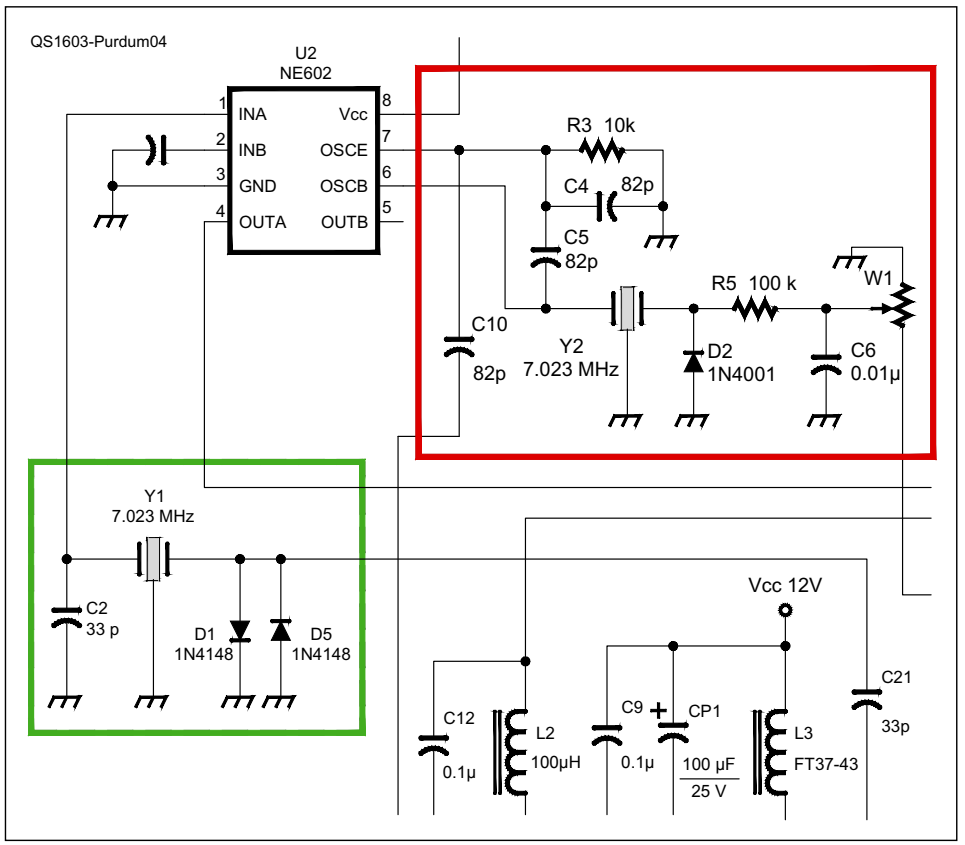

Figure 4 — Original Forty-9er frequency control schematic.

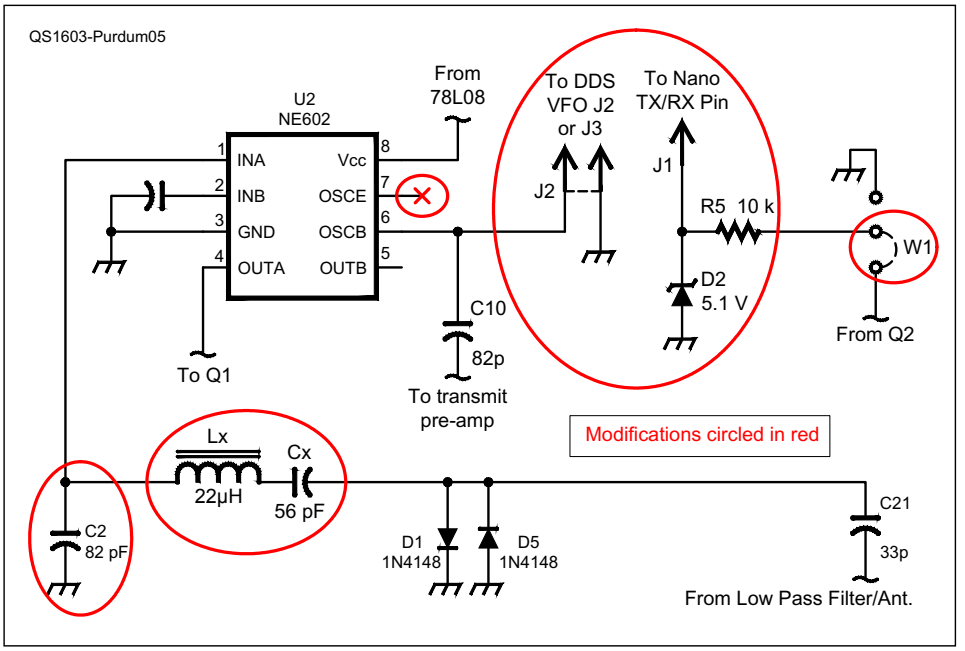

Figure 5 — Modified frequency control for the Forty-9er.

Build the board following our assembly instruction manual. The manual and Table 2 show a detailed list of components that are either replaced or omitted. Figure 5 shows the modified circuit. Changes to the basic kit are minimal. The image on the left in Figure 6 shows the silkscreen for the board and highlights the modified areas. The image on the right shows the actual board.

The Nano and VFO Board

We use a small PCB (see the schematic in Figure 7) to mount the AD9850 module and the Nano. The VFO output can be taken from either J2 or J3. Our RF output power was consistently lower than 3 W using the "straight" VFO design. The VFO output from J3 benefits from transistors Q1 and Q2 that raise the peak-to-peak voltage to about 4 V (adjustable by R1) that, in turn, drives the Forty-9er to about 3 W. Use the J2 VFO output if you prefer to run lower output power. The Nano and VFO can be built on a prototype board, perf board, or on our small PCB board.

Construction

Our project case measures approximately 6.75 × 4.75 × 1.75 inches, which is larger than needed. The case selection can accommodate more hardware later on. In Figure 1, the VFO/Nano board is on the left side, and the Forty-9er board is on the right. You can see the BNC antenna connector centered on the back panel, and the power connector hot-glued in place on the left side of the rig. After drilling a hole for the power plug, place a blob of hot glue inside the case and slide the internal power connector in place. We held it in place by pushing a wall-wart plug through the case hole and into the internal power connector until the glue was set. The headphones and key jacks on the Forty-9er board are accessible through holes drilled on the right side of the case.

We centered both the Nano/VFO and

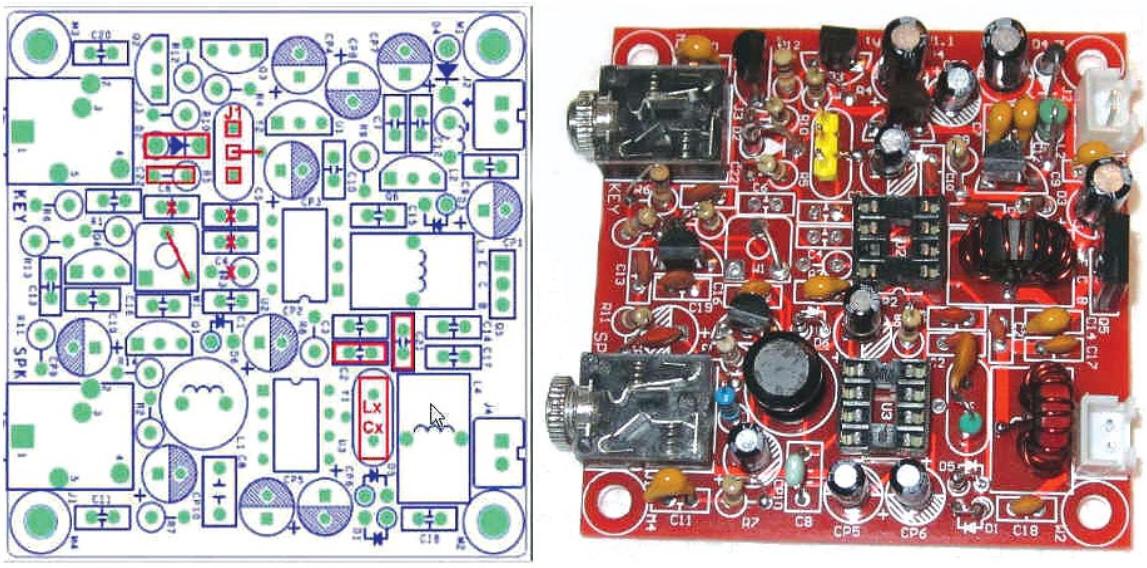

Figure 6 — Parts placement for modifying the board.

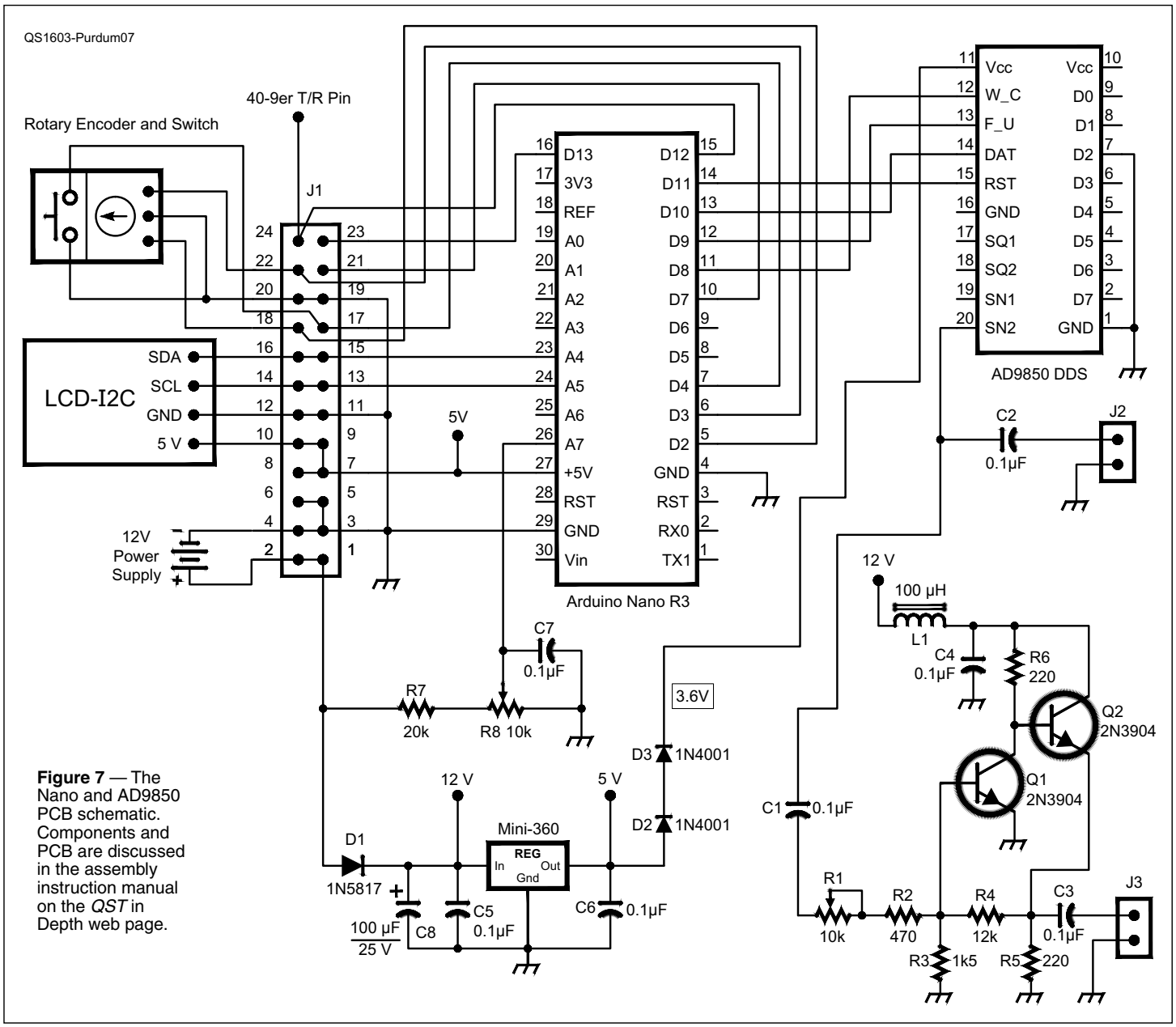

Figure 7 — The Nano and AD9850 PCB schematic. Components and PCB are discussed in the assembly instruction manual on the *QST* in Depth web page.

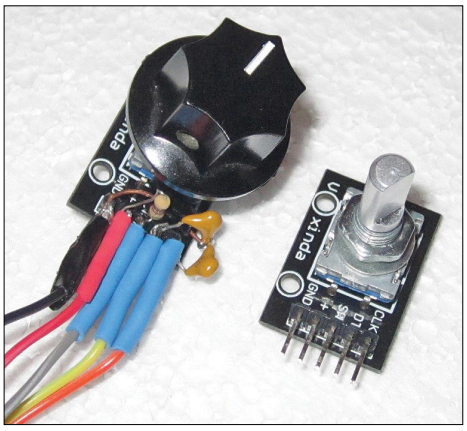

Figure 8 — The rotary encoder.

Forty-9er boards in the case. In retrospect, we would mount the VFO/Nano board more toward the rear of the case to allow easier access to the USB connector on the Nano for program changes. A pushbutton power switch mounts on the left side of the case front, and the right side holds the rotary encoder. Parts placement is not critical.

The LCD Display

Our LCD display uses the I2C interface to connect the display to the microcontroller board. The I2C interface uses just two control lines and two power lines — Pins 10, 12, 14, and 16 on J1 in Figure 7. A small potentiometer on the back of the display controls the LCD backlight. Set it once and forget it. You could use a non-I2C LCD display for the rig if you wish, because there are more than enough I/O pins available on the Nano. However, you will need to modify the interface from that shown in Figure 7, and you need to modify the software.

The Encoder

Figure 8 shows a KY-040 encoder. We purchased a package of 10 on **www.ebay.com** for about $10. Rotary encoders are designed to send a series of pulses as you turn the shaft. By measuring the sequence of the pulse chain, you can determine whether the shaft is rotating clockwise or counterclockwise. This particular encoder sends out 20 pulses for each full rotation of the encoder shaft. This means that a "detent" marks a new pulse sequence with every 18 degrees of shaft rotation. The Nano processes these pulse sequences.

A software-defined frequency tuning increment ranges from 10 Hz to 100 kHz per encoder pulse chain. Change the increment by pressing the encoder shaft. An internal switch in the KY-040 encoder signals the software to set the frequency increment value. We tied this switch to Pin 17 of J1 (D4 of Nano board), as seen in Figure 7. Each press of the switch increments by a factor of 10. If you increment past 100 kHz, the increment wraps around back to 10 Hz. Because the increment values are controlled in software, you can redefine them as you wish. The "100Hz" displayed in Figures 1 and 9 is the current frequency increment.

The encoder (Figure 8) has five pins. Two pins are the clock and data pins used by a software Interrupt Service Routine (ISR). Rotating the encoder shaft triggers an interrupt. When the software senses the interrupt, it suspends all other activity and immediately executes the ISR code. In our case, it changes the frequency of the rig and shows the new frequency on the LCD display. We use the Nano external interrupt pins D2 and D3 for the encoder. Pins 18 and 22 of J1 are tied to the two interrupt pins.

The center encoder pin links to the encoder switch. Because the KY-040 uses a mechanical mechanism for encoding, it is subject to bouncing. That is, the contacts vibrate as you move from one detent to the next, and the Nano is fast enough to read each vibration as an event in the pulse chain. This can result in sending a series of false pulses to the microcontroller before it stabilizes. We needed to de-bounce the encoder.

Figure 9 — The LCD display.

You could use software to filter the pulses by introducing a small delay in the pulse chain until the state of the encoder has stabilized. A delay of around 250 ms should remove the false pulses. Another alternative implements a hardware de-bounce solution by connecting 0.1 µF capacitors from the clock and data lines to ground. This option is shown in Figure 8. We wired the two capacitors directly to the encoder clock and data pins, and then to ground. In the final design, we tied the center switch pin to the +5 V using the internal pull-up resistor on the pins of the Nano. Program code activates the internal pull-up resistors. Pressing the rotary encoder shaft grounds the switch, and the increment value is adjusted in the software accordingly.

The modifications to the encoder result in a smoothly operating tuning knob. You can feel the detents as you turn the encoder shaft, and you can stop without over-shooting a target frequency. We opted not to use labels on the case for aesthetic reasons.

Software

The Arduino family of microcontrollers uses three types of memory — flash, SRAM, and EEPROM. The Nano and Uno boards include 32 Kbytes of non-volatile (memory state persists even after power is removed) flash memory for storing program code. The bootloader code uses about 2 Kbytes of flash memory, so you have slightly less than 30 Kbytes for your program. SRAM (Static Random Access Memory) holds the data stored in variables as the program runs. There are 2 Kbytes of volatile (loses data when power is removed) SRAM. There is 1 Kbyte of non-volatile EEPROM (Electrically Erasable Programmable Read-Only Memory), which is slower than flash memory.

You can do quite a lot with 32 Kbytes of program space. The program code that manages the transceiver display, VFO frequency, and encoder processing uses about 9 Kbytes of flash memory. The program data consumes 575 bytes of SRAM, and 8 bytes of EEPROM. The remaining 21 Kbytes of flash memory leaves plenty of room for new or expanded features.

VFO Calibration

The AD9850 spec sheet shows an equation that explains how the frequency is determined. The equation is $FOUT = (TW \times CLKIN)/2^{32}$

FOUT is the desired output frequency, CLKIN is the input clock reference frequency (here 125000000), and TW is a 32-bit integer tuning word. Let's say that the clock is functioning perfectly

for 7.050 MHz. Rearrange the equation terms, with FOUT = 7050000, and CLKIN = 125000000, so TW = (7050000 × 4294967296)/125000000 TW = 242236155.4944, which truncates to an integer.

It follows that your measured frequency using that tuning word (TW) and the other constants produces the exact desired frequency. You can rearrange the equation to 242236155.4944 = FOUT × 34.359738368, where the right side number is the tuning constant.

Our frequency output was a little off the mark. When we plugged in the actual output from the VFO for FOUT, and changed our tuning constant to 34.35910479585483, our output frequency was dead on. You will need to make a similar adjustment, using either a frequency generator or an accurate receiver to determine your offset multiplier. Near the top of the program code, you will see a line that says

```
#define MYTUNINGCONSTANT
34.35910479585483 // Your calculated
TUNING CONSTANT
```

Once you determine the tuning constant for your VFO, you can replace that constant with yours in the program code line above. Now recompile, upload the new version of the code to the Nano board, and you're done!

Using the Transceiver

Power your rig with either a 9 V (for 1.8 W output) or 12 V (for 3 W output) supply. You may want to put a heatsink on the power transistor if it gets too hot. I use a 13.8 V power supply when at home and a small 12 V SLA battery in the field.

The AD9850 draws its power from the Nano board. The Nano can accept voltages between 6 V and 18 V, but we regulate it to 5 V. We added a Mini 360 voltage regulator U1 (see Figure 7) to the Nano/VFO board to reduce the stress on the Nano regulator.

When you turn the rig on, there is a brief "splash" screen, then the display looks as it does in Figures 1 and 9. The top number displays frequency. The number on the second line is the frequency increment. That is, as you rotate the encoder, each detent changes the frequency by the amount of the frequency increment value (100 Hz shown). The text word towards the right side of the second line is a reminder of the frequency limits that apply to US hams on the 40 meter band. For example, if you tuned down to 7024.990 kHz, the text word changes to EXTRA because you must hold an Amateur Extra class license to operate on that frequency.

The frequency and increment values are read from EEPROM when the rig powers up. If you tune to a new frequency and stay on that frequency for more than 60 seconds, that frequency and increment value are written to EEPROM automatically. If another minute passes and you have not changed frequency, the frequency and increment values are not updated. The reason for not updating is because EEPROM has a finite write life of about 100,000 cycles before it loses reliability. If you had left your rig on, and we didn't check for a frequency change, after about 70 days of continuous operation the EEPROM could become unreliable. Using our approach reduces this possibility.

The next time you apply power to the rig, the frequency and increment values are read from EEPROM and sent to the display. This and the band edge markers are done in software, so you can modify this feature if you wish. The code is well commented.

Conclusion

Driving forces behind this project are to show that a viable transceiver need not be expensive, and to encourage more hams to use microcontrollers in our hobby. With an effective antenna and favorable conditions, 3 W is enough to work the world. We think you'll get a sense of pride sending *RIG HR IS HB*.

You need not be an expert programmer to use microcontrollers.[5] The Arduino family of microcontrollers is open source, so there is much free plug-and-play software available. Once you start programming, you may wish to add an electronic keyer, "canned" contest messages, battery voltage reading, clock, and S meter to the transceiver.[6]

Notes

[1]The Arduino programming software from **arduino.cc/en/Main/Software**

[2]**www.arrl.org/qst-in-depth**

[3]Our PCB uses plated through holes and is silkscreened. See the assembly instruction manual for details and availability.

[4]Jack Purdum, Dennis Kidder, *Arduino Projects for Amateur Radio*, pp 439 – 477, ARRL Item no. 5007. Telephone toll-free in the US 888-277-5289, or 860-594-0355, fax 860-594-0303; **www.arrl.org/shop/**; **pubsales@arrl.org**.

[5]If you have no programming experience, see Jack Purdum's *Beginning C for Arduino*, 2nd Edition, 2015.

[6]Glen Popiel, *Arduino for Ham Radio*, ARRL Item no. 0161, available from your ARRL dealer, or from the ARRL Store. Telephone toll-free in the US 888-277-5289, or 860-594-0355, fax 860-594-0303; **www.arrl.org/shop/**; **pubsales@arrl.org**.

All photos courtesy of the authors.

Dr Jack Purdum, W8TEE, retired from Purdue University College of Technology in 2009. He authored 18 programming texts, including *Arduino Projects for Amateur Radio*, and continues writing about various programming and ham radio topics. Jack is a Life Member of the ARRL and has been licensed continuously since 1954. You can reach him at **jjpurdum@yahoo.com**.

Farrukh Zia, K2ZIA, was licensed in 1987 but his homebrewing passion dates to the '70s. He earned an MS and PhD in Electrical and Computer Engineering from Syracuse University, and was a member of the Syracuse University Amateur Radio Club. Farrukh currently teaches Embedded Systems, Wireless Communication, Robotics, and Computer Technology at New York City College of Technology, City University of NY. He enjoys building low-cost and open-source robots and other embedded systems projects as well as low-power radio circuits and accessories. You can reach Farrukh at 799 Carpenter Rd, North Brunswick, NJ 08902-2231 or **farrukh.zia@usa.net**.

Dennis Kidder, W6DQ, was first licensed in 1969. In 2007, he was granted the call sign of one of his high school Elmers, Chek Titcomb, then W6DQ. You can reach Dennis at **w6dq@arrl.net**.

For updates to this article, see the *QST* Feedback page at www.arrl.org/feedback.

Feedback

■ In "Meter Bands and Megahertz — Why We Use Both" by John Stanley, K4ERO, in the February issue of *QST*, Table 1 shows 0.400 – 0.410 as having a wavelength of 0.7477 – 0.7496 meters. The frequency should have been listed as 400 to 401 MHz. The comment about this being close to our present 472 to 479 kHz allocation should instead have mentioned our present 420 to 450 MHz band. Finally, in Table 2, the label at the top of the left-hand column should have read "Band, *Meters*."

■ In the January 2016 "Hints and Kinks" column, the e-mail address for Art Horovitch, VE3AIH, author of the "Using Heil Headsets" hint, is incorrect. Art's correct e-mail address is **suzanandart@gmail.com**.