QEX

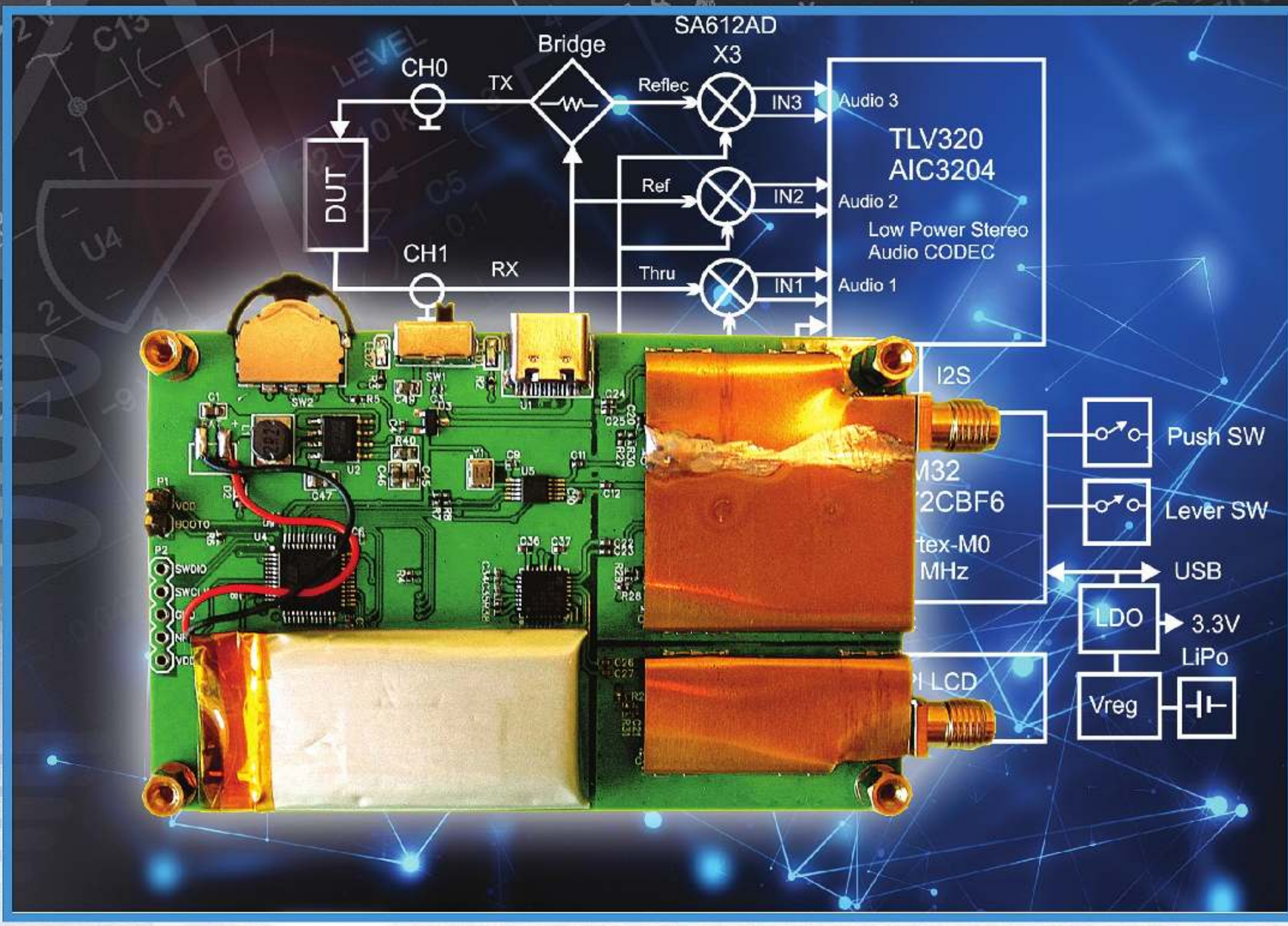
$7

January/February 2020

www.arrl.org

A Forum for Communications Experimenters

Issue No. 318



WB9LVI reviews an ultra-low cost VNA.

ARRL The national association for AMATEUR RADIO®
225 Main Street
Newington, CT USA 06111-1400

APRS® / D-STAR®

TH-D74A 144/220/430 MHz Tribander

The TH-D74A represents the ultimate in APRS and D-STAR performance. KENWOOD has already garnered an enviable reputation with the TH-D72A handheld APRS amateur radio transceiver. Now it has raised the bar even further with the TH-D74A, adding support for D-STAR, the digital voice & data protocol developed by the JARL, and enabling simultaneous APRS and D-STAR operation – an industry first.

- APRS compliance using packet communication to exchange real-time GPS position information and messages
- Compliant with digital/voice mode D-STAR digital amateur radio networks
- Built-in high performance GPS unit with Auto Clock Setting
- Wide-band and multi-mode reception
- 1.74" (240 x 180 pixel) Transflective color TFT display
- IF Filtering for improved SSB/CW/AM reception
- High performance DSP-based audio processing & voice recording
- Compliant with Bluetooth, microSD & Micro-USB standards
- External Decode function (PC Decode 12kHz IF Output, BW:15 kHz)
- Free software for Memory and Frequency Control Program
- Data Import / Export (Digital Repeater List, Call sign, Memory Channel)
- Four TX Power selections (5/2/0.5/0.05 W)
- Dust and Water resistant IP54/55 standards

APRS (The Automatic Packet Reporting System) is a registered American trademark of WB4APR (Mr. Bob Bruninga).
D-Star is a digital radio protocol developed by JARL (Japan Amateur Radio League).

KENWOOD

Customer Support/Distribution Customer Support:
(310) 639-4200 Fax: (310) 537-8235

www.kenwood.com/usa

ISO9001 Registered

ADS#38619

QEX

QEX (ISSN: 0886-8093) is published bimonthly in January, March, May, July, September, and November by the American Radio Relay League, 225 Main St., Newington, CT 06111-1494. Periodicals postage paid at Hartford, CT and at additional mailing offices.

POSTMASTER: Send address changes to: QEX, 225 Main St., Newington, CT 06111-1494
Issue No 318

Publisher
American Radio Relay League

Kazimierz "Kai" Siwiak, KE4PT
Editor

Lori Weinberg, KB1EIB
Assistant Editor

Scotty Cowling, WA2DFI
Ray Mack, W5IFS
Contributing Editors

Production Department

Steve Ford, WB8IMY
Publications Manager

Michelle Bloom, WB1ENT
Production Supervisor

Sue Fagan, KB1OKW
Graphic Design Supervisor

David Pingree, N1NAS
Senior Technical Illustrator

Brian Washing
Technical Illustrator

Advertising Information Contact:

Janet L. Rocco, W1JLR
Business Services
860-594-0203 – Direct
800-243-7768 – ARRL
860-594-4285 – Fax

Circulation Department

Cathy Stepina, *QEX Circulation*

Offices

225 Main St., Newington, CT 06111-1494 USA
Telephone: 860-594-0200
Fax: 860-594-0259 (24 hour direct line)
e-mail: **qex@arrl.org**

Subscription rate for 6 issues:

In the US: $29;

US by First Class Mail: $40;

International and Canada by Airmail: $35

Members are asked to include their membership control number or a label from their QST when applying.

In order to ensure prompt delivery, we ask that you periodically check the address information on your mailing label. If you find any inaccuracies, please contact the Circulation Department immediately. Thank you for your assistance.

January/February 2020

About the Cover

Dr. George R. Steber, WB9LVI, reviews a tiny vector network analyzer (VNA) introduced for about US$50 and compares its capabilities with full-featured lab-grade analyzers that cost thousands of dollars. An RF VNA is the instrument of choice for measuring the electrical parameters of antennas, components, filters and more. Dr. Steber describes his experiences with the tiny VNA, termed a NanoVNA. He begins with a short description of some technical specifications, and relates how he acquired the NanoVNA. He then includes historical details on the evolution of the product. Next, he describes the general architecture of the instrument, and finally he describes the operation of the unit and including examples.

In This Issue

Features

Index of Advertisers

The purpose of *QEX* is to:

1) provide a medium for the exchange of ideas and information among Amateur Radio experimenters,

2) document advanced technical work in the Amateur Radio field, and

3) support efforts to advance the state of the Amateur Radio art.

All correspondence concerning *QEX* should be addressed to the American Radio Relay League, 225 Main St., Newington, CT 06111 USA. Envelopes containing manuscripts and letters for publication in *QEX* should be marked Editor, *QEX*.

Both theoretical and practical technical articles are welcomed. Manuscripts should be submitted in word-processor format, if possible. We can redraw any figures as long as their content is clear. Photos should be glossy, color or black-and-white prints of at least the size they are to appear in *QEX* or high-resolution digital images (300 dots per inch or higher at the printed size). Further information for authors can be found on the Web at **www.arrl.org/qex/** or by e-mail to **qex@arrl.org**.

Kazimierz "Kai" Siwiak, KE4PT

Perspectives

Morse Challenge Initial Results, and QEX Changes

The use of a Morse paddle to input text to a personal computer in Morse code seemed like a promising solution to the age-related and disabilities-related keyboard handicap. Our *Morse Input Design Challenge* to the amateur radio community was inspired by a letter from Dave "Doc" Evison, W7DE, who was experiencing age-related difficulties with typing skills. He's not alone. Doc reasoned that the amateur radio community is overflowing with exceptional people who are skilled technically, resourceful, and compassionate. As a result we opened the *Challenge* with an announcement and rules in the May / June 2019 issue of *QEX*, and in *QST*. The *Challenge* ended on December 1, 2019, and the response has been gratifying. While all of the entries are still under review, we can announce that the earliest complete entry was received from Joseph M. Haas, KEØFF. Joe will receive a free one-year *QEX* subscription (or extension of his subscription). Other winners will be announced in the next issue of *QEX*, and you can expect articles describing the entries to appear soon.

We begin the new year by noting changes in *QEX*. The *QEX* style was recently changed to increase the readability of the text. The changes were in response to a reader's request — we hope you agree that this is a significant improvement. We solicit your comments and suggestions. There are further improvements and enhancements planned for *QEX* later in 2020.

In This Issue

Gerald T. Whitney, KG2BK, in a *Technical Note* determines matching components with his T-network calculator.

John L. Stensby, N5DF, in a *Technical Note* calculates coax loss directly from impedance measurements.

James A. Koehler, VE5FP, home-builds an inexpensive vector impedance analyzer that is based on a 32-bit single-board processor sub-system and operates up to 600 MHz.

Michael D. Lewis, K7MDL, provides life-extending support for multiple transverters.

Dr. George R. Steber, WB9LVI, relates his experiences with the recently available tiny NanoVNA.

Scott H. Cowling, WA2DFI, revives his "Hands on SRD" column with an introduction to the TangerineSDR.

Writing for *QEX*

Keep the full-length *QEX* articles flowing in, or share a **Technical Note** of several hundred words in length plus a figure or two. Let us know that your submission is intended as a **Note**. *QEX* is edited by Kazimierz "Kai" Siwiak, KE4PT, (**ksiwiak@arrl.org**) and is published bimonthly. *QEX* is a forum for the free exchange of ideas among communications experimenters. The content is driven by you, the reader and prospective author. The subscription rate (6 issues per year) in the United States is $29. First Class delivery in the US is available at an annual rate of $40. For international subscribers, including those in Canada and Mexico, *QEX* can be delivered by airmail for $35 annually. Subscribe today at **www.arrl.org/qex**.

Would you like to write for *QEX*? We pay $50 per published page for articles and Technical Notes. Get more information and an Author Guide at **www.arrl.org/qex-author-guide**. If you prefer postal mail, send a business-size self-addressed, stamped (US postage) envelope to: *QEX* Author Guide, c/o Maty Weinberg, ARRL, 225 Main St., Newington, CT 06111.

Very best regards,

Kazimierz "Kai" Siwiak, KE4PT

Dr. George R. Steber, WB9LVI

11105 N. Riverland Ct., Mequon, WI 53092; **steber@execpc.com**

An Ultra Low Cost Vector Network Analyzer

A review and description of the author's experiences with the tiny NanoVNA.

An RF vector network analyzer is the instrument of choice for measuring the electrical parameters of antennas, components, filters and more. However, professional full-featured analyzers found in many labs cost thousands of dollars and are out of reach for most experimenters. But in recent years, new and lower cost designs have emerged, creating a niche market that offers similar products for those wishing to experiment with this technology.

This highly competitive market offers a variety of products through non-traditional channels, web stores or auction sites to make the prices amazingly low. Recently an ultra-low cost vector network analyzer (VNA) offering many of the features of its big brothers appeared on the scene. This battery powered device covers 50 kHz to 900 MHz and includes a touch screen LCD panel. In this study we will take a close look at this unusual instrument. You may be surprised at the performance contained in this tiny package.

A Review of Network Analyzer Uses

Let's review why a network analyzer is useful to the RF designer. Mainly it is used to characterize or measure the response of devices at RF or even at microwave frequencies. Obtaining the response of the device or network makes it possible to understand how it will work within the circuit for which it is intended. Typically, RF analyzers are used to measure a variety of things ranging from antennas, filters, frequency sensitive components such as crystals, to devices such as transistors, amplifiers and mixers.

Most common in the RF network analyzer family are the Scalar Network Analyzer (SNA) and the Vector Network Analyzer (VNA). The SNA is the simpler of the two, measuring only the amplitude (scalar) properties of the components and devices. A VNA measures both amplitude and phase and generally can measure more parameters than a SNA. The SNA, in spite of its limitation, can be very useful in many situations, and can be easier to setup and use. An in-depth study of a typical low cost SNA, the NWT70, was recently presented in *QEX* **[1]**. The capabilities and construction of VNAs are discussed in more detail in **[2, 3]**. For those interested in the fundamental principles of VNAs, the application note **[4]** by Keysight Technologies is an excellent way to begin.

Prices of VNAs from the major names have been very high, often many thousand dollars or more. Commercial work being done with these units probably justifies this expense. But for many of us, fortunately, there is a second tier available with more affordable prices. Within this group, new

Figure 1 — Contents of the NanoVNA package as it was received. The parts came inside a plastic box with a USB cable, two RF cables, and a complete calibration kit including thru, 50 Ω, short, and open fittings as seen along the bottom.

models seem to be introduced at a rapid pace.

However, I was not prepared to see a tiny VNA introduced for about $50. That got my attention! The vendor claimed it had many of the features of the more expensive units — all while fitting into the palm of your hand.

What follows is a description of my experiences with the tiny VNA, termed a NanoVNA. We'll begin with a short description of how the NanoVNA was acquired and some technical specifications. This will include historical details on the evolution of the product. Next, there will be a section describing the general architecture of the instrument. The sections after that will describe the operation of the unit and include examples.

The Tiny NanoVNA

One day I received news in my email of a low cost VNA at the incredible price of US$50. Naturally I was interested. It described a NanoVNA with a 2.8 inch LCD with built-in battery. Specifications were as follows.

- Size: 54 mm by 85.5 mm by11 mm (not counting the connectors and switches).
- Measurement frequency: 50 kHz to 300 MHz (50 kHz to 900 MHz with extended firmware).
- RF output: –13 dBm (maximum –9 dBm)
- Dynamic Range: 70 dB (50 kHz to 300 MHz), 60 dB (300 MHz to 600 MHz) and 50 dB (600 MHz to 900 MHz) with extended firmware.
- Port SWR: < 1.1
- Display: 2.8 inch TFT (320 by 240), Touch screen.
- USB interface: 5 V, 120 mA.
- Battery: 300 mAh or 500 mAh. Maximum charging current 0.8 A.
- Number of scanning points: 101 fixed.
- Display tracking: 4, Marking: 4, Save: 5.
- Measured S-parameters: S11, S21, SWR, phase, delay, Smith chart.

Simple PC control software that can plot and export *Touchstone* (SNP) files for use by other programs was also included.

Some background research was needed before rushing off to place an order. This involved translating some Japanese and Chinese user groups and web sites, which proved very helpful. It seems that the original NanoVNA was designed by a Japanese experimenter with web name **edy555**. He placed all of his source code on **github.com** and began to offer a kit on his own web site (in Japanese). Apparently, after some time he sold all of his kits, lost interest and discontinued the project.

Another entrepreneur based in China, who goes by the web name **hugen** or **hugen79**, took up the mantle. He made some important changes to the circuit. This included improving the battery charging circuit, boosting the voltage on the RF mixers, changing to a type-C USB interface from the microUSB, and replacing the VCXO with a TCXO.

Another major improvement involved redesigning the PCB to include shielding, which had a definite advantage over the **edy555** design. He also found and fixed a major calculation error in the firmware. Finally he modified the firmware to use harmonics of the Si5351 square wave oscillator, which allowed extended operation to 900 MHz from the original 300 MHz. He recently made all of the firmware code available on the web **[5]**.

At this point I was ready to take a chance and see for myself how this very tiny palm-sized VNA worked. After looking around the internet I decided to order one from the Ham Radio Store on AliExpress for about US$50. I chose this store because the vendor had sold several units world-wide and there was no negative feedback. But the fact that I was the first US buyer made me a bit nervous. In any case after two weeks, a nice package arrived. A photo of the contents is shown in Figure 1. The package items are as follows.

- NanoVNA host (with 500 mAh battery) x1.
- USB Type-C data cable x1.
- SMA male-to- male RG174 RF 30 mm cable x2.
- SMA simple calibration kit x1 (OSL).
- SMA female-to-female connector x1.

All of the material was nicely fitted into a plastic case, which can be used for storing the items when not in use. There was no manual or software. After contacting the vendor, a link was provided to the manual **[6]**, software and firmware. Caution, the manual is a bit hard to follow.

Opening the unit, I was pleasantly surprised to see a well designed PCB with isolated ground planes and RF shields. A photo of the PCB is shown in **Figure 2**. The switches and the type-C USB port are along the top. The large object on the lower part is the battery. The back cover is also copper clad FR4 PCB for some additional shielding. A close-up view of the front of the unit is shown in **Figure 3**. CH0 is the transmit output and CH1 is the receiver input connector.

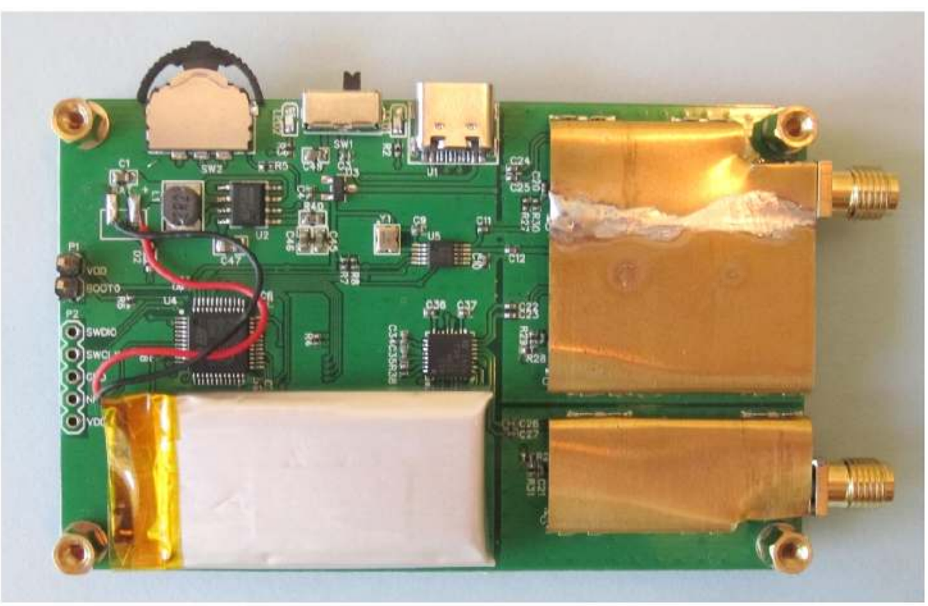


Figure 2 — Bottom side of the NanoVNA PCB. Along the top row are switches and a type–C USB connector. The right side is nicely shielded over the SMA connectors. A small battery fits snuggly along the bottom. On the left side is a port P1 used when installing new firmware.

The NanoVNA Clones

After receiving my NanoVNA, I began looking around on auction sites for similar items. I was surprised to find many clones available. One clone, a white Gecko model, was in an auction that I was lucky enough to win for US$28. Looking at the unit closely, it was obvious that this was a copy of the original one I had purchased. Its components and PCB were very different.

Finally, I found a group on the web **[7]** that was just starting out and devoted to the NanoVNA. As discussed in this group, many users had bought cloned versions of the VNA. But how can you tell the difference? Short answer is that it is difficult to do so, especially for the latest clones. A visit to this user group may help you if you are thinking of getting a NanoVNA for yourself. But remember: *caveat emptor*.

Initial Observations of the NanoVNA

I was surprised to find that there is no case per se around the NanoVNA. It consists essentially of a sandwich of PCBs that are held together by four screws in the corners. The top side had a milled opening for the LCD touch screen. In spite of this unusual design, I did not have any problems using it. I particularly liked the cables and calibration standards supplied with the unit. The 'thru', 50 Ω load, 'open' and 'short' calibrations elements are shown in Figure 1 at the bottom of the plastic case. My unit also had a battery installed that seems to work well.

Basic Operation of the NanoVNA

The signal processing in the NanoVNA is very similar to the network analyzer of T. C. Baier, DG8SAQ, described in *QEX* [3] and the antenna analyzer by M. Knitter, DG5MK, described in *QEX* **[8]**. It also shares some heritage from the Y. Kuchara, EU1KY, antenna analyzer V3, which is also an open source project **[9]**.

Figure 4 shows a block diagram of the NanoVNA based on material in the user manual and schematic. The front end, consisting of a resistive bridge and mixers, shown here is becoming common for these types of analyzers. The resistive bridge is balanced for 50 Ω, which works well in this application. Its main drawback is that it has a limited range if the unit is used for measuring impedances. It is very similar to the DG8SAQ and DG5MK resistor bridges. On the other hand the EU1KY unit has a slightly different bridge that permits operation over wider impedance ranges.

As seen from the block diagram, the signals are processed by three RF mixers, in this case the venerable SA612. This permits the unit to operate without an IF strip — by processing of the reflective, reference and thru signals in the STM32 processor. Most units today use a similar process.

Using SA612 mixers for this type of work has always been a bit controversial. This is because of the low signal levels and the possible distortion of these mixers at high signal levels. Despite this, the NanoVNA, the DG8SAQ and the EU1KY units have found it possible to work with these devices. But the DG5MK unit has eschewed them in favor of wider dynamic range digital mixers. During operation, the

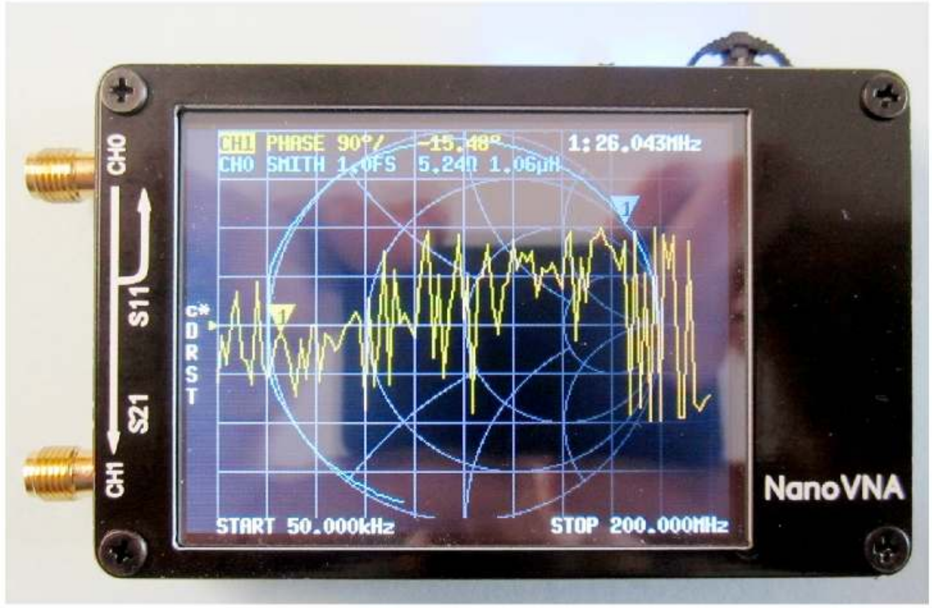


Figure 3 — Front side of the unit showing two traces. Some firmware versions can show up to four traces.

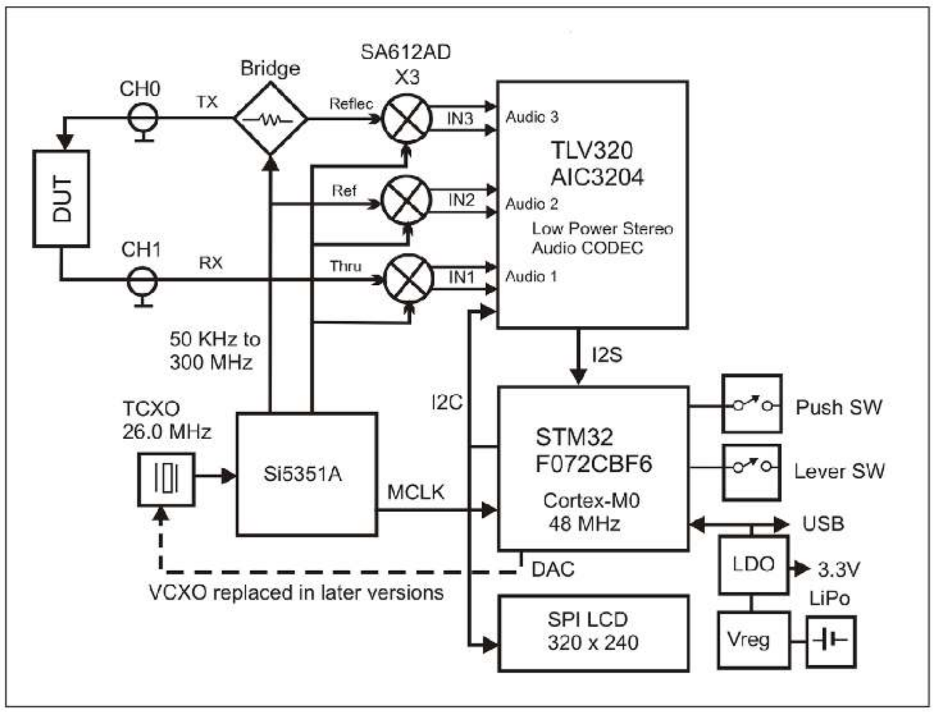


Figure 4 — Basic block of the NanoVNA. The DUT is on the left.

mixers in the NanoVNA are fed two signals approximately 10 kHz apart, that provides audio signals to IN1, IN2, and IN3, which are processed by the A/D converters of the TLV320. As will be shown later on, the digital filtering done in the software effectively creates a narrow band filter.

The main signal source is a Si5351A square wave generator. It is controlled by the STM32 processor. A square wave can be a problem if the harmonics are not accounted for in the signal processing. A good discussion of this processing of square waves can be found in Knitter [8]. Note, as shown in Figure 4, that the original design used a VCXO and was later changed to a 26 MHz TCXO. Consequently the accuracy of the resulting frequency of the Si5351A is dependent on this fixed frequency. Some users have reported significant errors in frequency produced by their unit while others had no problems. My unit was within a couple of Hz at 100 MHz. But this is no guarantee that other units will be this close. As of this writing, there have been no plans to include calibration in the firmware.

Firmware Updates for the NanoVNA

Many users have found that the firmware in the NanoVNA is not the latest version. Thus to get the latest features one must download the latest firmware to the unit. The downloaded firmware can be installed via the USB port. Options for updating include changing from a four channel display to two channel display with larger characters and to fix some calibration interpolation issues. Firmware updates and software necessary to perform the update can be found on this web site [6]. In order to perform a firmware update you must first short out the "boot" pads on the main PCB. They are shown in Figure 2 on the left side. The author has soldered in two pins on port P1 for this purpose.

Using the NanoVNA

When the power switch is thrown, a blue LED indicates that the NanoVNA is powered on. It will stay on for a short time after power off too. Initially it should be connected to a USB port for several minutes to make sure the battery is charged. The touch-screen controls are on the right side of the screen. I had trouble using this feature with my fingers until I found that a soft wooden stick worked reliably as a substitute digit. Alternatively, one could use the rotary switch to perform the same function. But the touch method seems easier to use.

The unit's menu operation is too complicated to discuss here, but suffice it to say that there is a learning curve required for navigating through the various menus. More details are given in the user manual.

It is important to recognize at the start that the unit captures only 101 points regardless of any other setting. So if you have a large span, say 300 MHz, then you will have a frequency resolution of 2.97 MHz. Anything between these frequency points will be missed! So, if you are searching for specific detail, like for a crystal frequency, it can be overlooked. Fortunately, many measurements involve slowly changing variables so in these cases there is no problem. Never-the-less, you need a bit of intuition to use this device.

It should also be realized that the audio signals captured by the TLV320, which has a large dynamic range, will be limited by noise. This will limit the dynamic range to about 70 dB on the low frequency end and less at higher frequencies. In many applications this is sufficient but, of course, not close to that obtained in commercial units.

The device requires calibration to work properly. There are five calibration memories, cal0 to cal4, that can be saved and recalled. Cal0 is returned first when the unit is turned on. Before you start, you need to decide on the frequency range that will be stored with the calibration. The calibration process involves connecting — in turn — a short, open, load, thru and isolation to the inputs. This is discussed in more detail in the manual. Having calibrated at least one of the cal memories the unit can be used. It will retain these calibration settings until replaced with new ones.

Free *NanoSharp* software is available for this unit on the firmware site mentioned before, but it needs *Windows Framework 4.0* to run. Many older computers, for example those using WinXP, do not have this installed. A nice feature of the software is that it allows the saving of *Touchstone* (SNP) files that can be read in and analyzed by other software. But it won't read SNP files from other programs unless they are in the MA format.

Battery operation of the unit permits taking it along to make measurements directly at the antenna. In my WSPR work, various dipoles for different bands are often cobbled together. I have found the NanoVNA to be very handy for checking these antennas. The SWR of my WSPR 17 meter dipole was quite close compared to the value found with my AIM4160 antenna analyzer.

S21 Measurement of a Low Pass Filter

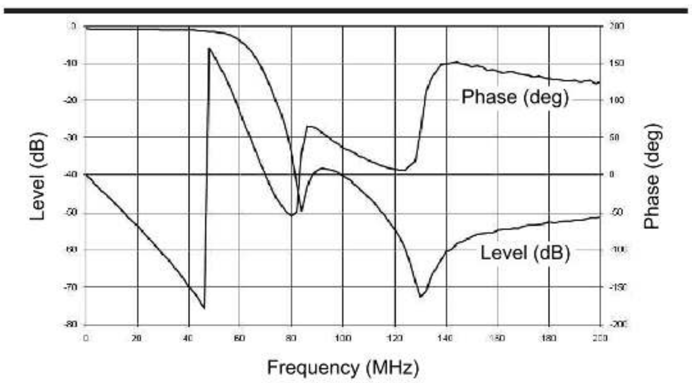
When testing components like filters, crystals, and other devices, it important that the unit be normalized (calibrated) to include the effects of the fixture and connecting cables. In other words, perform the calibration with exactly the same test setup used for measuring the DUT but with the DUT replaced with a straight-though connection. This has been done for all the subsequent tests.

I tested a seven-pole Coilcraft P7LP-606 60 MHz low-pass filter using my calibrated NanoVNA. To find the S21 curves of the DUT, it was placed between CH0 and CH1 with appropriate scan frequencies chosen. The NanoVNA was then set to obtain the S21 variations caused by the device.

Results of a 200 MHz scan are shown in **Figure 5**. Plotted here are the magnitude (left scale, upper trace) and the phase (right scale). The results compared favorably to the curves on the manufacturer's data sheet that were taken with an HP8753D network analyzer. This is amazing when you realize that this NanoVNA measures only 101 points and has only 70 dB of dynamic range.

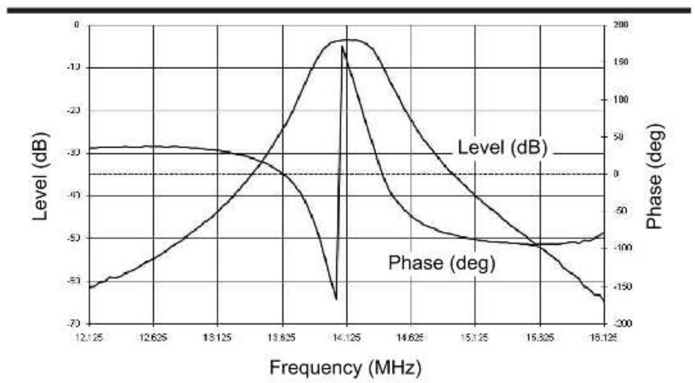
S21 Measurement of a Band-Pass Filter

Next I wanted to see how the NanoVNA worked on an old 14 MHz band-pass filter. The NWT70 scalar network analyzer studied in *QEX* [1] had problems with this



QX2001-Steber05

Figure 5 — S21 magnitude (top trace, left scale) and phase (right scale) response of a Coilcraft 60 MHz low-pass filter.



QX2001-Steber06

Figure 6 — S21 magnitude (top trace, left scale) and phase (right scale) response of a 14 MHz band-pass filter.

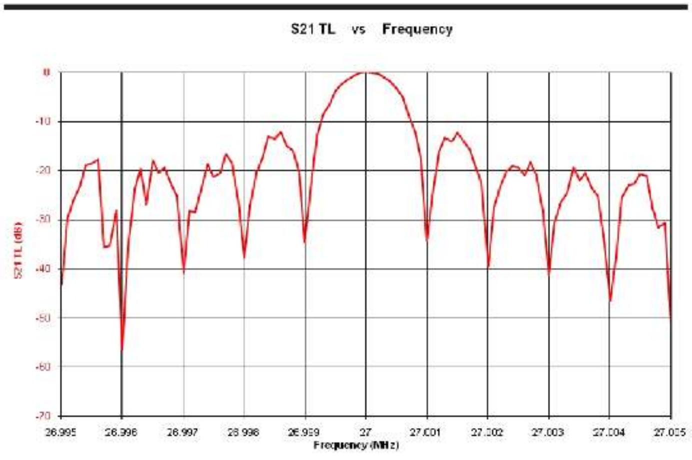


Figure 7 — NanoVNA RBW test result shows the response of CH1 input to a 27 MHz sine wave. A narrow 10 kHz scan was used to show the filter response.

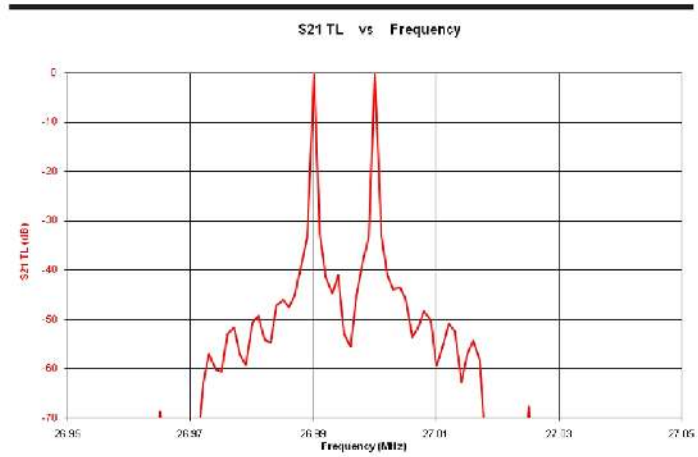
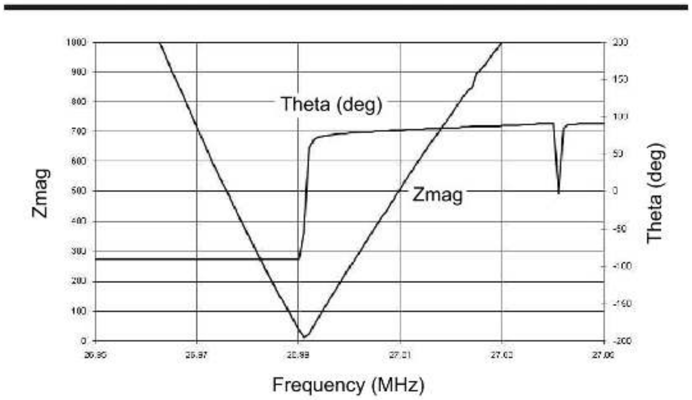
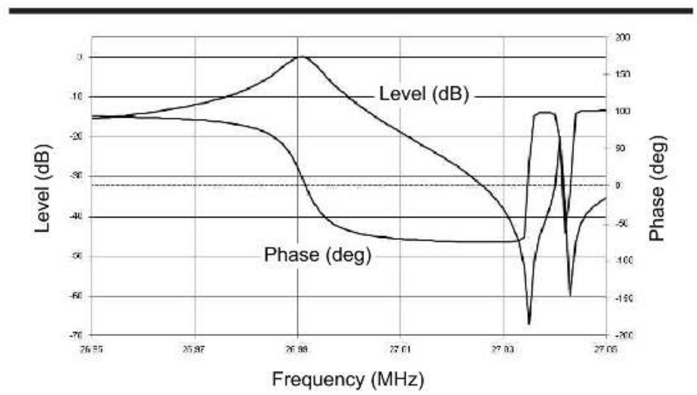


Figure 8 — A wider frequency scan of the RBW filter, shows the double-peak response of the CH1 input to a 27 MHz sine wave. A 100 kHz scan shows the real and image responses.



QX2001-Steber09

Figure 9 — Impedance plot of a 27 MHz crystal connected as a direct load to the CH0 port. The 12 Ω dip is around the resonant point.



QX2001-Steber10

Figure 10 — Impedance of the same 27 MHz crystal as in Figure 9, but this time connected as a thru element. The resonant point is at the peak where the phase is 0°.

type of filter because of harmonics in the signal that affected the input measurement circuit. As can be seen in Figure 6, the NanoVNA did not have this problem and was free of harmonics.

Resolution Bandwidth of the NanoVNA

At this point I became obsessed with finding the resolution bandwidth (RBW) of the NanoVNA. RBW is a means of specifying the smallest frequency sampled and is commonly used with spectrum analyzers. The smaller this value, the greater will be the resolution of the analyzer.

Studying the open software module *dsp.c*, it became obvious that the 16-bit audio samples were processed by a sine cosine filter. It used 48 samples and operated like the familiar Goertzel Fourier transform algorithm. If the audio signals have a correctly chosen frequency it would be possible that they would fall right in the middle of a FFT bin. This would minimize the so called 'spectral leakage' that significantly influences measurement precision.

Assuming this is done, all the digital signal processing could be done in the STM32. Hence, from the Fourier transform, the magnitude and phase difference between channels could be found, hardware errors corrected, OSL correction applied, and the final S11 or S21 parameter calculated.

Based on the software module, it appeared that 48 samples were taken at 48,000 samples per second. For Goertzel's Fourier algorithm, this implies that the resulting filter would have a bandwidth of 1 kHz.

There turned out to be an easy way to verify this conjecture in the hardware. From the block diagram of Figure 4 it is apparent that CH1 forms the input of an RF mixer. If a sine signal was applied to CH1 and a suitably small span is chosen, the NanoVNA will take its own picture of the RBW filter.

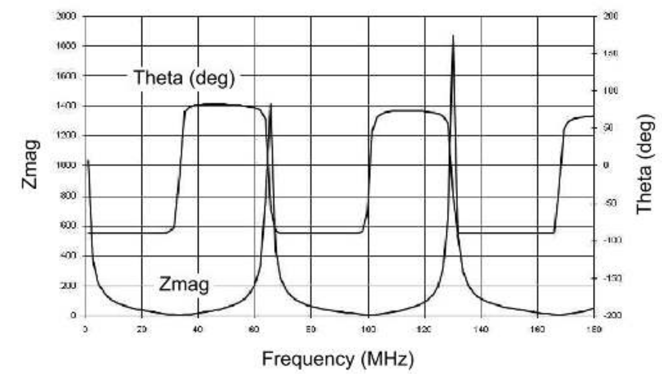
To test this hypothesis, a 27 MHz signal was applied to CH1. The result is shown in **Figure 7**. Here we are using a span of 10 kHz. We see a nicely shaped frequency bin around 27 MHz. Measurements indicate that it has a 3 dB bandwidth of 1 kHz.

Using a wider span, 100 kHz, we can see the two peaks of the DSB signal as shown in **Figure 8**, one real and the other an image. Of course, the image signal would not appear in normal operation of the NanoVNA.

Testing Crystals

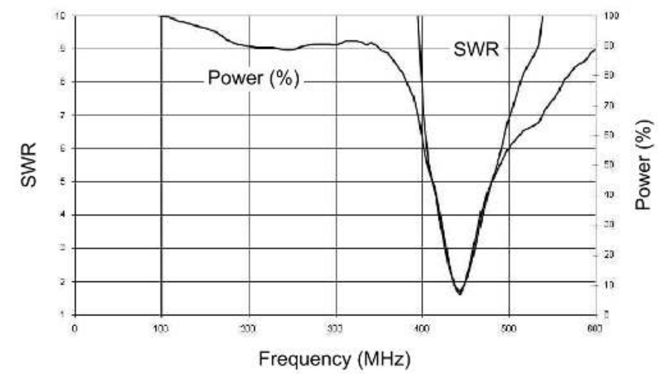
Crystal checking is very easy with the NanoVNA. But recall that, because of the limited number of points in a scan (101), it's possible to miss the resonant point if the scan frequency range is too large. Also the RBW filter will limit frequency resolution to about 1 kHz or so.

A 27 MHz crystal was tested in both the impedance and filter modes. **Figure 9**



QX2001-Steber11

Figure 11 — Impedance response of a 63-inch long open-ended coax cable. The high impedance peaks are probably not accurate in amplitude.



QX2001-Steber13

Figure 13 — SWR and reflected power response of the antenna in Figure 12. While the 430 MHz band seems okay, the lower 144 MHz band is missing. See text for details.

Figure 12 — A multi-band antenna advertised for 144 MHz, 430 MHz and 1200 MHz.

shows the impedance (V-shaped curve, left hand scale) and phase angle (right scale) for the crystal connected across the CH0 port. The low impedance point of about 12 Ω is easily seen.

The same crystal in transfer function mode (S21) is shown in **Figure 10**. The primary resonant point is located below the peak (upper trace, left scale) at 0° (right scale). The peaks to the right are due to the other artifacts of the crystal.

Impedance Measurement

Impedance measurements with the NanoVNA can be made but the accuracy will vary with the magnitude of the impedance. This is a basic characteristic of resistance bridges [8], as they are most accurate near their design value, in this case 50 Ω. The useable range around this design point is open to some discussion. For the NanoVNA it was found that in the range of 12 Ω to 200 Ω the accuracy was very good. And the accuracy was acceptable from about 5 Ω to 500 Ω. Beyond this range the measurements may not be reliable.

For fun, the impedance of a 63-inch piece of coax with one end open was measured. The scan range is 180 MHz. Shown in **Figure 11** are the impedance magnitude (left scale peaky trace) and phase (right scale) responses. We can see that the peaks in the impedance are well above 1000 Ω in some cases. So these values are probably not accurate.

This example illustrates one of the dangers of using such an analyzer. If the impedance values are presented in conventional units of capacitance, inductance and resistance, then these values may be far off from their actual values if the associated impedance is too large or too small.

SWR of a Multi-Band Antenna

In this last example we will take a look at the SWR and reflected power of a small multi-band antenna shown in **Figure 12**. It is claimed to have low SWR at 144 MHz, 430 MHz and 1200 MHz. While the 1200 MHz band is beyond the range of the NanoVNA, the two other frequencies are in range. As can be seen in **Figure 13**, the SWR (left scale) and reflected power (right scale, curve between 100 and 600 MHz) have a minimum near 430 MHz but none at 144 MHz. This is an example of an antenna clone that has been improperly constructed. The lower band is missing!

Summary and Conclusions

This article discussed the operation of an ultra-tiny, very low cost VNA. While this particular unit worked well there are clone versions that may not work so well. Also, since it is an open source project the firmware is constantly evolving. Anyone obtaining such a unit needs to be aware of this and be prepared to make firmware modifications as needed. Those not skilled in the art of making these changes may be in for a lot of frustration.

That said, if you like to play with hardware and software you may find this little gem a wonderful toy for experimentation. However, one must be aware of its limitations.

Because the manuals were originally written in Japanese and Chinese you may find that the translated manuals are very challenging to read. However, this may not pose a problem since most people skilled in this art will probably not use the manual. Plus, there is a user group to call upon if needed.

A more subtle problem is to continually realize that only 101 points are measured on any one scan and that the dynamic range is limited to 70 dB or less. This can be vexing if you are trying to locate and measure a crystal frequency.

The *NanoSharp* software was a bit disappointing since it required *Framework* to operate. But it does a good job and the SNP files are useful. However, there are some additional features I would like to see, as well as software that would permit it to be used with some of my old laptops that are running WinXP.

While this tiny analyzer worked reasonably well, it does not possess the accuracy and dynamic range of its big brothers. Of course the professional models cost quite a bit more than the US$50 for the NanoVNA tested here.

Experimenting with this nano-sized VNA was enjoyable and educational. Hopefully you have learned something about its strengths, weaknesses and applications. A VNA is a nice general purpose instrument for the advanced radio amateur. You need to decide for yourself if a NanoVNA is in your future.

George R. Steber, PhD, WB9LVI, is Emeritus Professor of Electrical Engineering and Computer Science at the University of Wisconsin-Milwaukee. He is now semi-retired, having served over 35 years. George, has an Advanced class license, is a life member of ARRL and IEEE, and is a Professional Engineer. He has also worked for NASA and the USAF.

His last article for QEX *was "An Unusual Multi-Band WSPR Transmitter" (May/Jun. 2019). George also recently penned an article on the hidden story behind "The Discovery of Radio Waves" in the Jan./Feb. 2019 issue of* Nuts and Volts Magazine. *He also wrote a science oriented article on "Dark Energy and the Expanding Universe" in the Mar./Apr. 2019 issue of* Nuts and Volts Magazine.

George still lectures occasionally on science and engineering topics at the University. He is currently involved in cosmic ray research and is developing methods to study them on a global basis. When not dodging protons, pions and muons, he enjoys WSPR/JT9 Amateur Radio, racquet sports, astronomy, and jazz. You may reach him at ***steber@execpc.com*** *with "Nano" in subject line and email mode set to text.*

Notes

[1] G. R. Steber, WB9LVI, "Low Cost RF Scalar Network Analyzers," *QEX*, Nov./Dec. 2018, pp. 3-9.
[2] G. R. Steber, WB9LVI, "An Unusual Vector Network Analyzer," *QEX*, Sep./Oct. 2007, pp. 11-23.
[3] T. C. Baier, DG8SAQ, "A Small, Simple, USB-Powered Vector Network Analyzer Covering 1 kHz to 1.3 GHz," *QEX*, Jan./Feb. 2009, pp. 32-36.
[4] "Understanding the Fundamental Principles of Vector Network Analysis" **https://literature.cdn.keysight.com/litweb/pdf/5965-7707E.pdf**.
[5] NanoVNA source code, **https://github.com/hugen79/nanovna_900**.
[6] NanoVNA manual, firmware and software, **https://drive.google.com/drive/folders/1-JViWLBOlzaHTdwdONX2RP8S4EgWxoND.**
[7] NanoVNA user group, **https://groups.io/g/nanovna-users/messages**.
[8] M. Knitter, "The DG5MK IV Meter – An Accurate Antenna Analyzer," *QEX*, May/June 2017.
[9] Y. Kuchura, EU1KY, VNA source code, **https://bitbucket.org/kuchura/eu1ky_aa_v3/wiki/Home**.

Scotty Cowling, WA2DFI
P.O. Box 26843, Tempe, AZ 85285; **scotty@tonks.com**

Hands-On-SDR

This sixth installment revives the author's series.

It hardly seems possible that it has been three and a half years since my last installment. Time marches on, as they say, and we have exciting new advances in the SDR field to discuss! The last few years have seen a virtual explosion of new SDR hardware available to us: from SDRPlay **[1]** and RTL-DVB **[2]** dongle to Red Pitaya **[3]**, KiwiSDR **[4]**, Kerberos **[5]** and the several variants of LimeSDRs **[6]**. This list is hardly exhaustive, as there are so many from which to choose!

In spite of all the new SDRs on the market, it still seems difficult to find that one SDR that has all the features and capabilities that you want. Some are very inexpensive, but leave you wanting more performance. Some of the more expensive ones leave you wondering why they didn't offer some critical (to you, at least) feature, like a low phase-noise oscillator, for example. In this installment, I will introduce you to a new TAPR **[7]** project, called the TangerineSDR, that will attempt the impossible: to be all things to all users. We will try, at least!

The inspiration for TangerineSDR came from the HamSCI **[8]** group as a request for SDR hardware that could be used as a Personal Space Weather Station (PSWS). Their need resulted in the four-hour Sunday seminar at the TAPR Digital Communications Conference in Albuquerque, NM, on September 16, 2018. At TAPR, our first response was, "Let's find a commercial SDR that we can incorporate into a PSWS kit". Existing hardware would be our best bet for a quick solution. After some research and further consultation with the scientists at HamSCI, it became clear that there was no affordable (less than US $500) solution that met PSWS requirements. What are the requirements, you ask? Wait, not so fast…

What Do We Need to Get Started?

As with each of these columns, I always try to define what you need in the way of knowledge and equipment to get the most out of the "Hands On SDR experience".

I will assume a basic knowledge of direct sampling (DDC and DUC) SDR techniques, as well as some familiarity with the Single Board Computers (SBCs) of the day. For this initial introduction to the TangerineSDR, that is about all you need to bring to the party, except your interest! Future columns will dig deeper into the details. As the new hardware becomes available and firmware and software applications are written, it will be advantageous for you to obtain hardware. But for now, let's dig into the TangerineSDR features and see if it can deliver on our wild claims.

A Bit of History

Back in 2006, TAPR became involved in the production of boards for the openHPSDR project. As one of the participants in that project on both sides (TAPR as a facilitator and the openHPSDR group as a contributor), I think it was an unqualified success! For its day, openHPSDR was a state-of-the-art design, produced a working radio and contributed to a successful commercial SDR company. OpenHPSDR pre-dated FlexRadio System's highly successful direct sampling SDRs; it also pre-dated crowdfunding as a fund-raising method for building expensive hardware. How many of you reading this today sent your money to TAPR for an openHPSDR board before any hardware was even produced? TAPR thanks you for your trust! But I digress. TangerineSDR follows in the footsteps of openHPSDR. We hope to learn from the shortcomings of openHPSDR (yes, there were a few) and expand upon the good ideas that permeated that design.

TangerineSDR will reduce system complexity by using one central FPGA instead of multiple FPGAs distributed across several boards. TangerineSDR will increase performance by eliminating the narrow, slow, unterminated Atlas bus. TangerineSDR will provide high-bandwidth paths between ADC/DAC and the FPGA, as well as high-speed communications channels from the FPGA to the outside world. TangerineSDR will be modular, allowing you to customize the performance (and hence, the cost) of the hardware to a specific task, PSWS being only one of many use cases. Modularity is the key to being all things to all users! I am using this phrase so often, we should use an abbreviation for it: ATAU **[9]**.

Basic Requirements

The PSWS is the initial use case, but there are others that I will cover at the end of this installment. Many of the basic features will be useful to other users, and some will not. Here are the basic PSWS system requirements:

- Dual-channel, synchronous 14-bit direct sampling DDC receiver to 60 MHz
- Eight 192 kHz virtual receiver data streams
- Selectable attenuation and optional pluggable RF filtering to reduce overload
- Switchable on-board noise source for amplitude calibration

- Very accurate time stamping, to within 100 ns
- Accurate frequency accuracy, less than 30 ppb frequency error
- Gigabit Ethernet interface
- Networked receivers capable of secure communications with a global server
- Able to continuously store 24 hours of sample data history in a ring buffer
- 3-axis magnetometer with ~10 nT resolution at 1 sample/second.

We can meet most of these requirements with a USRP **[10]** system, but the cost is high (typically more than US $2000). Two things to note in the above list are the accurate time stamping, which will require a GPS Disciplined Oscillator (GPSDO), and a magnetometer with enough sensitivity to be useful. Neither of these requirements can be met with inexpensive off-the-shelf components. Our solution will be to "roll our own" in an effort to reduce cost and obtain just the right cost/performance ratio that we require.

Pairing with Single Board Computers

The TangerineSDR block diagram is shown in **Figure 1**. One of the unique features of this SDR is the pairing of the SDR-specific hardware (the Data Engine, or DE, and its associated boards) with a computer. We considered using an SoC FPGA in place of the logic-only FPGA and a general-purpose Single Board Computer (SBC), such as a Raspberry Pi or Odroid N2. The SoC FPGA contains a dual-core ARM CPU alongside the FPGA logic on the same silicon die. There are a few problems with the SoC approach. First it is much more expensive, adding more than US $100 to the cost of the DE. A RPi 4 SBC costs US $35-$55, and easily out-performs the dual-core ARM system on the SoC FPGA. The SoC approach also flies in the face of our ability to match required performance to modular hardware. The SoC-based DE might work very well in a simple system with a light CPU load, but if you need more CPU power, you will be locked into the SoC CPU limitations. On the other hand, if we pair with an external computer, we can pick a low-cost, low-performance SBC for "easy" tasks or substitute a full-size i9 desktop system when massive computational power is needed. Using the external computer will also allow us to take advantage of the constant upward trend in capability and downward trend of size, cost and power consumption of tomorrow's SBCs.

High-speed Sample Paths

A closer look at Figure 1 shows the high-speed paths between the three RF Modules along the left side of the DE and the FPGA in the center of the DE. These paths consist of three Low Voltage Differential Signaling (LVDS) channels for each RF Module: two 17-bit input ports to the FPGA and one 14-bit output port for RF Modules 0 and 1, and a more restricted set of pins for RF Module 2. (RF Module 2 will be used mostly as a debug port, so we will wait until

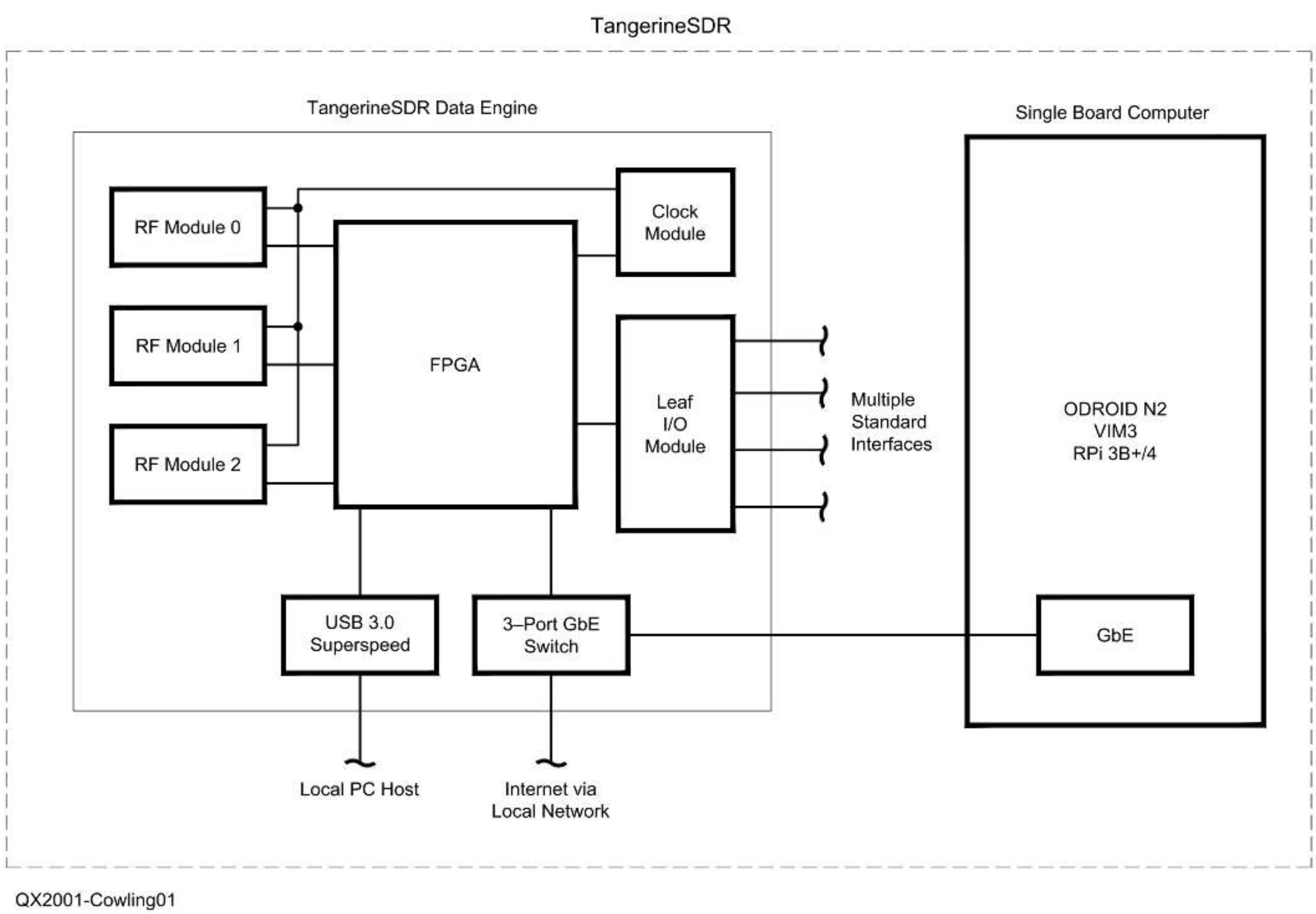


Figure 1 — TangerineSDR system block diagram.

a future installment to describe its use.) The two input ports are used for digitized receive data from an Analog-to-Digital Converter (ADC) and the output port is used to send transmit data to a Digital-to-Analog Converter (DAC). Both ADC and DAC circuits will reside on the RF Modules. Each of these three data paths is capable of nearly 500 MByte/s transfer rate for each RF Module. None save the most expensive SDR models provide this data rate between the antenna and the FPGA. Also keep in mind that all of the signals in the three paths connect to programmable pins on the FPGA. This means that any pins that are not used for data communications between the FPGA and an ADC or DAC can be re-programmed and used for other purposes, including single-ended connections. Since there are 96 of these signals (2 buses X 2 pins/pair X 17 pairs, plus 1 bus X 2 pins/pair X 14 pairs = 96 pins) per module, or 192 pins total, there are plenty of pins available between the FPGA and the RF Modules.

High-speed Communications Interfaces

Okay, so we have buckets of bytes arriving at the input to the FPGA, what are we going to do with all this data? The typical approach is to reduce the data rate using a mathematical process known as decimation. By performing decimation on a stream of data we do, in fact, reduce the data rate. But at the same time we reduce the information content of the stream. It may seem obvious that (for example) a 122.88 Msps data stream contains more data than a 192 ksps data stream, but how we derive the lower-rate stream from the higher-rate stream is not so obvious. We will leave this derivation to a later installment, but we will discuss the reason why we need to reduce the data rate from the relatively speedy data streams that we took so much care to create into the FPGA.

Consider the above example of 16-bit sample data arriving at the FPGA at 122.88 Msps. That is two bytes (16 bits) per sample or 245.75 Mbyte/s. At 8 bits per byte, this is a bit rate of nearly 2 Gbit/s (1.96608 Gbit/s, to be precise). Comparing this bit rate to a few common communications interfaces, Gigabit Ethernet (at 1 Gbit/s), Super-speed USB 3.0 (1 lane at 5 Gbit/s) and High-speed USB 2.0 (at 480 Mbit/s) we can see why the communications interfaces are just as important as the sample paths to/from the FPGA on the RF side. Note that even GbE will require some reduction of the input data rate. And here is a thought to ponder: it is not likely that our SBC will be able to keep up with the demodulation and filtering (DSP tasks traditionally performed by the PC in an SDR system) at full Gigabit Ethernet speeds. The characteristics of Ethernet networking will allow the FPGA to split the high-speed input stream into many smaller (decimated) output streams that can be directed at multiple SBCs (or more powerful PCs). The higher the communications speed, the more streams we can accommodate. Remember, there are two very high-speed input streams from each RF Module. You have heard of Multiple Input, Multiple Output, or MIMO **[11]** for transmitters and receivers? Well, this is MIMO for data streams. Most SDRs today provide only one communications interface, often times at a much lower throughput than on the RF side, limiting the width or number of output streams. TangerineSDR will provide both GbE and SuperSpeed USB 3.0 at 5 Gbit/s communications interfaces. Future DE implementations will likely migrate to 10 Gigabit Ethernet (10GE) or 40 Gigabit Ethernet (40GE) or future higher-speed versions of USB. So, while we cannot eliminate the restriction of the output data rate, we can at least minimize it by implementing multiple high-speed communications channels to the outside world.

The communications and GPIO data paths are shown in **Figure 2**. The bold lines are high-speed (LVDS and LVCMOS parallel, typically greater than 10 MHz or high-speed communications) paths, and the thin lines are low-speed paths (GPIO, I2C, SPI and UART serial interfaces, typically up to a few MHz). Although it might not be obvious from the figure, there are two GbE ports on the DE. They are bridged by a three-port GbE switch to the FPGA. This arrangement allows the DE and SBC to communicate with the external network, as well as with each other, over the same Ethernet wire. The protocol is designed to

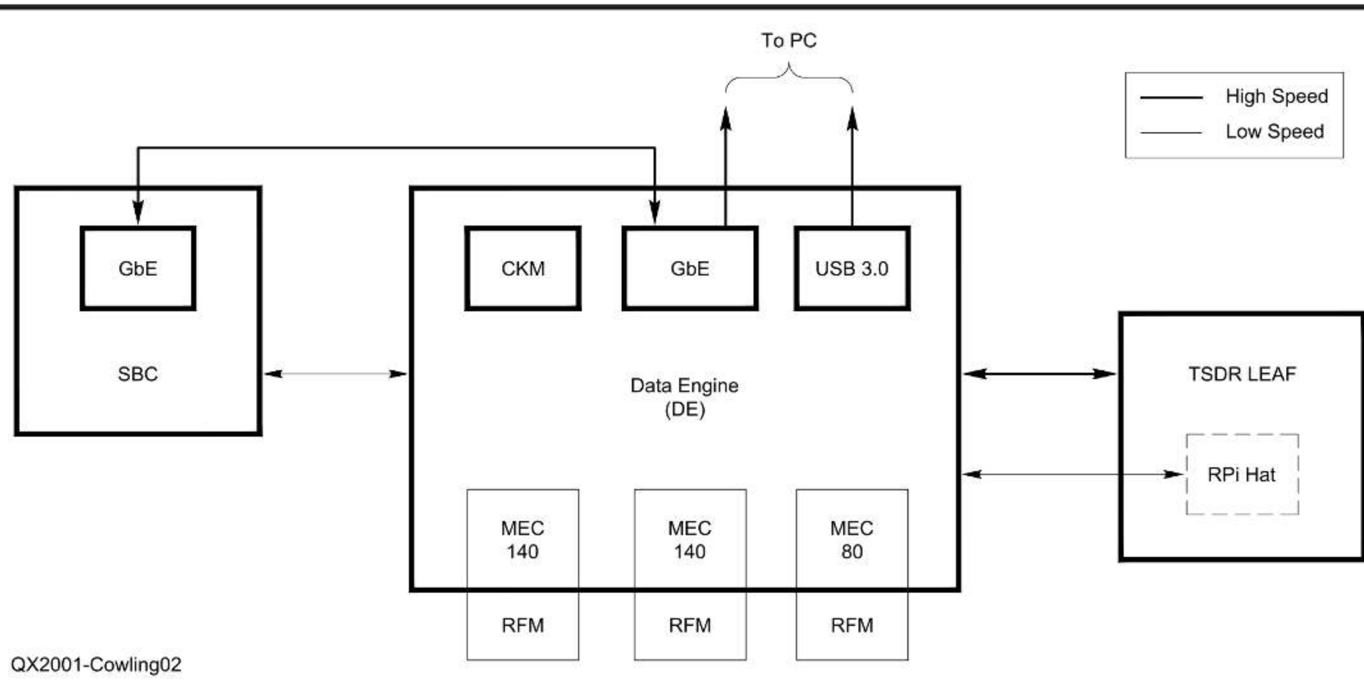


Figure 2 — TangerineSDR communications and GPIO data paths.

protect the DE to SBC path from outside traffic. Authentication is performed by the SBC, simplifying the FPGA system design.

Modularity

Referring again to Figure 2, the GbE and USB 3.0 circuitry are part of the DE, while the Clock Module (CKM) and the three RF Modules (RFMs) are pluggable options. Even though the LEAF Module is shown alongside the DE, it is also a pluggable module.

The LEAF Module is somewhat unique in its design, and part of the ATAU philosophy. LEAF stands for Low-speed Expansion Adapter Fixture, along the lines of the Raspberry Pi Hat (Hardware Attached on Top), the Arduino Shield (arbitrary name appropriation as far as I can tell) and the Beagle Bone Cape (due to its cape-like shape). To fit in with the crowd, we need a catchy acronym, and believe it or not, it fits all of the terms in our description: it has Low-speed I/O connections, it Expands these connections to other modules (ironically, to Shield and Cape modules), Adapts I/O to other connectors (like Click **[12]** and Ultra96 Boards **[13]**) and acts as a Fixture for adding other components. You can even use a standard Raspberry Pi Hat board in place of the LEAF Module (see **Figure 3**.) What distinguishes the LEAF Module from the Hat is the additional high-speed I/O expansion connector on the edge opposite the RPi low-speed expansion connector. It is a high-speed M.2 connector (formerly NGFF, or Next Generation Form Factor) of which you may already be

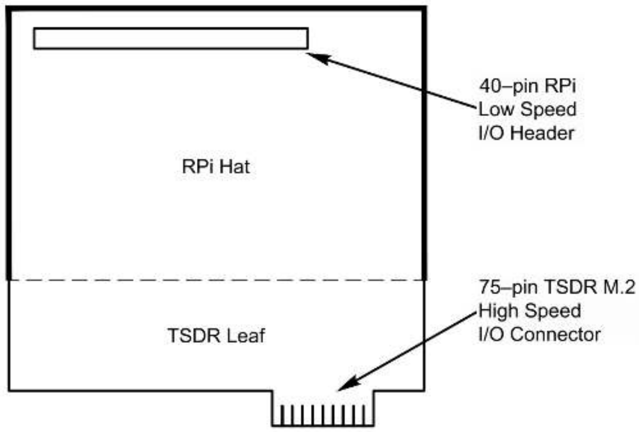


Figure 3 — TangerineSDR LEAF *vs.* RPi HAT I/O expansion boards.

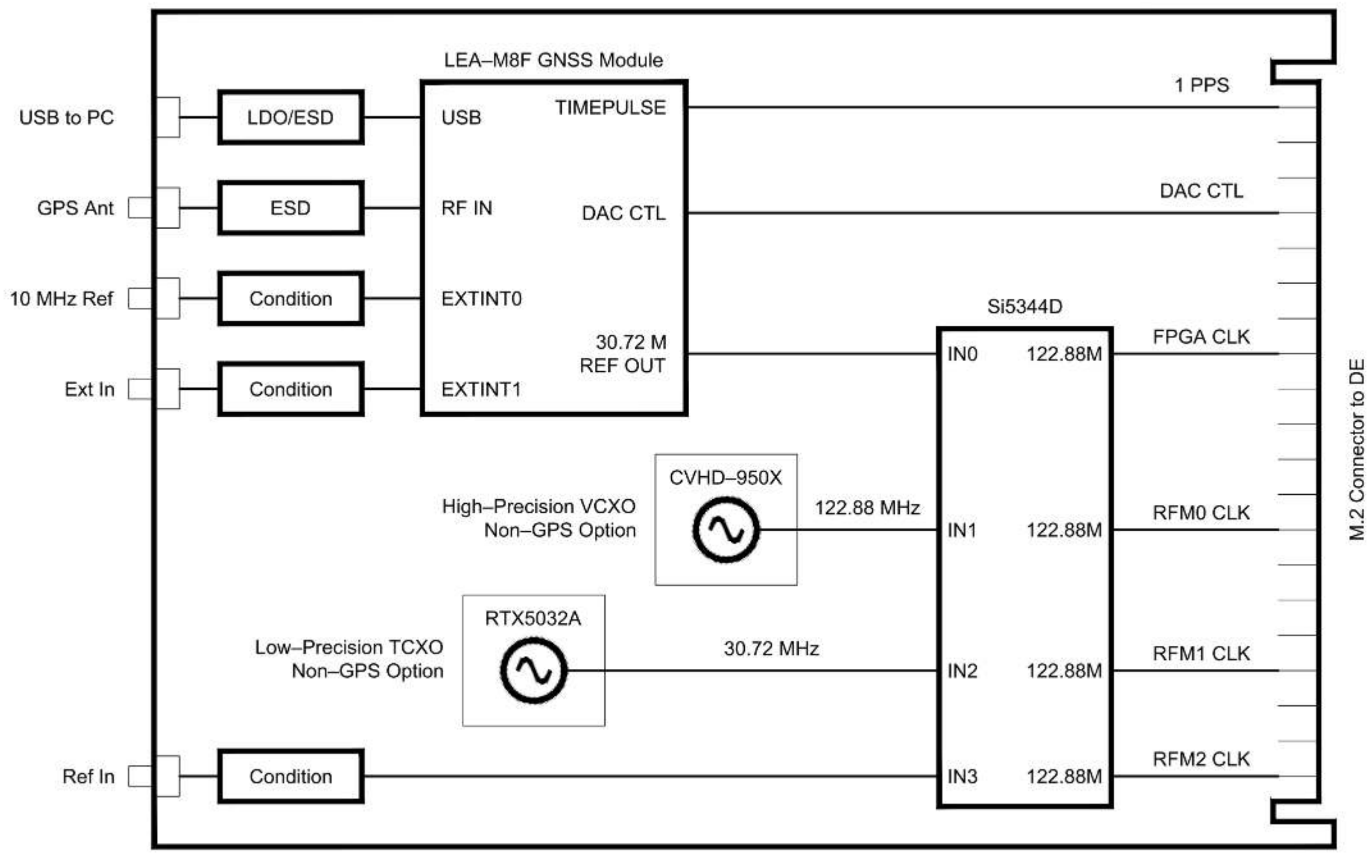


Figure 4 — TangerineSDR GPSDO clock module (CKM).

familiar: it is used for PCIe (PCI Express), NVMe (non-volatile memory express) and SATA (Serial ATA) solid state drives (SSDs). This allows LEAF Module designs to interface to higher speed devices, such as the Xilinx Ultra96 Board, which has a high-speed I/O expansion port. Many LEAF boards are planned, including a version that has Arduino Shield connectors, one with Cape connectors, one with a Click interface, and others.

Clocking

The Clock Module (along with the RF Modules, below) is probably the most performance-defining piece of the TangerineSDR design; see **Figure 4**. It will also have a significant cost impact in its higher-performance versions. In actuality, the lowest performance TangerineSDR will not have a Clock Module at all! The "standard" clock oscillator will be located on the DE board, and will be selected as a reasonable tradeoff between performance and cost. Higher performance (lower phase noise, higher frequency accuracy, oven stabilized, etc.) oscillators are under development even as we work on the DE and RF Module designs. The first TangerineSDR use case is a very demanding one: the PSWS. The PSWS requires very accurate frequency stability as well as a very accurate data arrival time stamp, down to 100ns (or at least as close as we can afford to get to that accuracy). Since this will require a GPS to stabilize, or discipline, the oscillator (hence the name GPS Disciplined Oscillator, or GPSDO), this Clock Module will be expensive. For others, who may not need the performance (and expense) of the GPSDO Clock Module, but desire better performance than the standard on-board DE oscillator, we will build an intermediate-performance Clock Module that will focus more on low phase-noise rather than highly accurate time stamping. Something similar to the Crystek CVHD-950 used on the openHPSDR Mercury receiver is a possibility.

RF Modules

While there can (and likely will) be many types of RF Modules, a preliminary block diagram of the first variant is shown in **Figure 5**. The details may change somewhat as we proceed down the design path, but the basic architecture will remain as shown. For example, the two 14-bit data paths from the ADCs to the 140-pin MEC connector may be differential, instead of the single-ended paths shown. The features of this RF Module are driven by the requirements of the PSWS. Note the on-board noise source

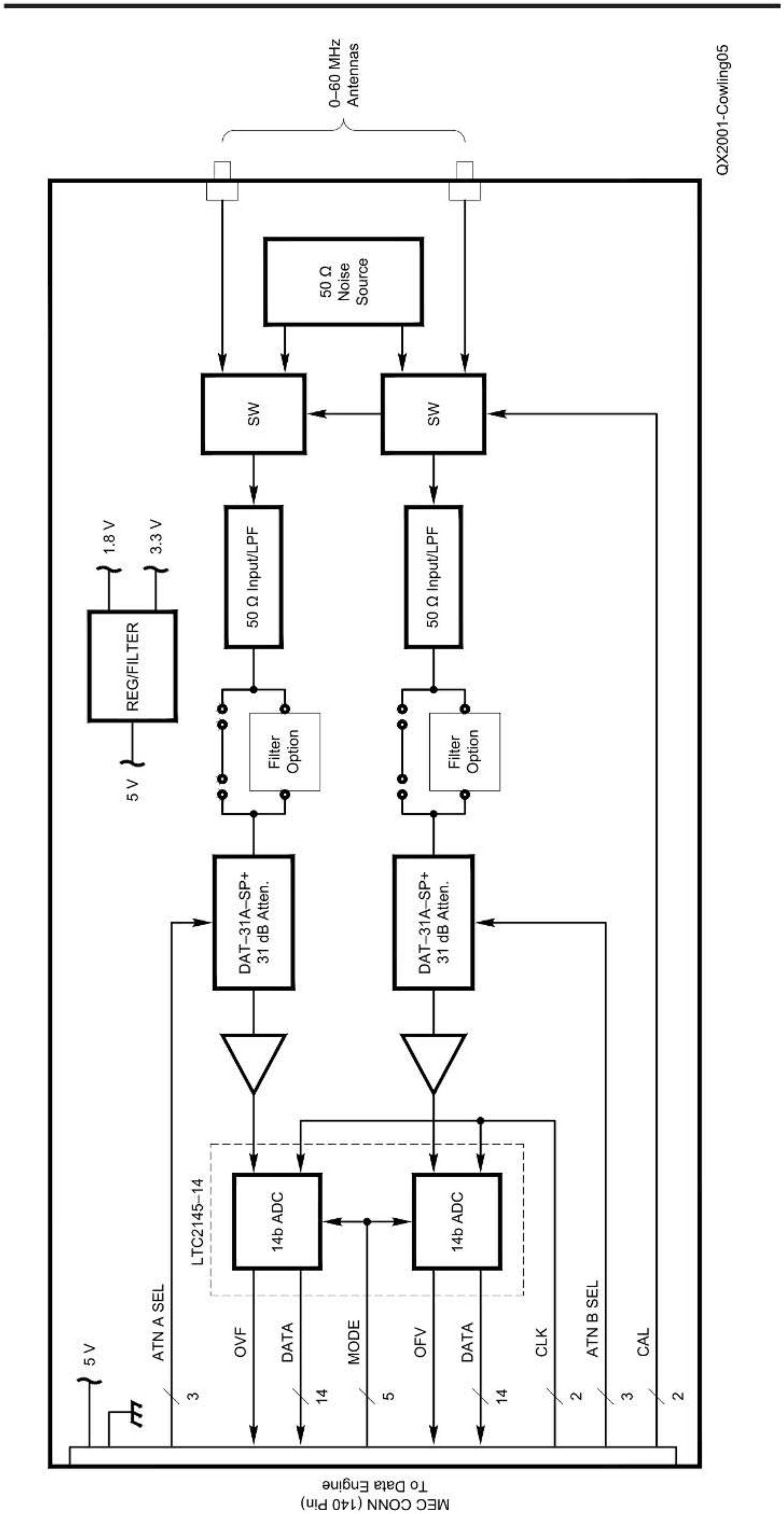


Figure 5 — TangerineSDR Dual-Channel 0 – 60 MHz RF Module (RFM).

that can be switched onto the receive path for calibration of each receive channel. Also of note are the optional in-line filters, one for each channel. These filters are normally bypassed by jumpers, but you can remove the jumpers and install filter modules to adapt the front-end response to whatever local situation might arise. Live near a local high-power MW AM station? Add a notch filter to take its signal down to a manageable (i.e., non-ADC-saturating) level. Live in between two or more high-power MW AM stations? Add a high-pass filter to eliminate overload from anywhere within the whole MW broadcast band. We will provide power to the filter mounting headers, so you can even put active circuitry on the filter boards! The 31-dB step attenuator may be replaced by a three-step 0-10-20 dB passive attenuator in order to keep as few active components as possible ahead of the ADC. The clock for the ADCs comes from the DE, and its characteristics will depend on the type of Clock Module installed. This first RF Module will be designed to clock at the ever-popular frequency of 122.88 MHz. Just in case you are still wondering why we use such an odd frequency, it happens to be an integer multiple of 48 ksps (2,560 times 48,000 to be exact), which makes our common output sample rates of 48 ksps, 96 ksps, 192 ksps, 384 ksps and 768 ksps easier to generate.

Use Cases

Hopefully I have given you a reasonable introduction to the hardware in this installment. I fully intend to fill in the gaps in this description (there are many) in much more detail as the project unfolds. Since the project's inception, many individuals have come forward (some at my direct request, and others by word of mouth) with ideas for how they could put a TangerineSDR to use. I call these our use cases, and the idea is to make sure to include as many features as we can so as not to overlook any important ones. Some use cases are simple (listen to SSB on my favorite 40 m frequency), while others are more difficult (PSWS). Following is a partial list of TangerineSDR use cases. If you have others, please contact me. ATAU!

- PSWS (of course)
- Satellite Ground Station (requires new RF Modules)
- High Performance HF transceiver
- WSPRnet/RBN on multiple bands simultaneously
- HF noise sniffing/calibrated receiver
- Remotely controlled stations
- Radio Astronomy (Project Jove **[14]**, SARA **[15]** pulsar detection)
- Academic Learning.

What's Next?

Remember that the TangerineSDR project is open source, and there are many use cases for the hardware. There are opportunities for FPGA programmers, applications software developers, GUI designers, and yes, even hardware designers for the next generation. Each DE board variant will have an on-board FPGA and Verilog code to match. Each new RFM will need an RF expert to oversee its development. I will be covering the evolution of the hardware, firmware and software in subsequent columns. Hardware will start to become available this spring with the DE and HF RF Module, followed closely by the Clock Module. I expect a long and evolutionary (hopefully revolutionary, too!) lifetime for TangerineSDR. OpenHPSDR started in 2006, nearly 14 years ago now. It will be interesting to see where TangerineSDR development is in the year 2034!

As always, please drop me an email if you have any suggestions for topics you would like to see covered in future Hands-On-SDR columns or even just to let me know whether or not you found this discussion useful.

Notes

[1] SDRPlay: **sdrplay.com**.
[2] RTL-DVB Dongle: **rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles**.
[3] Red Pitaya: **redpitaya.com**.
[4] KiwiSDR: **kiwisdr.com**.
[5] Kerberos SDR: **othernet.is/products/kerberossdr-4x-coherent-rtl-sdr**.
[6] LimeSDR: **limesdr.com**.
[7] Tucson Amateur Packet Radio, or TAPR is the creator of the TNC1, TNC2 and numerous other advances in the art of digital radio. SDR is the logical continuation of TAPR's effort to advance the state of the art in digital radio. Visit **tapr.org** for more information.
[8] Ham Radio Science Citizen Investigation, or HamSCI, is a group of hams and scientists (many are both) dedicated to advancing scientific research through amateur radio activities. Visit **hamsci.org** for more information.
[9] A Google search turned up nothing for "ATAU" or "All things to All Users," so I will stake the claim of ownership to the abbreviation "ATAU."
[10] Universal Software Radio Peripheral: **ni.com/en-us/shop/select/usrp-software-defined-radio-device**.
[11] Multiple Input Multiple Output (MIMO) is a method for multiplying the capacity of a radio link using multiple transmission and receiving antennas: **en.wikipedia.org/wiki/MIMO**.
[12] Click is a small I/O module that uses mikroBUS™: **mikroe.com/mikrobus**.
[13] Ultra96 MPSoC FPGA development board: **96boards.org/product/ultra96**.
[14] Project Jove studies radio emissions from the planet Jupiter, the Sun and our galaxy: **radiojove.gsfc.nasa.gov**.
[15] Society of Amateur Radio Astronomers (SARA): **radio-astronomy.org**.

Mike Lewis, K7MDL

268 Alta Via Dr., Camano Island, WA 98282; **K7MDL@hotmail.com**

Multi-Band VHF+ Local Oscillator and Remote Antenna Switch

Provide support for, and extend the useful life of, multiple transverters.

Like many VHF enthusiasts, I have a stack of older transverters. They worked fine for years using their internal crystal local oscillator (LO) but drift too much when used with digital modes. One solution is to replace the LO in each with a synthesizer, possibly one that can be locked to a 10 MHz reference. There is a fair amount of expense involved in purchasing one for each transverter. Effort to squeeze one into the chassis is not always possible and may need attenuation or amplification added for the correct LO level. An alternative is to switch a programmable LO among the transverters. That is the approach I took for this project.

With the availability of products such as the digiLO 6 GHz (see **q5signal.com**) programmable PLL, a single synthesizer unit can handle most LO needs if it is connected and switched among the stack of transverters. Only one 10 MHz reference connection need be made. Recently high-performance parts that work up to 6 GHz have become available at relatively low cost mounted on small PC boards or modules. These pair well with the digiLO 6 GHz capability. A means to control these parts is needed to make them useful when sharing the output among among several transverters.

Since each transverter has unique needs, function modules were added such as an amplifier and a programmable attenuator. A configurable system was needed to make it reasonably general purpose and able to work with transverters that I or others might use today and in the future. It should control remote antenna switches and/or enable a transverter, the PLL frequency, display GPS data and measure the LO chain output power level. Future features should be easy to add such as more band decoding input methods, GPSDO functionality, or maybe a pair 915 MHz transceivers to replace the remote switch control cable. The last is interesting to me for increased lightning protection by reducing the number of wires entering the station. To satisfy Covenants, Conditions & Restrictions (CC&R) rules, and for easier remote operation, controlling remote antenna switches let me use a common coax feed line to stealth antennas for 50 MHz through 1296 MHz hidden in trees, attics or under decks with a minimum of cabling. A builder should not need to be a programmer to build this and the code should be relatively easy to adapt to new versions of modules since the lifespan of low-cost consumer modules is finite.

The Project Described

I recently retired, so I found time to dust off my 20 year plus old software skills for a fun creative project that is now central to my remote-controlled radio operations. I built a centralized LO using a capable CPU platform that provides a single stable low noise PLL generated LO source at the correct frequency and power level automatically routed to as many as 6 transverters. The transverters are selected either automatically via a radio's digital band info or manually via the front panel controls. Band selection is translated to external control for switching transverters and/or remote antennas. An inexpensive CPU development board combined with a digiLO 6 GHz programmable PLL/VCO and several off-the-shelf functional modules make this work. Other programmable synthesizers could be used with reasonably easy source code changes. Some of the modules may come from your junk box or be found as surplus parts, or omitted if not required. I wired the first version on perf board and used junk box parts. Those old parts used parallel control interfaces so used many CPU board I/O ports.

Needing to build more units for microwave tripod use in the field and at the vacation home station, I created a PC board to make assembly easier. During this time, I began switching to serially controlled solid-state imported modules to free up I/O port pins for more features.

This article is based on the V1.0 PCB and uses a mix of parallel and serial parts. One could eliminate the modules and design a PCB with the equivalent piece-parts but that is a significant design and build effort given the high frequencies involved and minimum quantities for some parts. The module approach lets you mix and match parts as needed and upgrade parts more easily at minimum cost. You could build

(A)

(B)

Figure 1 — Front (A) and back (B) views of a compact LO box.

this to function only as a GPS display, work only as a programmable band decoder, or as a programmable weak signal generator with adjustable output and a display for the fun of it. **Figure 1(A)** shows the front view, and **Figure 1(B)** shows the back view of a compact version of the LO box.

What is in the Box

The modules used in this project include a CPU dev board, LCD display, programmable PLL, programmable attenuator, a mechanical or solid-state multi position coaxial switch (SP6T), a wideband 6 GHz amplifier, and a 6 GHz capable log power detector. The LCD display and rotary shaft encoder with pushbutton allow manual control, navigate the **config** menus, and optionally view GPS data on 2 status screens. Many will want or already have a GPSDO for a 10 MHz reference which has a serial output that will provide the GPS data. Only TTL level serial GPS connections are supported in this version PCB. Configuration and operational state are saved in EEPROM. **Figure 2** shows inside of the compact version with its populated PCB. The modules are stacked for a small footprint. In **Figure 3** the logarithmic power detector is at the top center, the wideband amplifier is next to it. These are a stack of 3 modules, top to bottom: the PLL, the attenuator, and the RF switch. **Figure 4** shows the block diagram.

How it Works

The system provides 7 configurable band 'slots' containing all the needed parameters for the function modules to be used with a specific transverter. I chose 7 to allow for 1 direct connect and 6 switched ports. I chose 6 ports because SP6T RF switches are commonly available, and I had a box full of them, and wanted to cover 6 bands 50 MHz through 1296 MHz. The direct connect or 'switch position 0' could be used to turn off a mechanical coax switch reducing power usage depending on your switch of choice.

Any band slot can be configured for any combination of switch position, attenuator level, PLL frequency, and other controls such as band decoder and antenna relay control patterns. Other settings are band-independent such as Hi/Lo polarity on PTT input and output. The output of the PLL-ATT-AMP chain is normally connected to the RF switch input port via a short coax jumper. If you use a solid-state RF switch module for LO output, it will likely be the most frequency limited component in this build example. Remove the coax jumper and the switch is bypassed when minimal loss is required. I created the 7th band slot to store the parameters used in this configuration for easy use. You can use mechanical switches, which are often rated to 18 GHz, and they can be any number of ports limited only by the 6 control wires currently provided to drive them. The LO RF switch needs to be a TTL or a low side-drive (ground to select) type. I built one with a surplus TTL controlled SP6T mechanical relay and another compact build with a solid-state SP6T RF switch module rated to 3 GHz. Maybe you need only a SP3T or SP4T switch.

Any frequency from the digiLO PLL preprogrammed list can be assigned via the configuration menus. The current software includes a subset of the 255 possible choices in the digiLO PLL unit. I included weak-signal and popular LO frequencies for bands 50 MHz to 24 GHz. Bands at 10 GHz and higher require an external multiplier since the digiLO is limited to 6 GHz. The remaining choices will be added to the software later or can be added by many builders in the code themselves. This is time consuming so I deferred it for now.

The solid-state serial-controlled programmable 0 – 31 dB step attenuator that I used can do 0.25 dB steps. I felt 1 dB steps were all that were needed. It is possible to chain-multiple 31 dB units by using extra chip select pins with a slight change to the software and a spare I/O pin. Parallel controlled attenuators can also be used with a minor software change but are limited to 4 or 5 control pins, limited only by available I/O pins.

A 20 dB gain 6 GHz rated wideband amplifier is connected in line with the attenuator to boost the typically +2 dBm digiLO output to around +22 dBm when needed. Combined with the 31 dB attenuator you achieve an output range of +22 dBm to -9 dBm (less the module frequency dependent insertion losses), which should

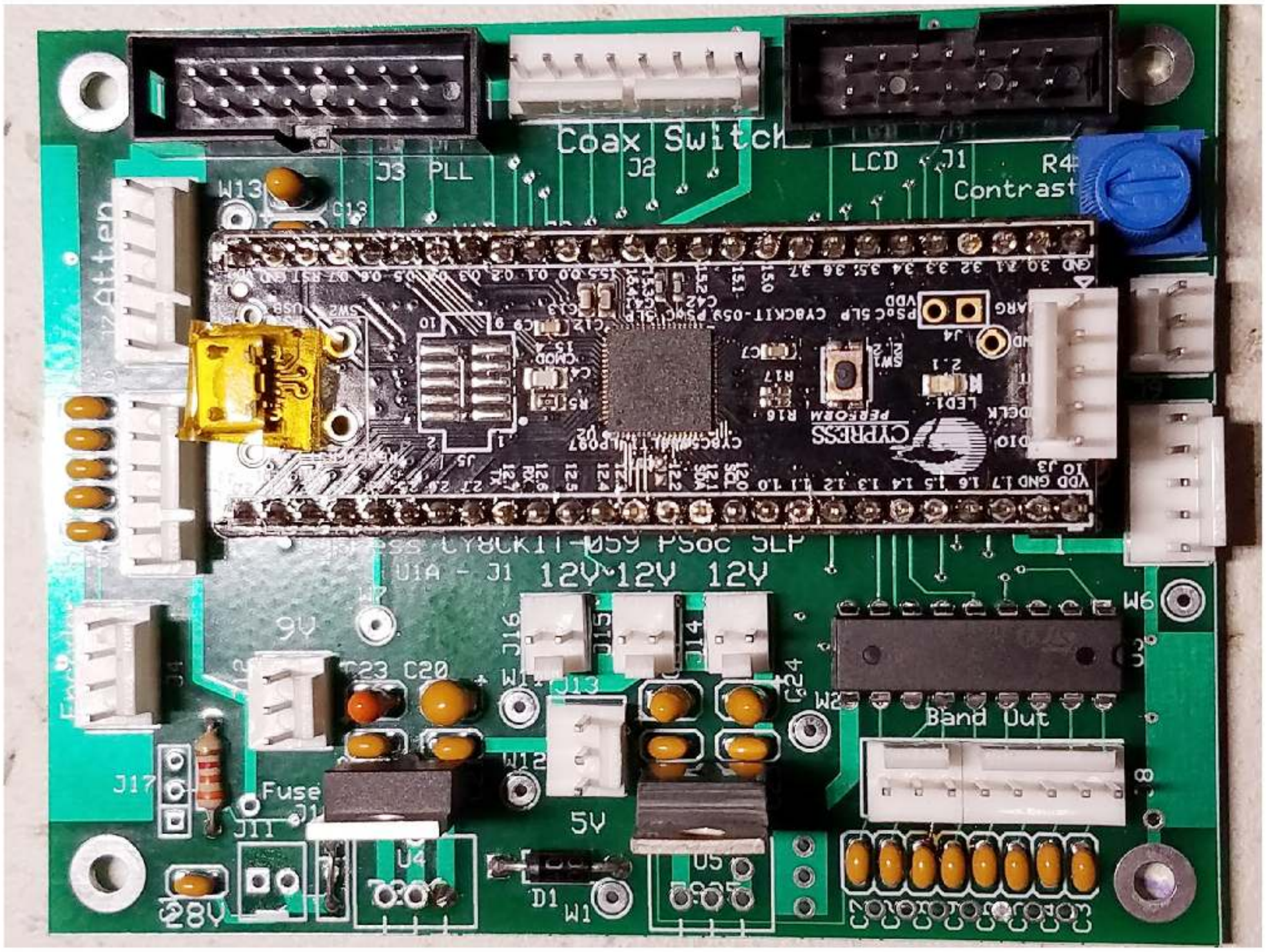


Figure 2 — Inside look at the compact version with its populated PCB. The modules are stacked for a small footprint.


Figure 3 — The log power detector is at the top center, the wideband amplifier is at the top of the lower position. Under that is a stack of top to bottom: the PLL, the attenuator, and the RF switch.

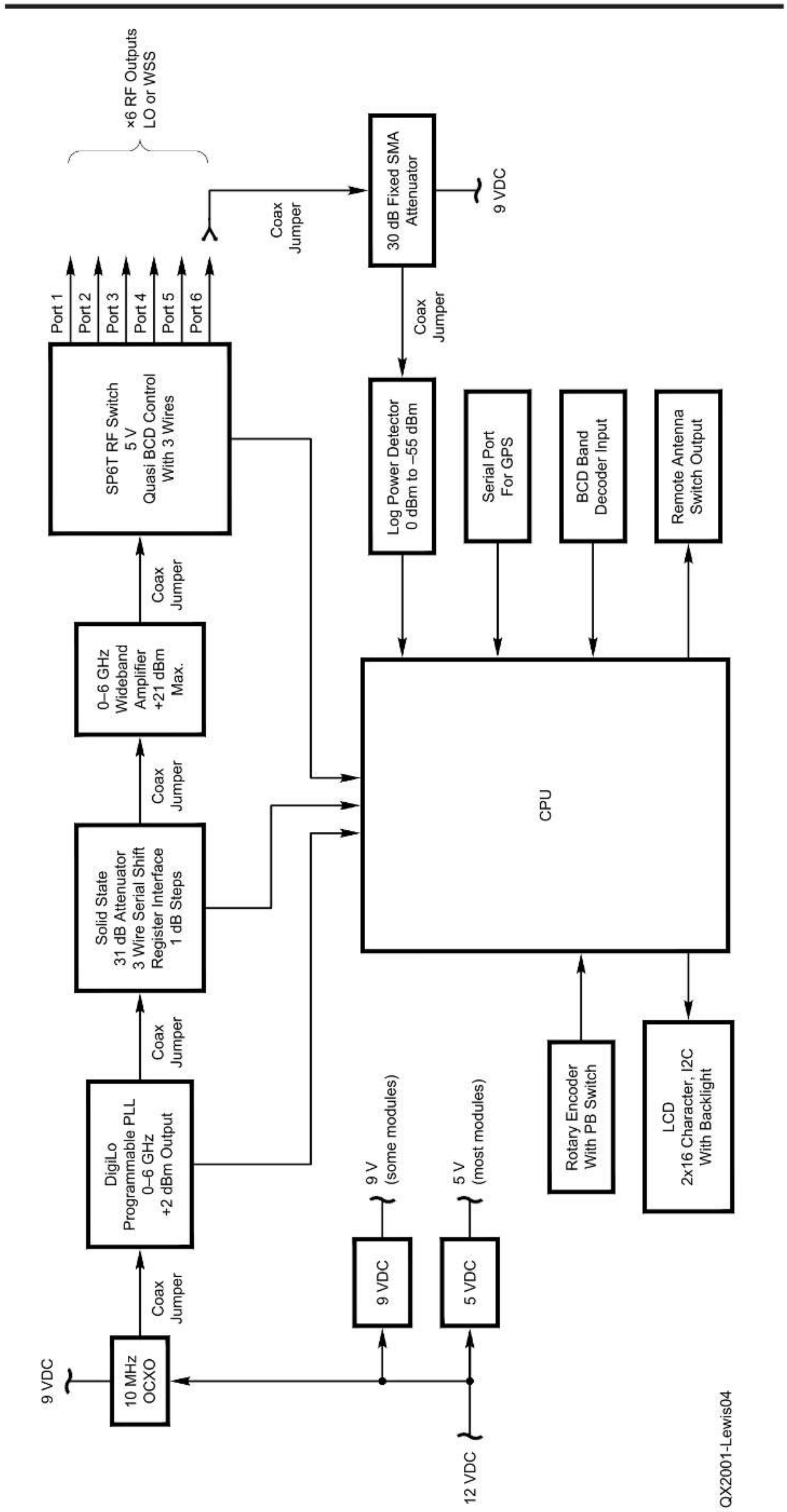


Figure 4 — Block diagram.

suit most use cases. You may want to use a fixed attenuator at the PLL output to lower the range if your transverter mixers use less than +17 dBm drive to help prevent accidental mixer LO overdrive accidents. You should account for some insertion loss from modules in the LO path at the highest frequencies you use.

The PCB was laid out for a parallel LCD connection, but I now only use a few wires having switched to an I2C serial connection via a 'backpack' module mounted on the LCD. There are now a few free analog capable I/O ports available there. Details on how I modified the backpack for backlight dimming control are in the online Appendices at **www.arrl.org/QEXfiles**. An unmodified backpack will have full time backlight. My LCD of choice has an onboard series resistor for the back-light LED at 5 V. Ensure you have an appropriate resistor on or off your LCD board. **Figure 5** shows the LCD backpack. You can see the backlight enable jumper on the left side, modified for PWM dimming connection. Note the two trace cuts on the left side. See **Appendix 7** in the **QEXfiles** materials for a better view of the trace cuts I made.

Please refer to the on-line materials with respect to the following details. An AD8318 based 6 GHz+ rated log-power detector module is connected to a spare LO switch port, if available. This module was added after the PCB was created, so I installed a 2-pin Molex power connector at the spare 9 V regulator pads and soldered the detector output wires to a spare I/O pin (CPU port P3 pin 2), which can be found on the LCD header connector. The input range is about 0 dBm to -55 dBm, so a 30 dB fixed SMA attenuator is connected in front of the detector to shift LO output from +22 to -9 dBm down to a range of -8 dBm to -39 dBm, right in the sweet spot for this detector. It is accurate enough in this range that adding frequency-based calibration data is a future task. There are constants defined in the software for offset and the attenuator size, which are easy to change if needed.

Configuration settings and operational status are stored in EEPROM. Operating state is stored immediately and on band changes, menu items are stored only to EEPROM when you use the **Save Config** menu. There is a default configuration that hopefully represents most of what you need to offer a good starting point to adjust to your needs. There is also a factory reset menu item that takes effect at the next power up. No computer is required for configuration. A Windows computer with a USB port is required for initial programming of a new CPU module during build time. The LCD display and rotary encoder are used thereafter.

If GPS serial port data is connected (TTL level) latitude, longitude, and a calculated 8-digit Maidenhead Locator System grid square value are displayed on the LCD screen. The grid square is stored in EEPROM so it can represent the last known good location in case GPS signal is lost. Also the number of satellites in view and an SNR figure are displayed. I used the QRP-Labs QLG-1 GPS module kit. It has TTL level, is very sensitive so it works decently indoors, provides a 1 pulse per second signal (not used so far), and is inexpensive. It does require your own enclosure, and you can optionally use an external antenna. If you are building this project primarily for outdoor use and using a plastic enclosure, you could mount the GPS module with the internal patch antenna soldered on in the top portion of your case.

In a build using an internally mounted 12 V dc oven-controlled crystal oscillator (OCXO), I used separate power switches for main unit power and OCXO heater power. This enables some power savings while keeping the OCXO warmed up in the field. If the OCXO is not required, you can save power by leaving it turned off letting the internal temperature compensated crystal oscillator (TCXO) do the job. This is enabled by your choice of power wiring inside your box. With my 12 V dc powered OCXO I split the oscillator section off from the heater and ran it from a dedicated on board 9 V regulator for less heat generation, lower power use, and a more stable supply voltage.

A PWM output is provided for an adjustable backlight level control for the LCD display. LCD backlight devices draw from 20 mA to as much as 250 mA. It is routed through the ULN2803A coax switch driver chip. If you leave this chip out for a TTL driven coax switch, then solder in a suitable transistor. I used a 2N7000. A common PN2222 will also work for most cases. Just make sure it can handle your maximum backlight current.

Clean 12 V dc, 5 V dc and 9 V dc power to subsystems make for a clean LO signal. It needs to work on low 12 V battery levels so I used a low dropout voltage 9 V regulator. I try to limit use of raw 12 V to non-voltage sensitive parts and use the 9 V and 5 V supplies for the critical parts. Provision is made in some parts of the PCB to allow use of 12 or 24 - 28 V dc switches common in the surplus market. Solid state switches use 5 V dc. Switches need to be low-side drive or TTL. When a 28 V source is required you could use a small 12 V dc-dc converter in series with the main 12 V supply.

The CPU used is an inexpensive and compact Cypress Semiconductors PSoC 5LP based development module part number CY8CKIT-059 (US$10). PSoC stands for Programmable System on Chip and has lots of I/O pins with programmable analog and digital hardware function blocks on chip. These include logic gates, comparators, op amps, EEPROM, SRAM, Flash Memory, voltage references, clocks, dividers, PWMs, UARTs, filters, de-bouncers, de-glitchers, timers, counters, status and control registers, and several types of ADC/DACs. The software and hardware signals are all routable to most any I/O pin. The I/O pins can be configured with internal 5 kΩ resistors that pull up, pull down, both, or none, with digital or analog high impedance inputs.

The free Integrated Development Environment (IDE) schematic drawing tool is used to configure all of these components as well as code creation and debugging. This allows significant hardware design changes usually not requiring PCB changes. The IDE uses the C language. Code samples are provided in the IDE for each component for fast project startup. Debugging and programming from the IDE is via a USB connection to the included break-off programming board connected to the CPU module. The IDE generates standardized APIs for on-chip hardware functions configured and is used to initially program a CPU. No coding skills are required to program a new CPU module so some procedural steps and program installation are performed, but no writing or editing of software is typically required. See the **QEXfiles** on-line Appendices for instructions.

The PCB was designed to ease assembly time and provide for some external I/O protection, voltage scaling to CPU levels, voltage regulators, and relay drivers. No external logic parts are required, they are all within the CPU so far.

On my larger-size enclosure build, I used a one piece 4-position power-pole module on the back panel to distribute 12 V dc to other gear while sitting on a tripod. I used a 2.1 mm coaxial power plug due to space constraints on the compact enclosure.

Currently only low side drivers are used. The driver chips can be bypassed for TTL switches. High-side drive switches will need level translation in this PCB version. I use small 4-pin opto-coupler based solid state relays (Toshiba TLP222AF) mounted on a perf board near the switches for this purpose. I anticipate using a serial controlled configurable Hi/Lo driver part in the future.

The block diagram in Figure 4 represents my last build. The function blocks are self-contained hardware modules and the interconnection order could be different or left out when not required. There is some flexibility in the hardware and software for your choice of some function modules. A fixed attenuator might replace the programmable one if your needs are simple or you place the attenuation needed in each external transverter connection.

The serial LCD driver used is community code so requires you follow the installation instructions included in the code's PDF data sheet to add it to your development environment properly. I have summarized those steps in the online Appendices.

Your choice of enclosure may need to fit an OCXO, mechanical switch, larger connectors, and 28 V dc-dc converter, so plan accordingly. The more mechanical parts used, the higher the current drain and heat produced so additional heat sinking or even a small fan may be required. For an all solid-state module version that I built, I used small bolt-on heat sinks on the 5 V and 9 V regulator tabs. For the mechanical

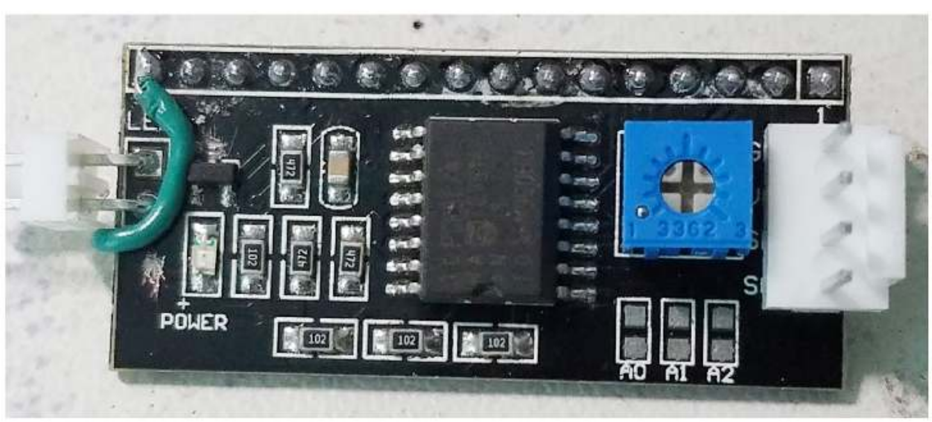


Figure 5 — LCD backpack, left side shows the backlight enable jumper modified for PWM dimming connection. Note the two trace cuts on the left side.

parts version I mounted the regulators on the steel chassis bottom and used a quiet fan. Many of the solid-state modules had PCB mounting holes that permitted stacking them easily with #4-40 or M2.5/M3 size standoffs.

The driver chips may be omitted and jumpered for TTL devices. Voltages will be jumpered for some modules used, like relay drivers. GPS and wide band amp are 5 V. PLL and power detector are 9 V. Depending on your choice of hardware you may need an additional voltage. Spare pads exist for connecting to various voltage supplies and spare CPU I/O pin access.

For the band decoder feature set, I tested with two different vintage Elecraft K3 radios, which supply a 4-line BCD output at TTL levels. Logic 1 is floated to 5 V and switches to ground for logic 0. The LO box band decode input uses four wires and can be programmed for any pattern, BCD or 1 to 4-lines parallel — any value 0x0 to 0xF. You just select a pattern in the configuration menu, same for the remote antenna control output (via U3 driver). The PTT signal on my two K3 examples had 8.8 V dc on one radio and a weak 5 V on another, both grounding on TX. This required different values of R2 and R3 or even leaving resistors out. R2 and D2, a 4.7 V dc Zener diode, protects the input pin against high input voltages. The A/D converter voltage reference is set to 0.250 V but is easily changed to a few other values in the component editor if needed.

Operation

The optical rotary encoder and push switch is used to control the LO box. A short button push (~50 ms to 800 ms) is **Select**, a long button push (800 ms to 1600 ms) is **Back**. An extra-long push (>1600 ms) is used to change modes. **Back** and **Mode Change** will abandon whatever operation was in progress.

There are three operation modes. **Band Status**, **Configuration** and **GPS**. The three modes are cycled by holding the push button for more than 1.6 s (extra-long push), or until the screen changes to the next mode in a cycle.

Band Status Mode

This mode has one screen. It displays most of what you need to know about the operational state of the box. The first three characters in the upper left corner represent status, the rest of the screen presents a band name label, active frequency, attenuation level, and LO switch position. The Remote Antenna Switch output and Band Decode input patterns are activated per band in the configuration menus. PTT state and GPS speed are global settings.

Character Position #1 shows **Auto/Manual** mode and is represented by an **A** or **M** in the first character of the upper left corner. In **Auto** mode short or long button pushes are ignored. In **Manual** mode short button pushes cycle through all 7 bands.

Character position #2 is a custom symbol for "GPS Data is Valid" that looks like a satellite and ground station dish. It is blank when there is no valid GPS data.

Character #3 is **R** or **T** representing Rx/Tx status, which follows **PTT** input state. It is the state of Tx or Rx as interpreted by the **Input Polarity** (**Hi/Lo**) setting.

Upper row of text is the current selected band label. This is a free-form label you create during configuration. Lower row text contains the attenuation level Axx where xx is 0-31, the frequency selected on the digiLO (frequency text displayed is hard coded in software tables), and in the lower right corner is the LO switch position Sx where x is 0-6.

GPS Status Mode

This mode has two screens. A short button push cycles between them. The first screen shows GPS reported time, 8-digit Maidenhead Locator System grid square calculated from the GPS reported latitude and longitude also shown. There is no internal real-time clock so there is no time displayed when the GPS signal is lost.

The second screen shows grid again plus GPS status such as Satellites in View and S/N ratio. When GPS Data is invalid (bad signal, no GPS connected) the system times out after a few seconds and changes the display to say there is no signal, and posts a time since last good message. Note this counter is arbitrarily limited to 255 seconds and will simply wrap around to back to 0.

Configuration Mode

This mode is for all your customization. It presents a multi-level menu system that uses the rotary encoder for right/left navigation. Here, for each Band Slot, you can set the Band Name label, set values for modules, enable/disable the GPS mode screen and set the bit rate and more. See the Menu Table in the on-line **Appendix 3** for an explanation of each menu item usage.

The submenus will either offer a scrollable list of parameters or a list of alphanumeric characters to build a Label. On choosing to change an item, the current value is typically displayed. Rotate the encoder knob initially right to show the list, then rotate right or left to display a value and do a short push to select it. For label type items, repeat for any following characters up to the max characters allowed where **Accept** will show. Push to accept and the menu returns one level up with your new setting active, but not yet stored in EEPROM.

The top-level menu contains **Version**, **Auto/Manual** mode toggle, **Callsign**, and **Configuration**. **Version** is a hard-coded software version number that contains the date. Callsign can be edited by choosing **Callsign Edit** in the **Config** submenu. It exists to personalize your project. The submenu choices are shown in **Table 1**.

Some configuration menus like **Attenuation**, **LCD Backlight**, and **PLL frequency** update in real time but are not stored real time as you scroll through the setting values. This is convenient for testing. You could watch the effect of LO drive level on an IF radio's pan-adapter, watch the power output level change as you spin the knob and change frequencies or attenuator settings. Do a long or extra-long push to abandon any potential change or simply toggle the power switch. With an antenna connected to the LO switch WSS frequencies can be programmed into all 7 bands slots and all slots assigned to the

Table 1.
The Configuration Menu offers these submenus choices.

Choice	*Description of parameter*
Band Name	any alphanumeric character in the list
Coax Switch Port	1-6
Frequency	select from list
Attenuation	0 to 31 dB in 1 dB steps, select from list
Band In	any value 0x0 to 0xF, select from list
Band Out	any value 0x00 to 0x3F, select from list
PTT In	Hi or Lo on Tx, select from list
PTT Out	Hi or Lo on Tx, select from list
GPS	4800, 9600, Disable, select from list
Backlight Dimming	0 to 100% bar graph
Save to EEPROM	—
Factory Reset	—

same LO switch port 1 for easy WSS band selection via the push knob when viewing the **Band Status** operational menu screen.

Power is displayed (**Figure 6**) when the detected power is greater than -20 dBm or the **Band Name** is set to **POWER**. The coax switch must be set to route signal to the detector first. Certain settings are saved to EEPROM immediately or on band change as a group. These include **Band in Use**, **Auto/Manual**, **Mode Screen**, **PTT** in and out **Hi/Lo** mode, **LCD backlight level**, **GPS Setup**, and **GPS Grid Square**. All other configuration-related changes are saved only to EEPROM when you select **Save Config Data** in the **Configuration** menu. Doing this will ensure all your changes are saved during power off. The **Config** menu also has a "**Factory Reset**" selection that will mark the EEPROM saved data as invalid and upon the NEXT power up, will initialize the EEPROM with default settings. If you choose factory reset and later decide against that, do not power cycle the box, do a regular Save Config and it will mark the EEPROM as valid again and store the values you are currently using.

The software operation is based on a simple flexible database concept. The 7 records or band slots can be assigned to any purpose and have any setting and name. An extreme example is you could set all 7 band slot settings to identical values. This would of course be confusing and of little value but illustrates that there are few limits. This means you can mix and match assigned settings in creative ways to suit your needs. Some examples are provided in the full build document online, which may or may not make sense for you, but might for another user or a for a special test on your workbench.

Here is one configuration described for example.

Band Slot Label = 432, **Band Decode Input** = 0x4, **LO Freq** = 404 MHz (for 28 MHz IF), **Atten** = 16 dB, **Coax Switch** position = 4.

This will provide a proper LO for a 432 MHz transverter with a 28 MHz IF on LO switch port 4 at likely around +6 dBm at 404 MHz (+2 dBm digiLO output +20 dB gain amp -16 dB attenuation for a net +6 dBm final output). At higher microwave frequencies you are likely to see increased component losses based on each module's capabilities and interconnect cabling. In **Auto** mode this band slot is selected when the radio is changed to 432 MHz based on a matching band decoder input pattern, or the first matching band if dupe patterns exist for this input from the radio or external switching.

Figure 6 — Attenuation Config Menu shows 432 MHz band with 31 dB attenuation setting and -8 dBm measured output.

Conclusion

If you build two or more of these, you might do what I did and dedicate one as low power in a small package connected to an external 10 MHz precision reference and use it on a tripod for microwave bands. You might omit the wide-band amplifier and LO switch. Connect a small antenna and you have a weak signal source or a test RF generator. The 2nd unit built could supply the LO drive for a stack of transverters in your shack or for rover radio. Maybe you do not require more than 4 ports or more than 2 dBm LO drive level, and choose to skip the amplifier module and choose a 4-port RF switch for the LO output. With such flexibility comes ability to choose, but it is also more difficult to explain.

As most any combo is possible, it is up to you to decide what makes sense for your collection of hardware. The software will not likely be a limitation. Future developments could support directly programming a PLL for any frequency desired within its limits, and likely a range of different PLL chips. I envision a sweep function added to help align filters.

The Appendices, PCB layout files and images, schematics, software package, pictures, and more are located online at the **www.arrl.org/QEXfiles** web page. Feel free to give me your feedback, and I might possibly assist you with questions and improvements.

Mike Lewis, K7MDL, has been active since high school in 1976 originally as WB7OSE. Since about 2000 he has focused mostly on VHF and above, working satellites, EME, meteor scatter, and terrestrial weak signals embracing digital modes, equipment building and contest roving. His career included engineering roles in nuclear submarine plant operations, hardware and software design, international telecom, automation for desktop and server operating system mass deployment, and marine electronics. This has naturally spilled out into fixed and rover station packaging and automation leveraging computing.

Mike is now semi-retired and travels between homes in Camano Island, WA and Bradenton, FL in a motor home. Current projects have focused on home station remote operation, working any and all bands despite CC&R antenna challenges, and starting to catch up on a backlog of work on the microwave bands.

James A. Koehler, VE5FP
2258 June Rd., Courtenay, BC V9J 1X9, Canada; **jim.koehler@usask.ca**

A Simple Inexpensive Accurate Vector Impedance Meter

An inexpensive 32-bit single-board processor sub-system is the basis for this wide frequency range vector impedance analyzer.

Impedance meters are valuable tools for designing and constructing antennas, implementing RF designs and measuring RF component parameters. Professional vector impedance meters have been very costly but the amateur fraternity has come up with a number of excellent and more affordable designs, see **"Notes" on page 23**

I was inspired by DG5MK's **"Notes" on page 23** use of the MCP3911 dual-channel A/D converter and by EU1KY's **[4]** use of the Si5351A single chip dual-output signal source, to design my own version, which used these components along with a rather old semi-obsolete 32-bit ARM processor. It worked very well **[5]** but the processor was becoming difficult to obtain, was expensive and was somewhat under-powered by modern standards. At about this time, I came across an excellent book by Warren Gay, VE3WWG **[6]**, which described how to use, with the *gcc-arm* compiler and *FreeRTOS*, the ubiquitous little STM32F103 boards, commonly called the 'blue pill', which have been manufactured in the thousands. This small board is a completely assembled and functional 32-bit single-board processor sub-system that can be bought online for less than US$3. In this article, I am going to describe how I used this processor as the basis for a very accurate vector impedance analyzer, which covers the frequency range from below 1 MHz to 600 MHz, and that can be built for less than about US$50.

This impedance meter needs an external computer to display its output. This external computer can be as simple as a Raspberry Pi Zero W that costs just US$10, or it could be an old laptop or desktop machine. The impedance meter is powered by and communicates with the external computer via a single USB cable. **Figure 1** shows an example of the display from a Raspberry-Pi Zero W displayed on my main desktop computer via Virtual Network Computing (VNC), see **[7]**. In this case, the impedance meter is measuring the inductance and Q of a small inductor that I wound on a binocular core.

Figure 1 — Example of the display from a Raspberry-Pi Zero W sent to a desktop computer via Virtual Network Computing.

The impedance meter is built on a single double-sided printed circuit board (PCB) and does not require any significant construction or machining skills. It is mounted inside a Hammond die-cast box. The instrument has two BNC test ports. One of these is used as the measurement port for measuring impedance; the second is a test port that can be used for measuring the transfer response of filters or amplifiers with the first port as the signal source.

Principles of Operation

The basic block diagram of the impedance meter is shown in **Figure 2**. A full schematic diagram is available on the **www.arrl.org/QEX files** web page along with many additional details. The signal source goes to a bridge where the characteristic impedance R, is 50 Ω. The signal is split into a reference arm with an output v_R and a signal arm with an output v_S. These two outputs go into mixers where they are each mixed with the same 3 kHz offset Si5351A tracking local oscillator signal. The resulting 3 kHz mixer output audio frequency signals go on to following audio frequency IF amplifiers. Phase is preserved through a mixer so the phase difference at the IF will be the same as the RF phase difference at the bridge. The amplitudes of the signals v_S and v_R do not matter. The ratio ρ of the two outputs is,

$$\rho = \frac{v_S}{v_R} = \frac{2Z}{Z+R} \tag{1}$$

which can be re-arranged as,

$$Z = \frac{\rho R}{2-\rho} \tag{2}$$

R is a real number, but ρ has both magnitude and phase, and can be represented by a complex number. The resultant complex number Z is the impedance at the test port. The function of the impedance meter is to measure ρ and from it calculate Z.

In Figure 2, the mixers are assumed to have an infinite input impedance. In this impedance meter the SA612 mixers have an input impedance of about 1.5 kΩ, mainly resistive. Since this impedance is effectively in parallel with the upper resistors in the bridge, these resistors should be increased so that their resistance in parallel with 1.5 kΩ will be close to 50 Ω. They should be 51.7 Ω. In the circuit I paralleled a 56 Ω, 1% resistor with a 680 Ω, 1% resistor.

The second BNC socket on this impedance meter is also terminated in 51.7 Ω and connected to a third SA612 mixer with the same local oscillator.

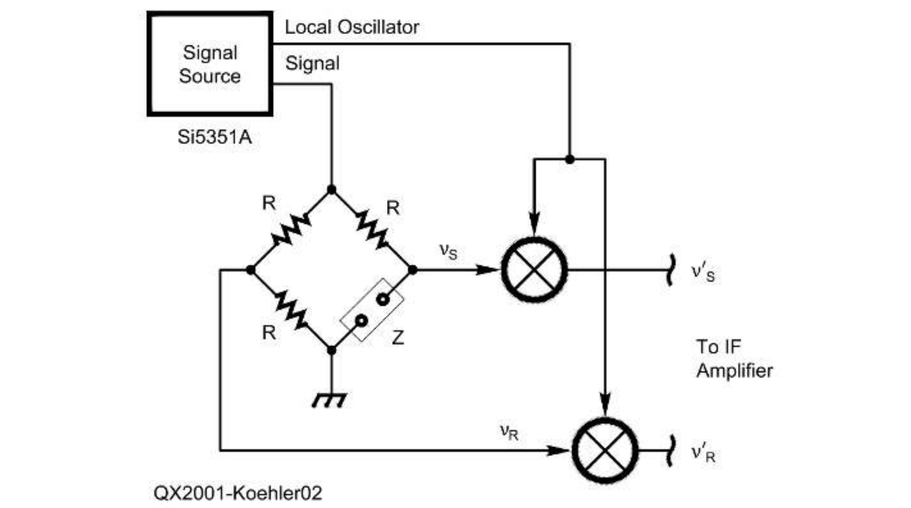


Figure 2 — Block diagram of the impedance meter.

Measuring ρ

I want to describe the algorithm I use for measuring the magnitude and phase of ρ because it differs from what is usually done. In many DSP applications, amplitudes of signals and their phases are measured by taking a number of samples, often some power of two, then 'windowing' the data set, taking the Fast Fourier Transform (FFT), selecting the signal at the desired frequency and determining its amplitude and phase from the transform. To be sure, the FFT is a very efficient algorithm for solving for all the frequency components in the spectrum. However, in this case, we only want to determine amplitude and phase for a single frequency so doing the whole FFT is a waste. Windowing the data reduces the signal-to-noise ratio of the data set because it reduces the amplitude of some portion of it.

In this instrument we know the desired signal frequency; it is the IF. So, we can just do a straight Fourier transform for that frequency. We can omit the windowing if we ensure that the data set consists of an integral number of cycles of the IF waveform. With an integral number of cycles the waveform is continuous in the sense that the data set wraps around itself so there is no discontinuity to cause spectral smearing. The algorithm is described in the **Sidebar** — *Sampling over an Integral Number of Cycles* — in terms of integrals, but for sampled data it is just the sum of samples multiplied by sine and cosine samples at the same frequency.

In the program to measure ρ I chose to sample over exactly 36 cycles of the IF with 25 samples in each cycle. Because there are three signals to be measured — the IF signal, the IF reference signal and the IF second port — there must be 75 samples in each IF cycle for a total of 2,700 samples over the 36 cycles. I chose an IF frequency of 3,042 Hz, so it will take about 11.8 ms to sample the data. Getting magnitudes and phases takes additional processing time; there are about 5,400 floating-point multiplications plus many more additions, and there is also the overhead. The STM32F103 is a general-purpose controller without a special DSP instruction set. I estimate the processing takes about 15 ms thus giving an overall measurement time at any one frequency of about 27 ms. In addition to making the measurements, the STM32F103 must set the frequencies of the Si5351A and do input and output via the USB interface.

Design Considerations

The schematic diagram of the impedance meter is available on the **www.arrl.org/QEXfiles** web page. The USB connection to the outside world, which supplies both power and communications to the unit, is built into the little STM32F103 board. The rest of the circuitry is powered with +5 V taken from one pin of the STM32F103 board and passed through three stages of LC filtering to remove processor noise. I used six-hole bead inductors (see **QEXfiles** for details) from my junk box but any comparable inductor might be used. The three SA612 mixers are powered by this filtered 5 V. A 3.3 V regulator from this filtered 5 V bus supplies power to the Si5351A oscillator and the two op-amps.

The Si5351A has a maximum frequency output of about 200 MHz. The outputs of the Si5351A are square waves and the mixing product of two square-waves is a triangular wave. **Figure 3** shows the plotted output of the A/D conversions for the first four cycles of the waveforms for the three channels of data. For operation higher than 200 Hz, we can use the third harmonic of the signal by setting the IF to be just 1,014 Hz offset. In this case, there will be only 12 cycles of the 1,014 Hz IF signal. However, the third

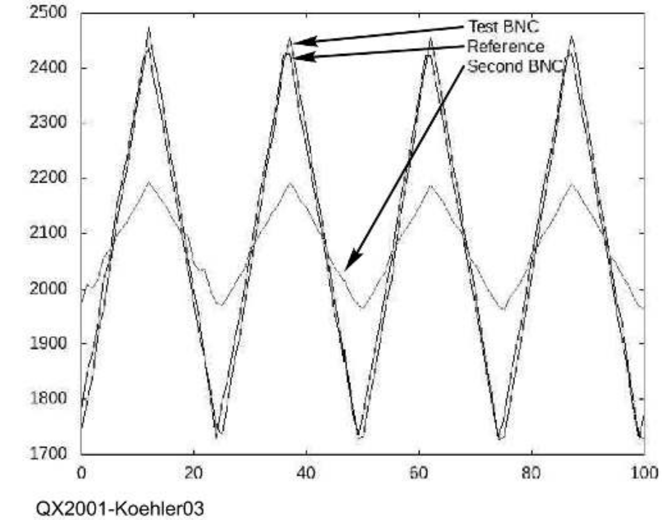


Figure 3 — Plotted output of the A/D conversions for the first four cycles of the waveforms for the three channels of data. The largest trace is the Test BNC, the smallest trace is the Second BNC.

Figure 4 — The PCB fits inside the lid of a die-cast box.

harmonics of this 1014 Hz signal, which is produced at three times the actual Si5351A output, will be at 3,042 Hz and will be extracted by the processing algorithm. The processing algorithm will ignore the main component at 1014 Hz and just measure the third harmonic, which is produced at the third harmonic of the signal frequency. At the upper limit of the Si5351A frequency range, near 200 MHz, the signal is not a very good approximation of a square wave so the third harmonic amplitude is far lower than at frequencies where the fundamental is used, so the accuracy will suffer. However, it is quite usable.

The signal level at the test socket and of the local oscillator is reduced by voltage dividers because the SA612 mixers have some gain and it is desirable not to have too high signal levels, which might put the mixers in a region where distortion occurs. Because of this, some post-mixing amplification was added. The op-amp stage gain for each channel was set to about four. The gain is limited to make sure that the IF signal is not clipped when the test socket is open-circuited. Under that circumstance the IF signal will have its maximum amplitude.

The A/D inputs of the STM32F103 have some capacitance and they are charged from the input signal in just over one microsecond. For this reason a voltage follower was inserted to give the circuit a very low output impedance going into the A/D input of the STM32F103.

Capacitors to ground at the outputs of the mixers, along with their output impedance, provide some low-pass filtering of the IF signal. The chosen op-amps have a gain roll-off with frequency that provides some additional low-pass filtering.

Here is how I arrived at the actual IF frequency to be used, the number of samples, and the number of cycles of data to be sampled. First, I measured how fast the STM32F103 could make a 12-bit sample of any particular A/D channel. To do so, the channel had to be selected, a sample started and then one waits for the A/D conversion to complete. I found it was possible to sample at about 229 K samples per second. I arbitrarily chose to set the number of samples per cycle to 25 for each of the three channels, a total of 75 for each cycle of the IF frequency. Dividing the sample rate by the number of samples needed per cycle gave me an IF frequency of just over 3 kHz. I wanted to have an integral number of cycles of the IF frequency and I wanted that number to be a multiple of three so that it would still be an integral number when using the third harmonic. I arbitrarily opted to make the integral number 36. Choosing the number of 36 sets how long it takes to just sample the data set. I did not want this to take much longer than about 10 ms. I made minor adjustments to the IF frequency until I observed that the sample rate was exactly an integral number of cycles in the data set. This turned out to be 3042 Hz. Since the clock of the STM32F103 board is set by a crystal, one can assume that this will all be the same, with sufficient accuracy, for every board.

The STM32F103 internal A/D subsystem has some built-in hardware that allows one to scan through any number of predetermined channels in sequence and store the data internally via DMA, all without having to set the channels during the process. I did try working with this but, I was unable to get this to work correctly using the Hardware Abstraction Layer (HAL) library provided by STM and using the STM version of the Atollic TrueSTUDIO interactive development environment. Had I been able to use this, I could have sampled the signals at rates approaching one megasample per second, and thus had an IF frequency of close to 12 kHz and a total sample time about 3 ms instead of 12 ms. This would have enabled making about 40 measurements per second instead of 30.

Construction

I used *KiCad* **[8]** to design the circuit and to lay out the PCB. The PCB is 1/32" thick and was designed to fit inside the lid of a Hammond 1590NI die-cast box. **Figure 4** shows how the board is mounted. Prior to board assembly, the bare board should be positioned inside the box lid and the various holes marked with a scriber to locate them for drilling. I drilled the mounting holes for the BNC connector and the two screw holes with a #43 drill and tapped them for a 4-40 thread. The lid of the Hammond box is thick enough for this. The center hole of the BNC connector should be enlarged to about 3/8" diameter. Because there are traces on the bottom of the board, you must make a paper shim to go between the PCB and the inside of the box lid.

The PCB layout is not very dense and most of the passive components are 0805 sized. These are large enough to handle without difficulty. Go over the board with a flux pen before soldering any of the surface-mount components. The 'test BNC' connector is a standard flanged type available from online sources.

When I first designed the board, I thought it might be useful to have some EEPROM memory in case I wanted to store some numbers in it. In the actual developed program this memory was not used, so the memory IC U8 and its associated by-pass capacitor C17 should be omitted. You can also omit low-pass filtering capacitors C33, C34 and C35; I found they were not needed. J1, J2 and resistor R28 are also not needed. They were meant to be used for a push-button switch and an external LED. Similarly I provided J3, a three-pin header, which would provide a serial port connection, and again it was not needed.

There are three SMD tantalum electrolytic capacitors; any convenient values greater than about 20 µF at 7 V or higher are acceptable. The ICs and crystal are listed in **Table 1** along with their Digi-Key® part numbers. It is important that you use the correct sizes of these components.

The 27 MHz crystal specified in Table 1 is the TXC 7M-27.000MEEQ, one of the crystals recommended in the Silicon Labs Application Note AN-551 where the accuracy is specified as ±10 ppm. This means that the oscillator output frequency will be correct within about 1 kHz in the HF region.

The only real difficulty in assembling the PCB is in the region near the Si5351A. The Si5351A has a very small pin spacing so mounting and soldering it must be done very carefully and using a good magnifying glass. The small 27 MHz crystal also needs to be soldered carefully. I found it best to put a very small amount of solder on one (only one) of the four pads, place the crystal and then just touch the edge of the soldered pad with a fine soldering iron tip. After the solder has flowed and the crystal has settled in place, the iron can be removed and the crystal will be held in place. Then carefully, again using the fine tipped soldering iron, you can wick some solder onto the other three pads under the crystal.

Some of the resistors in the bridge area of the circuit use 1% 0805 resistors. To make the 51.7 Ω resistors, I soldered a 56 Ω 1% resistor in the normal manner. I then held the 680 Ω 1% resistor on top of the 56 Ω one and touched one end with a hot soldering iron. After the solder had flowed and the 680 Ω resistor was in place, I touched the other end with the soldering iron to make the solder flow there also. The 49.9 Ω resistors should also be 1% tolerance; 5% is fine for all other resistors.

I made the socket for the STM32F103 board from strips of female header cut from strips of 40 pins. I chose to crush the strip using a diagonal cutter on the 20-th pin from one end, giving me two strips: one with 19 pins and the other with 20. I mounted the 19-pin piece on the left hand side of the row of holes so that the bottom, left-hand hole (when looking at the board, top surface facing you, with the holes for the socket away from you). This hole without a pin in it is not used on the STM32F103 board.

Before soldering the header strips to the board, I cut off each of the pins so that they were not longer than 1/32". After soldering the strips, you should dress down the bottom of the board with a fine file so that any sharp edges will not penetrate through the paper shim between the board and the surface of the lid.

When mounting the BNC connectors, make sure that all the hardware is tight before soldering the center pins of the BNC connectors to the traces on the PCB. Otherwise, if the pins are soldered first before tightening the hardware, the trace near the BNC center pin may get ripped away from the board. I used threaded spacers for the four outer screws of the BNC connectors. I then mounted a small piece of aluminum sheet to make a shield over the region where the RF bridge is located. This was meant not so much for shielding but to ensure that the micro-USB connector and its cable going to the STM32F103 does not get pressed into the region of the RF bridge.

Finally, to make the connection to the outside world, there is a chassis mount female micro-USB connector, with a short cable going to a male micro-USB connector, which is mounted on the bottom of one end of the box. I found this adapter cable in eBay. The final assembly, along with the aluminum shield, is shown in Figure 4.

After assembling any new PCB and before connecting to a computer for the first time, I normally do several checks. Before plugging in the STM32F103 board, I check the 5 V and 3.3 V lines on the assembled board to be sure there is no short to ground. If you have a power supply that can be current limited, connect +5 V power to the board using the +5 V socket pin on the STM32F103 socket. The current draw should be very small, typically about 25 mA. The STM32F103 board takes about another 30 mA for a total of about 55 mA.

The STM32F103 Board

The board can be found on **eBay.com** usually for less than US$3; search for STM32F103C8T6. These boards have a design flaw in that the USB pull-up resistor, which should be 1.5 kΩ, but is usually 10 kΩ or sometimes 4.7 kΩ. This might still work with your computer, but it is best to fix the problem to be on the safe side. The pull-up resistor R10 is a small 0603 sized resistor on the bottom side of the board. It is easy to just add a resistor in parallel to R10 to correct the resistance value. If R10 is 10 kΩ, then parallel it by soldering a 1.8 kΩ resistor between the 3.3V pin and pin A12.

The STM32F103 board is powered via the micro-USB connector on the main PCB. Be careful when plugging and unplugging this connector as the socket on the STM32F103 board is fragile. The STM32F103 board normally includes an installed test program that does nothing but cause the little red LED to blink.

Installing the Firmware

The firmware program for the STM32F103 is programmed via the four-pin connector at the other end of the board from the micro-USB connector. To program it, you need a ST-LINK V2 programmer. This inexpensive little device is available from the same suppliers that supply the boards. The firmware is a binary file *main.bin* available on the **www.arrl.org/QEXfiles** web page package of files for this project. If your computer runs Windows you must install the *ST-LINK* utility **[9]** to program this file into the STM32F103.

There is an equivalent utility program for the Mac called *stlink* available at the Mac App Store. I have no experience with the Mac, so cannot comment on how well it works. For Linux, there is a brief discussion

Table 1. ICs and crystals.

Part #	*Part*	*Digi-Key numbers*
U5	Si5351A	DK 336-3908-1-ND
U4, U6, U7	SA612	DK 568-1204-5-ND
U3, U9	LMV324	DK 296-9569-1-ND
U2	LD1117-3v3	DK 497-1241-1-ND
X1	ABM8 27.0 MHz	DK 887-1328-1-ND

[10] that describes the process for Linux machines. I use Linux and I found the process extremely straight-forward and simple.

Prior to running the *ST-LINK* utility, you must connect the four male pins on the STM32F103 board to the ST-Link V2 programmer. Usually when you order the programmer, you will receive four short jumpers with female sockets at each end. Be very careful when connecting these to make sure that you get the correct pins going to the correct pins at the other end. **Do not plug in both the micro-USB connection and the ST-LINK connection into your computer at the same time!** If you accidentally do this, you will certainly destroy the STM32F103 (ask me how I know this!). You may also damage the ST LinkV2 and you might even damage the USB port on your computer.

After the *main.bin* file has been 'flashed' into the STM32F103 nothing else needs to be done. Unplug the ST-Link V2 and then plug the STM32F103 board into its socket on the impedance meter board.

For those who do not want to go to the trouble of buying an ST LinkV2, pre-programmed STM32F103 boards are available from the author, see the end of this article.

Controlling the Impedance Meter

The impedance meter was designed to be controlled by an external computer via the USB port. From the external computer this communication is two-way. Commands are sent to the impedance meter and data is received from it. The external computer then displays the data in human-readable forms such as graphs, table of values, or as stored files for future reference (Figure 1). The external computer does this using programs for specific tasks.

Even without these programs, it is possible to check using the serial interface to the instrument. When you plug the USB cable from the impedance meter into the USB port of your computer, it will appear as a 'COMx' serial port on the device manager of a Windows computer or as a serial device, usually '/dev/ttyACM0' on a Linux computer. You can interact with the impedance meter using any program that lets you access this virtual serial port, for example *PuTTY* on Windows or *minicom* on Linux. As this is a virtual serial connection, you do not have to set the baud rate on your computer.

The commands to (and response from) the impedance meter are shown in **Table 2**. A command to the impedance meter is a single character, lower or upper case, which may be followed by a number, and the command is then terminated by a CR character produced by the Enter key. Because these commands were expected to be done by an external computer, I did not build any error checking into the impedance meter. An incorrect command is just ignored. Also, the impedance meter does not echo the command character(s).

There are commands to set the frequency,

Table 2. The commands are via a USB serial interface and consist of a single character followed by none or one number.

Command Table

f	n	; set the frequency to n Hz
v		; just show the sampled IF data; NOTE must do 'm' first
m		; make a measurement at the current frequency
n	x	; set number of frequencies to scan to be 'x'
o		; set scan mode to logarithmic
l		; set scan mode to linear
u	n	; set upper frequency bound for scan to n
b	n	; set lower frequency bound for scan to n
g		; start the scan
p		; print status

Sampling over an Integral Number of Cycles

Consider a sine wave with arbitrary amplitude and phase that has been sampled over an integral number of cycles. Let this signal sine wave be given by:

$$s = A\sin\left(\omega t + \phi\right)$$

Let's now multiply this by a sine wave of the same frequency and zero phase, and integrate it over the same integral number of cycles:

$$ss = A\int_0^{nT} \sin\left(\omega t + \phi\right)\sin\left(\omega t\right)dt$$
$$= \frac{nTA}{2}\cos\phi$$

where

$$T = \frac{2\pi}{\omega} \; .$$

Similarly, if we multiply this wave with a cosine wave of zero phase and integrate it over the same number of cycles, we get the sum,

$$cs = A\int_0^{nT} \sin\left(\omega t + \phi\right)\cos\left(\omega t\right)dt$$
$$= \frac{nTA}{2}\sin\phi \; .$$

From these two sums, we can calculate both the amplitude A and the phase ϕ of the signal waveform,

$$A = \frac{2\sqrt{cs^2 + ss^2}}{nT}$$

and

$$\phi = \arctan\left(\frac{cs}{ss}\right) \; .$$

In the case of digitized signals for which we only have samples, the sums will also depend on the number of samples taken over the integral number of cycles. This needs to be taken into consideration when calculating the amplitude. It does not matter for the phase; the magnitude of both sums is multiplied by the same factor and, since we are taking only the ratio of these two, the magnitudes cancel out.

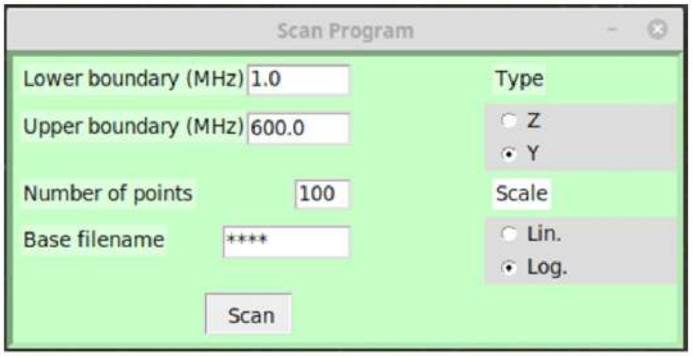


Figure 5 — GUI interface for the Scan Program.

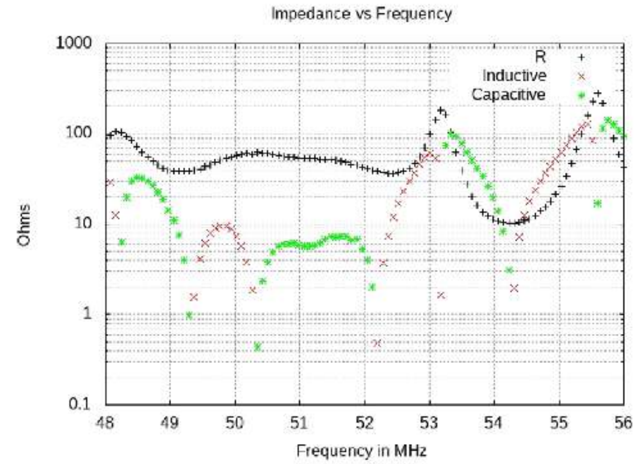


Figure 6 — An impedance plot output for the impedance Scan Program.

make a measurement, start a scan, etc. A scan is defined as making a number of measurements at discrete frequencies between two frequency limits. For a scan, the starting frequency, the end frequency, and the number of steps in frequency must be set before the scan is started. There is also a command to ask the impedance meter to output the raw digitized values of the 2,700 A/D conversions used in a measurement. This command was used to collect the raw data shown graphically in Figure 3.

The 'm' command (measure) causes the impedance meter to start a sample sequence, and calculate amplitudes and phases. The results are output as a data stream of seven numbers. The first number is the frequency in Hz. The next six numbers are three pairs of amplitudes and phases. The order of the number pairs is 'test BNC', 'internal reference', and 'second BNC'. The amplitudes are on an arbitrary scale and the phases are relative to the instant at which the A/D conversion was started. The actual information we need to calculate the impedance are the *ratios* of the amplitudes: 'test BNC to reference', and 'second BNC to reference', and the phase *differences*. It would have been trivial to do these calculations within the impedance meter and just output the ratios and differences. However, I wanted the freedom of choosing how to handle the data in the external computer rather than building it into the firmware of the impedance meter itself.

Programs for the External Computer

Most hams have Windows-based machines but there are significant numbers who use Linux or Mac computers. I use Linux exclusively and I originally wrote the programs in Linux. My friend Tom Alldread, VA7TA, has adapted them to the Windows 10 OS. The complete files for both versions, along with installation instructions, can be found on the **www.arrl.org/QEXfiles** web page. Unfortunately, I don't know anyone who uses a Mac and is proficient in Python, so Mac users who wish to duplicate this instrument might consider buying a Raspberry Pi Zero W, installing the Linux programs in it, and then accessing it via VNC. Instructions for doing this can be found in the same package as the Windows and Linux files.

(a) Calibration Program

If everything were perfect inside the impedance meter, we could just measure the amplitude and phase of the ratios, apply **Eqn (2)**, and we would get the impedance at the test socket. However, components have tolerances, the reference resistance might not be exactly 50 Ω, the measured A/D voltages from the three mixers might have different gains and phase shifts. So, we must calibrate the system by connecting a good load to the test socket. If the load is precisely 50 Ω resistive at all frequencies, the value of ρ should be precisely 1.0 with 0° phase. In practice it will not be, so we must measure it at all frequencies and use this measured value to determine a calibration factor that must be multiplied to the measured ratios. A Linux script file does this by running two Python programs to step through the frequency range and measure the correction factor at every integral MHz. To improve the accuracy, the program measures this correction factor 100 times at each frequency and uses the average value. Thus it makes 100 measurements at 600 frequencies. Even at about 30 measurements per second, it takes considerable time. However, you do this only once.

The program produces a file, 'calibration.txt' that gives the correction factor at each of the 600 frequencies. The correction factor at each integral MHz is used for the frequency interval to the next MHz. This calibration file is used by all the other programs.

To calibrate, put a good quality 50 Ω termination in the test socket, click on the executable script called 'calibrate' and wait for it to finish. You should use the best load that you have or can borrow for this calibration. Remember that a Type-N termination can plug directly into the BNC test socket; they are the same dimensions. Good Type-N terminations are more readily available than good BNC terminations, but be careful, there is little mechanical support in the arrangement.

(b) Scan Program

Program *scan.py* sets up and executes a frequency scan and then plots either the admittance or impedance as a function of frequency. **Figure 5** shows the GUI interface to the program. As an example, the scan limits can be set as 48 to 56 MHz, 100 data points, 'Type *Z*', and 'Lin scale' before the scan button is clicked. The plot appears on the screen, and two files are written to the hard drive. The base filename of these files is set in the GUI. Clicking on the scan button results in a graph such as **Figure 6** that here shows the impedance of my 5-element 6-m Yagi.

(c) Transfer Response Program

The impedance meter can also be used to measure the transfer response of a filter or amplifier. The dynamic range is very limited but, nevertheless it can be useful. If you want to measure the transfer response of an amplifier, you must put an attenuator pad in the circuit with a loss of equal to the expected maximum gain or you may exceed the linear portion of the response.

The GUI for *transfer.py* is shown in **Figure 7**, and the procedure

Figure 7 — GUI interface for the Transfer Response Program.

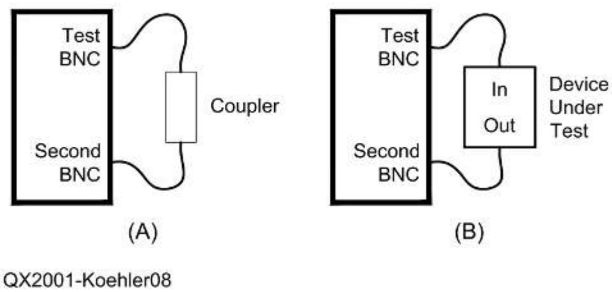


Figure 8 — Calibrate using the test cables with a simple straight-through coupling (A), then measure (B) with the device under test (DUT) replacing the coupling.

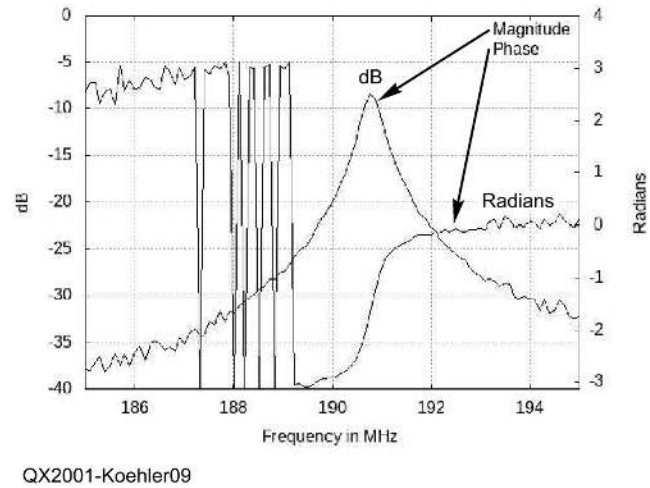


Figure 9 — A Transmission test plot output for the Transfer Response Program. The Magnitude plot peaks between 190 and 192 MHz.

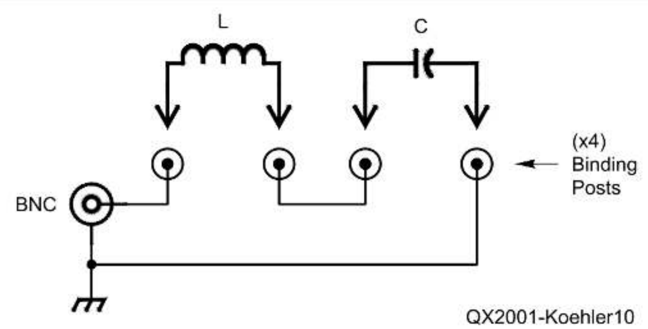


Figure 10 — Circuit for measuring the change of impedance near series resonance to infer the Q.

is illustrated in **Figure 8 (A) and (B)**. The frequency limits and number of samples are first set, a name given for the resulting graphs and text files. Calibrate (A) using the test cables with a simple straight-through coupling. Then press 'Cal'. Next (B) replace the coupling with the device under test (DUT) and press 'Meas'. A graph (**Figure 9**) will appear showing the response of the DUT, here a tunable filter.

Because the amplitude of the third harmonics of the Si5351A outputs are quite low at the upper end of the tuning range near 600 MHz, the dynamic range of the transfer measurements will be limited and the measured graphs may be very noisy.

(d) Inductance Measuring Program

With an impedance meter it is possible to examine how the impedance of an inductor changes near series resonance and from that infer the Q at that frequency. A suitable circuit for doing so is shown in **Figure 10**. In the example shown in Figure 1, the capacitor *C* is 220 pF silver mica with ±10% tolerance. I'd previously measured the inductor *L* as 18.5 µH using my home-made inductance meter.

Program *measure_L.py* measures inductance via a GUI interface seen in Figure 1. First use the 'scan' program, with the frequency range set from 1 MHz to 600 MHz, and with 'Y' selected, and a logarithmic scale. This showed a peak in conductance at about 2.5 MHz. So, in the *measure_L.py* program, I entered 2.5 MHz in the top left box and selected a frequency width of 0.5 MHz in the next box down and finally typed in the capacitance in pF in that box. I then clicked on the 'Plot' button. After a few seconds, a plot of the admittance *vs*. frequency was produced and I could see that the resonance was very narrow and that the peak was centered close to 2.4 MHz. I made the appropriate changes and again clicked on 'Plot'. I then kept reducing the plot width, re-plotting each time until the central conductance peak was close to the center of the graph and also spread out over about the central third of the plot. At that point, I pressed the 'Go' button and the plot shown in Figure 1 appeared. This shows the central peak in conductance along with the program's best fit to the equation for conductance. If the fit looks good, as it does in this case, you may be assured that the values for *L* and Q for that frequency have been calculated correctly. If the fit is not good, it is usually because the conductance peak is not close enough to the center of the plot so you should adjust the center frequency in the GUI interface and re-plot until it is close. After closing the graph, the calculated values for *L* and Q will appear in the GUI interface.

I was not surprised that the inductance values produced by my inductance meter and this impedance meter differed by about 8% since the tolerance of capacitor value was only ±10%. If I wanted an exact value of the inductance, I would measure the capacitor (my capacitance meter is probably correct to within 1%) and redo this measurement with the impedance meter. Another probably more significant reason for the discrepancy is that my inductance meter makes its measurements in the hundred kHz region whereas the impedance meter measurement was made at 2.4 MHz. The relative permeability of the ferrite core probably changes significantly over the frequency region.

If I wanted to know the Q of this inductor at some other frequency, I would replace the 220 pF capacitor with the value needed to resonate at the desired frequency and redo the measurement.

(e) VSWR and Return Loss Program

The VSWR and return loss program *VSWR_scan.py* interface looks very similar to that of the scan program. Again, you set the limits of the scan and the number of data points and then click on the 'Scan' button. An example of the resulting VSWR and Return Loss graph is shown in **Figure 11**. This plot corresponds to the earlier Figure 6 impedance plot of the same 6 m Yagi.

When making VSWR measurements of an antenna, please bear in mind that the impedance meter places a signal on the line to the antenna. At the end of a scan, it will sit at that frequency. Although the signal level going into the antenna is less than 0.1 mW, it may still be enough to annoy your ham neighbors. So, after making tests like this, you should be sure to either turn off (unplug) the impedance meter or at least remove the connection to the antenna.

(f) Measurement of Crystal Parameters

Impedance meters lend themselves to the measurement of crystal parameters, see [5], and this one has sufficient frequency stability to do it very well. Before you can use it for this you must measure how accurately it sets the frequency. To measure crystal parameters, you must know the frequency output of the instrument accurately to within a few Hz. When you build the impedance meter using the 27 MHz crystal I specified, the intrinsic frequency accuracy will be only within about ± ten parts per million. That is not accurate enough. But, the frequency of the impedance meter is very stable so it is necessary to just measure the frequency error at some frequency and then use that to compensate.

First set the impedance meter to a known frequency and then measure the actual frequency it provides using an accurate frequency counter. The easiest way to set the impedance meter to a fixed frequency is to do a scan with the desired frequency as the upper limit. It stays at the last frequency set when it finishes a scan. Suppose that you set the impedance meter scan to end at 15000000 Hz. When you measure with an accurate counter, the output is 15000270 Hz. You then must write a simple text file, named 'frequency_calibration.txt' that contains a single line listing the frequency you set, followed by one or more spaces, then the frequency you measured, all followed by the 'Enter' character. In this example, the single line would be:

15000000 15000270

This file should be stored in the same directory as the program files. You will need to have a fixture to hold the crystal and connect it to the impedance meter. **Figure 12** shows (left) a BNC test connector I found on **eBay**, and (right) the same connector fitted with some machined pins broken from an IC socket, soldered to the connector, both with a spacing to match HC-49 crystals. The GUI for the program shown in **Figure 13** is meant to facilitate measuring a batch of crystals. There is a text entry window where you can type a base file name where the results of the measurement will be stored. If you leave this window at its default setting, no file will be written. Type the nominal crystal frequency in the top left window. With the first crystal from a batch in the fixture, you click the 'Setup' button and after about a half minute, a graph will appear showing the series conductance and phase of the crystal over a frequency range of 4 kHz. The process during this phase measures and compensates for the total fixture capacitance and the baseline of the conductance over the frequency range of the scan. If this looks satisfactory with a smooth graphs you're set to measure the rest of the batch. If not, perhaps the nominal frequency is in error and you might try adjusting that by a few tens of kHz in order to 'find' the crystal. For each batch of crystals, you need to do this setup routine only once, but you must do it at the start of every batch.

Figure 12 — BNC test connector (left) found on eBay, and on (right) a homebrew connector, both with spacing to match HC-49 crystals.

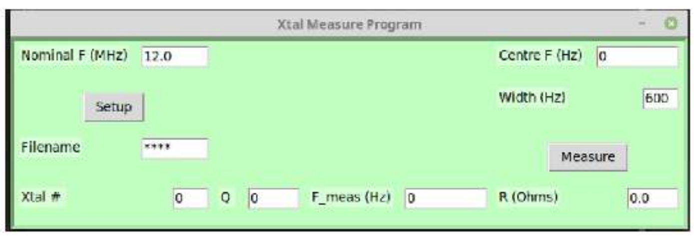


Figure 13 — GUI interface for the Crystal (Xtal) Measurement Program.

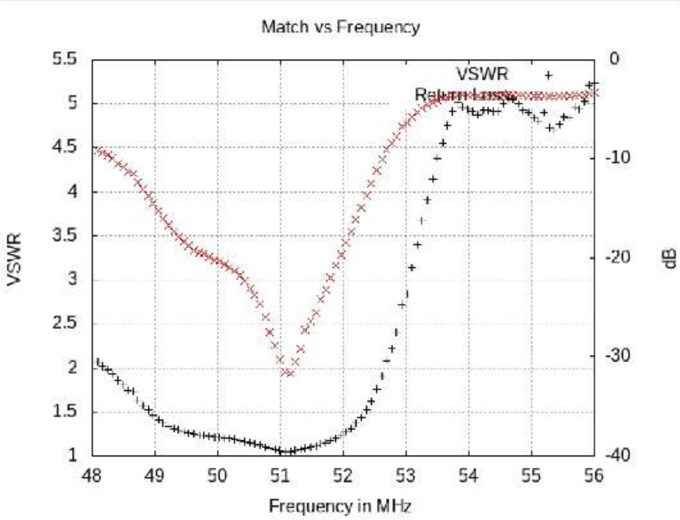


Figure 11 — A VSWR and Return Loss test plot output for the VSWR Program.

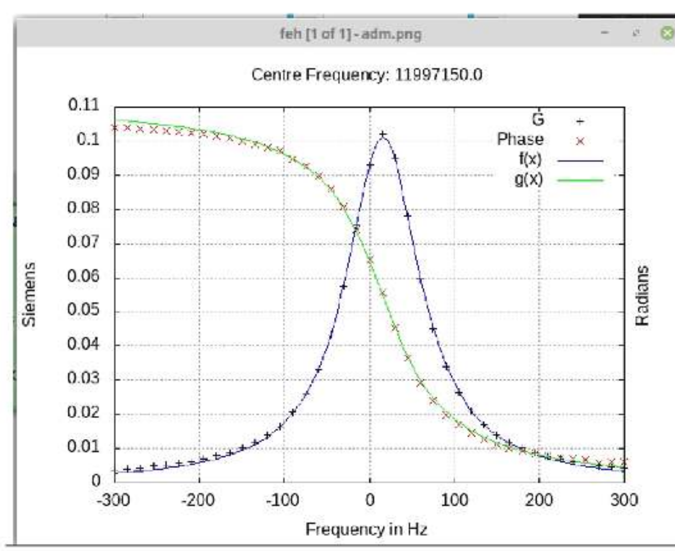


Figure 14 — Measured crystal parameters are denoted by 'x' and '+', solid lines are best fit curves.

Now, with the same crystal still in the fixture, click on 'Measure' and after just a second or two, a new graph will appear showing the same characteristics over the smaller frequency range set by the 'Width' window. **Figure 14** shows an example graph with 'best fit' $f(x)$ and $g(x)$ traces and the actual crystal parameters data points. If you had changed the filename to some text name, say, 'fred', a file named 'fred.txt' will be created and the program will write a line to that file with the crystal number, frequency, Q and series resistance. This file uses commas to separate the values on the line, so it can be imported into a spreadsheet later on. The crystal number will automatically increment (you can change it to what you want) and the next crystal can be put into the fixture and 'Measure' clicked again for each new crystal. On my main desktop Linux machine, a measurement takes about three seconds. If you have not entered a file name, no file will be written but the crystal parameters will be shown on the GUI.

If you're measuring overtone crystals, you will likely want to change the value of the 'Width' to several kHz from the default value of 600 Hz.

The parallel capacitance for each crystal is not measured in this program. Normally, parallel capacitance is measured with a separate capacitance meter for a few crystals in a batch and then the average value of these is used as representative for all crystals in the batch.

Conclusion

I have described a simple, inexpensive and accurate impedance meter (**Figure 15**) that is easy to build and that covers a useful frequency range. The construction does not require any advanced machining skills or any calibration other than that from a good 50 Ω load over the frequency range from 1 to 600 MHz.

I can supply a blank PCB, a pre-programmed STM32F103 board with the USB pull-up fixed and three 6-hole beads, all for US$20 plus shipping (inquire by email, **jark@shaw.ca**).

I thank Tom Alldread, VA7TA, for his help in adapting the programs to Windows. I also acknowledge Warren Gay, VE3WWG, for the valuable aid I got from his book [6]. The GUI programming I did was based on a book by Kenneth A. Lambert **[11]**; thanks.

Author's Note: this instrument was developed in 2018 and submitted for publication in early 2019. Since then, a very similar instrument called the nanoVNA ([**nanovna.com**]) has appeared. The similarity of that RF design just shows that convergent evolution takes place in electronics as well as biology. The author had no prior knowledge of the design or development of the nanoVNA when he developed this instrument.

James A. Koehler, VE5FP, was first licensed as VE5AL in 1953 at age 16. He attended the University of Saskatoon and was awarded the BA (Math), BEng (Physics) MEng, (EE and Physics) degrees. He attended the Australian National University on a scholarship, and received the PhD (Astronomy). On returning to Canada, he became Professor of Physics and Engineering Physics at the University of Saskatoon, staying there until he retired in 1996. He and his wife, Donna, now live on Vancouver Island. Jim's hobbies include electronic instrument design and construction; he has a well equipped workshop for that purpose.

Figure 15 — Impedance meter with the Raspberry Pi Zero W attached.

Notes

[1] P. Kiciak, N2PK, "Vector Network Analyzer," online: **www.n2pk.com**.
[2] T. Baier, "A Low Budget Vector Network Analyzer for AF to UHF," DG8SAQ, *QEX*, Mar./Apr., 2007
[3] M. Knitter, DG5MK,"The DG5MK IV Meter – An Accurate Antenna Analyzer," *QEX*, May/June, 2017.
[4] Y. Kuchura, EU1KY, AA_V3 antenna analyzer, **https://bitbucket.org/kuchura/eu1ky_aa_v3/wiki/Home**.
[5] J. Koehler, VE5FP, "Measure Crystal Parameters Using a Vector Impedance Meter," *QEX*, May/June, 2019.
[6] W. Gay, *Beginning STM32 Developing with FreeRTOS, libopencm3 and GCC*, 2018, ISBN-13 978-1-4842-3623-9.
[7] "Virtual Network Computing", online: **https://en.wikipedia.org/wiki/Virtual_Network_Computing**.
[8] KiCad EDA, **www.kicad-pcb.org/**.
[9] STM32 ST-LINK Utility, **https://www.st.com/en/development-tools/stsw-link004.html**.
[10] **https://fishpepper.de/2016/09/16/installing-using-st-link-v2-to-flash-stm32-on-linux/** .
[11] K. A. Lambert, *Easy GUI Programming in Python*, 2012, ISBN-13 978-0-9859567-0-7.

Technical Note

T-Network Calculator

The recent availability of inexpensive portable antenna analyzers makes it practical for the accurate measurement of various antenna configurations. Once the measurements are complete, the antenna must be matched to the transmitter. Depending upon the station configuration, the impedance of the transmitter and any associated coaxial cable feeding the antenna would likely be 50 Ω. At the point where the measured antenna impedance meets 50 Ω a tuning network needs to be installed.

This T-network calculator is designed to find the values needed to match an antenna to a transmitter or any other complex impedance quickly. It is implemented in Microsoft Excel (see the **www.arrl.org/QEXfiles** web page), and can even be installed on a smartphone.

To use the calculator, type in the input, output and phase values you need to match. For phase, start with -90°, for the input use 50 + *j*0 Ω, the typical transmitter output impedance. In most cases this will calculate a low-pass configuration (good harmonic attenuation) and give you the values needed for a match, along with the rms voltage and rms current requirements.

If some of the calculated component values seem excessive, try changing the phase value starting with -30 through -90°. Alternatively, try +30 through +90°.

It's helpful to be aware of the characteristics of the antenna you are trying to tune. For example, if you are matching a dipole that is resonant at 3.8 MHz, operating at higher frequencies would have a positive reactance. Operating at a lower frequency would have negative reactance. Since reactance must be canceled out, an inductor should be used when operating below resonance. Inductors are much easier to adjust and less expensive than a fixed or variable capacitor therefore, operating at a frequency just below resonance would be easier to manage.

The T-network equations are,

Input leg reactance:

$$X_{in} = \frac{\text{source resistance}}{\tan b} - X_{shunt} - (\text{input reactance})$$

Output leg reactance:

$$X_{out} = \frac{\text{load resistance}}{\tan b} - X_{shunt} - (\text{output reactance})$$

Shunt leg reactance:

$$X_{shunt} = \frac{\sqrt{(\text{source resistance})(\text{load resistance})}}{\sin b}$$

where:

$X_{[leg]}$ is the reactance value of each leg

b is the phase angle required.

The T-network equations for rms voltage and rms current are:

Input leg:

$$I_{in} = \sqrt{\frac{\text{power input}}{\text{load resistance}}};$$

$$E_{in} = I_{in} \cdot X_{in}$$

Output leg:

$$I_{out} = \sqrt{\frac{\text{power input}}{\text{load resistance}}};$$

$$E_{out} = I_{out} \cdot X_{out}$$

Shunt leg:

$$I_{shunt} = \frac{E_{shunt}}{X_{shunt}};$$

$$E_{shunt} = I_{in} \cdot \sqrt{(\text{source resistance})^2 + X_{in}^2}$$

For peak values take the rms values from the equations above and multiply by the provided safety factors. The values will help determine component ratings for various modes of operation.

$$\text{Voltage rating} = (\text{rms value}) \cdot 3.18$$
$$\text{Current rating} = (\text{rms value}) \cdot 1.34$$

The voltage rating safety factor is determined from an unmodulated carrier (1.414) multiplied by modulation factor of 125% (2.25); so 1.414×2.25 = 3.18. Note that 100% modulation would be 2.0 or double the carrier voltage, but would have no overshoot protection.

The peak current safety factor is derived from the same 125% modulation worst case value, and is derived from peak power where: carrier power = 1.0; upper sideband = 0.39; lower sideband = 0.39. Take the square root of the sum;

$$\sqrt{1.0 + 0.39 + 0.39} = 1.34$$

to get the peak current multiplier. At 100% modulation this would be the unmodulated carrier = 1.0; upper sideband = 0.25; lower = 0.25; yielding a 1.22 peak current. — *Best regards, Jerry Whitney, KG2BK, ked2083@gmail.com.*

Coax Loss Calculated Directly in Terms of Impedance Measurements

Coax loss can be calculated in decibels as

$$Loss_{dB} = -10 \log\left[\left|\frac{Z_{oc} - Z_0}{Z_{oc} + Z_0}\right|\right], \qquad (1)$$

where Z_{oc} is the input impedance when the coax is terminated in an open, and Z_0 is the coax characteristic impedance. Alternatively, when the coax is terminated in a short, input impedance Z_{sc} should be used in place of Z_{oc} in Eqn (1). This result appears in numerous publications that cover transmission line theory. Eqn (1) is implemented in the software of several antenna analyzers and vector network analyzers (VNAs).

To use Eqn (1), only *one* impedance measurement is necessary, given a manufacturer- specified nominal value for Z_0. However, two applications of Eqn (1), the first with Z_{oc} and the second with Z_{sc}, may result in different values for loss, both of which are in error. An explanation for this was given by Frank Witt, AI1H, in his May/June 2005 *QEX* paper "Measuring Cable Loss". Basically, a nominal value for Z_0 was used in Eqn (1), not the correct value. Real-world, lossy coax has a complex-valued frequency-dependent characteristic impedance. In Eqn (1), a (2) real-number constant approximation for Z_0 may be justified if coax loss is sufficiently small.

Instead of Eqn (1), Witt suggests using the much-improved loss formula

$$Loss_{dB} = -10 \log\left[\sqrt{|\rho_{sc}||\rho_{oc}|}\right],$$

where

$$|\rho_{oc}| \equiv \left|\frac{Z_{oc} - Z_0}{Z_{oc} + Z_0}\right| \text{ and } |\rho_{sc}| \equiv \left|\frac{Z_{sc} - Z_0}{Z_{sc} + Z_0}\right|$$

In application of Eqn (2), the two reflection coefficient magnitudes are calculated by using a manufacturer-specified nominal value for Z_0, sometimes programmed into the measurement instrument. However, the actual coax characteristic impedance may be significantly different from the nominal value Z_0 — due to moisture contamination of the coax dielectric, for example — and this may cause errors when using Eqn (2). Even if these errors are not significant, the coax loss calculation process would benefit from a formula that is simpler than Eqn (2). In what follows, a loss formula is described that does not explicitly contain Z_0 or require calculation of reflection coefficients.

A simpler, and more appealing, approach is to: 1) eliminate Z_0 from appearing explicitly in the loss formula, and 2) not requiring the calculation of reflection coefficients. Substitute

$$Z_0 \equiv \sqrt{Z_{sc} Z_{oc}}\ ,$$

a correct formula for characteristic impedance, in Eqn (1) to obtain

$$\begin{aligned} Loss_{dB} &= -10\log\left[\left|\frac{Z_{OC}-\sqrt{Z_{SC}Z_{OC}}}{Z_{OC}+\sqrt{Z_{SC}Z_{OC}}}\right|\right] \\ &= -10\log\left[\left|\frac{1-\sqrt{Z_{SC}/Z_{OC}}}{1+\sqrt{Z_{SC}/Z_{OC}}}\right|\right]. \end{aligned} \tag{3}$$

Like Eqn (2), this simpler formula requires the measurement and use of Z_{oc} and Z_{sc}. Unlike Eqn (2), it does not explicitly contain Z_0 or require the calculation of reflection coefficients. Eqn (3) remains unchanged if Z_{oc} and Z_{sc} are swapped, as can be shown by algebraic manipulation. Equivalently, analysis of both the open and short circuit cases lead to Eqn (3). Note that the square root of complex-valued Z_{sc}/Z_{oc} appears in Eqn (3). This is a calculation that is performed without issue by many software packages, including *Matlab* and *Octave*.

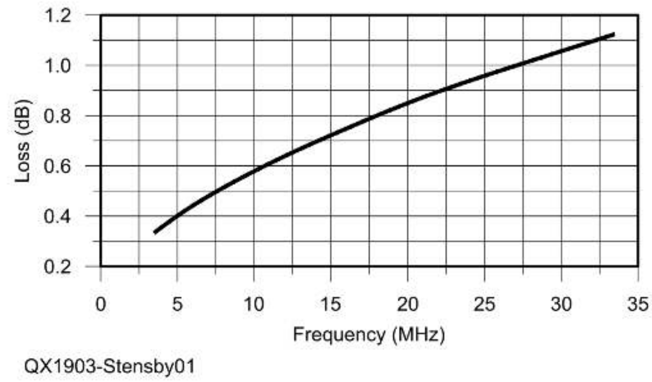


Figure 1 — Measured loss, dB, versus frequency, MHz, in 100 feet of RG 213/U.

Figure 1 depicts results from application of Eqn (3) to impedance measurements taken on 100 feet of old, but still in excellent condition, RG213/U. Over the frequency range shown, a VNA 2180 was used to sweep the coax and make data files containing values of frequency, Z_{oc} and Z_{sc}. With this data, a *Matlab* script was used to evaluate Eqn (3) and produce Figure 1. These computed loss values closely agree with data provided by the cable manufacturer.

As a function of frequency, both Z_{oc} and Z_{sc} exhibit extreme variations in value, especially near resonances in short, low-loss cables. These impedance extremes become smaller as line length (and loss) increase, making accurate loss measurements more practical. If difficulties are encountered, try using a line length that is an odd multiple of $\lambda/8$, where λ denotes wavelength taking coax velocity factor into consideration. For such lengths of low-loss line, the quantities $|Z_{oc}|$, $|Z_{sc}|$ and $|Z_0|$ are of the same order of magnitude.

This note summarizes a coax-loss algorithm that utilizes *two* input impedance measurements or scans. The first impedance scan is made with the coax output open, and the second with the output shorted. At each frequency value, the *ratio* of each scan-value impedance pair is used to *directly* calculate cable loss. This simplified approach eliminates 1) issues associated with use of a manufacturer-supplied nominal Z_0, and 2) the calculation of reflection coefficients. It is believed that this algorithm has not been described in the Amateur Radio literature even though it appears in some electrical engineering texts. It could be implemented in the software of recently-introduced PC-controlled VNAs, a suggestion that is probably the real value of this note. Following the input of a desired frequency range, the PC could command the VNA to scan for both Z_{oc} and Z_{sc}, providing time between scans for changing the coaxial cable termination. Finally, without further user intervention, loss values could be computed and used to construct a plot similar to Figure 1. — *Best regards, John Stensby, N5DF, n5df@arrl.net.*

Send your short *QEX* Technical Note to the Editor, via email to qex@arrl.org. We reserve the right to edit your letter for clarity, and to fit in the available page space. "*QEX* Technical Note" may also appear in other ARRL media. The publishers of *QEX* assume no responsibilities for statements made by correspondents.

Upcoming Conferences

2020 Utah Digital Communications Conference

February 15, 2020
Sandy, Utah
www.utah-dcc.org

The 2020 Utah Digital Communications Conference will be held February 15, 2020, at the SLCC Miller Campus in Sandy, Utah. The conference will be a fusion of amateur radio communications and maker topics. Amateur radio is the pioneer of digital modes. This conference will focus on the amateur radio hobby that surrounds utilizing digital modes or technology in the hobby. Current emerging topics such as digital modes for emergency communications and building your own components.

General schedule, planned demo tables, and registration information can be found on the website.

Call for Presenters & Demo Tables: If you are interested in presenting at the conference, or hosting a demonstration table, complete the form on the website.

TECHCON 2020

February 21 – 22, 2020
Punta Gorda, Florida
www.arrlwcf.org/wcf-special-events/wcftechconference

The 6th Annual TECHCON, the ARRL West Central Florida Section Technical Conference, will be held February 21 – 22, 2020 at the Charlotte County Emergency Operations Center in Punta Gorda, Florida. Friday afternoon seminar, *Introduction to Internet of Things* by Jon Pellant, W1JP. A Friday Evening Social is planned, and the main sessions begin Saturday morning. Registration information is on the website, and website will be updated as more information becomes available.

New Mexico TechFest

Albuquerque, New Mexico
February 29, 2020
www.rmham.org/wordpress/new-mexico-techfest

The 2020 New Mexico TechFest will be Saturday, February 29, 2020 at the New Mexico Veterans' Memorial Event Center located at 1100 Louisiana Blvd. SE, Albuquerque, NM 87108. The event is sponsored by Rocky Mountain Ham Radio, New Mexico.

Join fellow Amateur Radio operators for a day of quality presentations, demonstrations, and instruction provided by some of New Mexico's most experienced technical hams on a variety of emerging and relevant technical topics within Amateur Radio today. The New Mexico TechFest is designed to provide a unique opportunity for all hams interested in the technical aspects of our hobby to advance and expand their technical knowledge and to facilitate technical discussion, collaboration, and ideas with one another.

Ham-specific and presentation-related prizes will be drawn throughout the TechFest. Refreshments including coffee, water, and light snacks will be available. Lunch, catered by a local small business, will be available at an optional cost.

Additionally, a limited number of tables with ac power will be provided for attendees to set up and conduct informal ham-related technology/project demonstrations and technical poster-board presentations between the formal presentations, during lunch, and at the conclusion of the event. Due to space limitations, attendees interested in providing a demonstration or poster-board presentation should submit details to TechFest organizers for review and consideration. Sales and promotions are not permitted.

See website for complete details.

SCALE 18x

March 5 – 8, 2020
Pasadena, California
www.socallinuxexpo.org/scale/18x

SCALE 18x, the 18th annual Southern California Linux Expo, will take place March 5 – 8, 2020, at the Pasadena Convention Center. SCALE 18x expects to host 150 exhibitors this year, along with nearly 130 sessions, tutorials and special events.

SCALE is the largest community-run open-source and free software conference in North America. It is held annually in the greater Los Angeles area.

Present! Submit a proposal for a session via our call for papers. See website.

Sponsor! Sponsorship and exhibitor opportunities are available for commercial and non-profit exhibitors. See website.

Get Involved! Interested in helping to plan and/or to volunteer for SCALE 18x? Send us an email to **staff@socallinuxexpo.org**.

2020 SARA Western Conference

March 27 – 29, 2020
Socorro, New Mexico
www.radio-astronomy.org

The 2020 SARA Western Conference will be held at the Pete V. Domenici Science Operations Center in Socorro, New Mexico, March 27 – 29, 2020.

The keynote speaker will be Dr. Mark McKinnon, Assistant Director for New Mexico Operations at National Radio Astronomy Observatory. In addition to presentations by SARA members, plans include having other speakers from the NRAO SOC in Socorro, and from the UNM Long Wavelength Array (LWA) project. On Sunday, March 29, we will have a tour of the Very Large Array (VLA) site west of Socorro. Additional details will be published online and in the SARA journal as we get closer to the conference date. Register now to avoid the rush and to guarantee a seat at the conference.

Registration: Registration for the 2020 Western Conference is just $80.00. Attendees at the conference must be SARA members; if you are not yet a member, this will cost an additional $20. The fee includes lunch and snacks on Friday and Saturday, and lunch on Sunday. See website for payment options.

Hotel reservations: To be announced; there will be a special SARA rate at one of the hotels in town.

Call for Papers: Papers and presentations on radio astronomy hardware, software, education, research strategies, philosophy, and observing efforts and methods are welcome. The deadline for submitting a letter of intent to the conference coordinator, including a proposed title and informal abstract or outline, is January 15, 2020, and should be emailed to **westernconf@radio-astronomy.org**. Be sure to include your full name, affiliation, postal address, email address, and indicate your willingness to attend the conference to present your paper. Submitters will receive an email response, typically within one week. Formal proceedings will be published for this conference. We will also make presentations available to attendees electronically on a USB drive.

2020 Southeastern VHF Society Conference

April 24 – 25, 2020
Gainesville, Georgia
www.svhfs.org

The 2020 Southeastern VHF Society Conference will be held April 24 – 25, 2020 at the Ramada Inn in Gainesville, GA. Rates and additional conference information will be announced as received – watch website for information.

Aurora Conference

April 25, 2020
White Bear Lake, Minnesota
www.nlrs.org/home/aurora

The 2020 Aurora conference will be held April 25 at the Community of Grace Lutheran Church, 4000 Linden St., White Bear Lake, MN. Aurora, the largest annual gathering of weak-signal VHF'ers in the Upper Midwest, is the annual gathering of the Northern Lights Radio Society. These conferences, first hosted by Jon Lieberg, KØFQA, began in 1984.

In the morning hours, the club runs an outdoor antenna range from 0900 to 1130. The morning also allows for socializing, show & tell, and casual demonstrations. Members are then on their own for lunch. The technical programs start at 1300 and typically run until 1700 – 1730.

If you have a weak signal VHF topic that is of interest to you, or that you would like to present, please contact our Technical Chairman, Jon Platt, WØZQ (**w0zq@aol.com**). If a technical presentation is not to your liking, Aurora also features a poster session.

If you have an interest in weak signal VHF please plan on attending Aurora where you will leave re-energized for this exciting aspect of our hobby. Watch website for details.

2019 *QEX* Index

Features

About the Cover

Errata

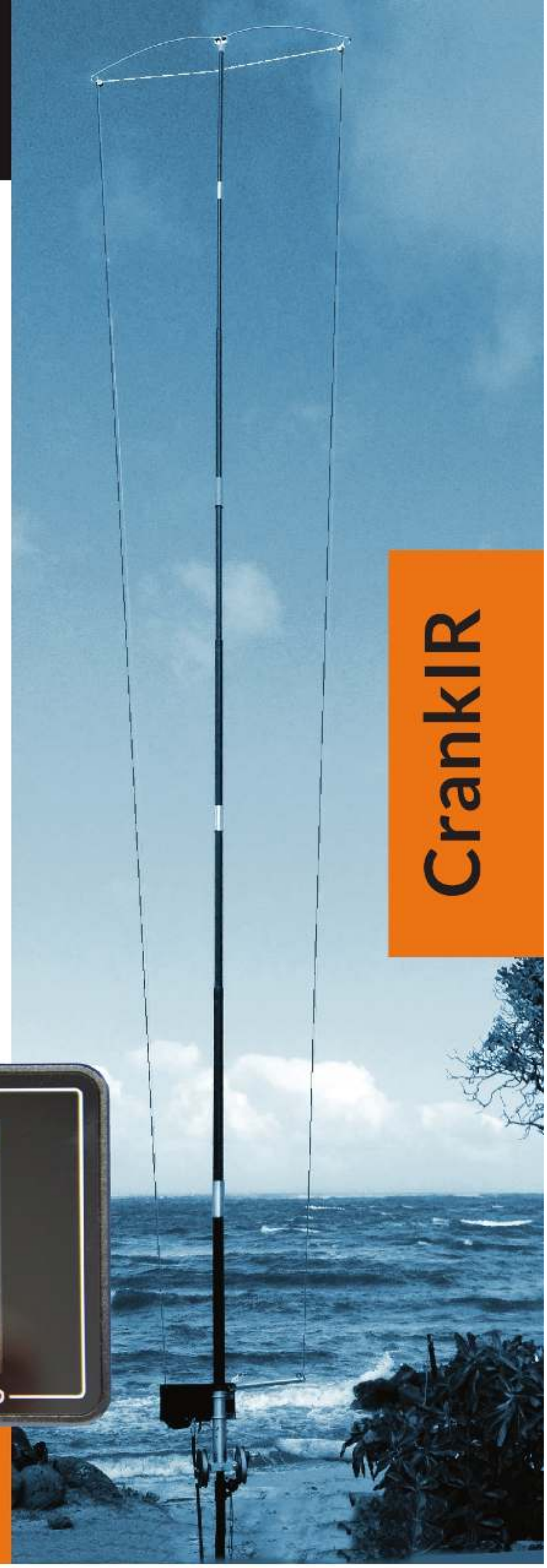
Letters

Perspectives (Siwiak)

Technical Notes

Upcoming Conferences

A PERFECT **PAIR**



Combine our manually tuned, ultra-portable yet high performance CrankIR Vertical with the SARK-110 pocket sized antenna analyzer!

CrankIR

A lightweight, high performance, extremely portable vertical antenna rated at 1500 watts key-down with fully manual operation (no electrical power or controller required). An optional portable tunable elevated radial system is available and its patented folded design allows for a 40% reduction in size with only 0.3dB reduction in gain performance when compared to a full sized antenna. With available versions that cover 80m-2m and 40m-2m (and every frequency in between), the CrankIR sets up quickly and provides flexibility to change the bands quickly. This antenna is the choice of amateur radio operators and emergency communications teams world-wide, in both portable and permanent applications. Consider purchasing one of our SARK-110 battery powered pocket sized antenna analyzers for use with the CrankIR – a custom 3D printed mounting bracket is available to secure the SARK-110 to your CrankIR!

SARK-110

The SARK-110 antenna analyzer is a pocket-sized instrument that provides fast and accurate measurement of the vector impedance, VSWR, vector reflection coefficient, return loss and R-L-C. Typical applications include checking and tuning antennas (such as the CrankIR), impedance matching, component test, cable fault location, measuring coaxial cable losses and cutting coaxial cables to precise electrical lengths. The SARK-110 has full vector measurement capability and accurately resolves the resistive, capacitive and inductive components of a load. The SARK-110 is intuitive and easy to use, and utilizes four operating modes: sweep mode, smith chart mode, single frequency mode and frequency domain reflectometer (cable test).

Ask us about our new 3D printed SARK-110 bracket, designed specifically for the CrankIR! *(Prototype holder shown)*

"We introduced the CrankIR to be a world-class portable antenna – little did we know that scores of amateur radio operators would make this their home station antenna as well!"

– John Mertel, WA7IR
CEO SteppIR Communication Systems

steppIR
COMMUNICATION SYSTEMS

FOR DETAILS ON THESE PRODUCTS AND TO ORDER:

www.steppir.com 425-453-1910