$5

QEX

July/August 2015

www.arrl.org

A Forum for Communications Experimenters

Issue No. 291

W3PM built a 1 to 112.5 MHz VFO using an Si5351A clock generator board. It can be built as a stand-alone device or stabilized using a GPS disciplined oscillator. The GPS version also displays your Grid Locator and time.

ARRL The national association for AMATEUR RADIO™
225 Main Street
Newington, CT USA 06111-1494

QEX (ISSN: 0886-8093) is published bimonthly in January, March, May, July, September, and November by the American Radio Relay League, 225 Main Street, 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 291

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: ARRL Member $24, nonmember $36;

US by First Class Mail:
ARRL member $37, nonmember $49;

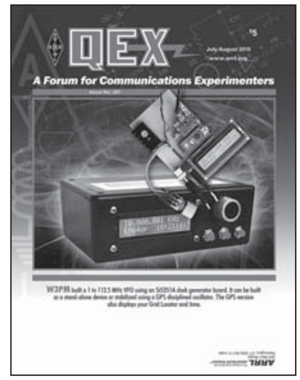
International and Canada by Airmail: ARRL member $31, nonmember $43;

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.

About the Cover

Gene Marcus, W3PM, used an Si5351A clock generator board from Adafruit to build a 1 to 112.5 MHz VFO. You can build this VFO as a stand-alone unit or with a GPS disciplined oscillator to produce a more stable signal frequency. The GPS version also displays your Grid Locator and the time in UTC.



In This Issue

Features

Index of Advertisers

Larry Wolfgang, WR1B

Empirical Outlook

Readers React

In the May/June issue I wrote about the change to a lighter paper stock for *QEX*, and the decision to use the same stock for the cover. I explained the main financial reasons for making the change, and I expected to hear from some subscribers about that decision. You did respond. I had more e-mails and phone calls about the paper change than I have had about any other issue of *QEX* since I became Managing Editor and Editor of the publication. I heard tales of woe regarding covers and entire copies destroyed by the US Postal Service. (Our Circulation Department mailed replacement copies to those who contacted us to say they had received a damaged or destroyed issue. We want to make sure you receive a readable copy of each issue of *QEX*.) I also heard back from one subscriber who wanted to tell me that his most recent *QST* also arrived damaged, so he wanted to assure me that it wasn't just the thin paper of *QEX* that was under attack by the USPS!

I also heard from readers who told me they understand the reasons for the change, and who indicated that they hope the change allows *QEX* to remain in print for many years. Some offered the opinion that they would be happy to pay an increased subscription fee to keep the same heavy cover stock and paper that we had been using.

As Amateur Radio operators, we know that crystals can produce "rock stable" oscillators and extremely "clear" filters, but as far as providing a clear picture of the future, "Crystal Balls" are notoriously unreliable. About all I can say for certain is that any publication has to change and evolve to remain strong, and ARRL wants *QEX* to continue as the solid technical publication you have come to expect. I also foresee plenty of top-notch technical content to ensure that *QEX* will continue to be the premier technical publication in all of Amateur Radio! Keep experimenting, writing, and sending us those fantastic articles, and we will keep publishing them! After all, it is the *content* of *QEX* that we are all so interested in!

Summertime Operating

As I write this editorial summer has not yet arrived, although it is getting close, and the warm weather is most welcome — especially here in New England, where we endured a long, cold, snowy winter and a very cold spring. I am looking forward to plenty of warm, sunny days, and all of the outdoor activities that go along with the summer. I try to make sure that ham radio plays an important role in my enjoyment of the great outdoors, too. By the time you read this, the greatest operating event of the year will be history. I hope you had a great Field Day!

What other operating activities do you enjoy during the summer? Perhaps you like to hike, and enjoy carrying a small radio station along for some trailside operating, or perhaps you enjoy taking a radio out in a boat — whether that be a small craft like a canoe or kayak or whether it is an installation in a sailboat for a weekend or longer on the water. Maybe there is a mobile station in your vehicle for a little operating time while you travel, or even a major installation in a recreational vehicle or second home, to enjoy your hobby while you camp or relax away from the hustle and bustle of everyday life.

Some of that "down time" can also be a perfect opportunity to make some progress on a project that has been on the back of your workbench. Perhaps you need a bit more testing time on your design or maybe you have been working on the documentation, and you are ready to turn it into an article. As always, we are happy to receive your next submission, and to consider it for publication in *QEX*, or perhaps even *QST*. Traditionally, summer is a slow time for receiving articles, and you can turn that to your advantage. You may see your article in print a little sooner when the coffers run low.

What types of articles are you most interested in reading? I try to include a wide variety of technical articles in *QEX*, but in reality, we can only print the articles that you submit. While looking through a couple of years' worth of *QEX*, it is pretty easy to find some common themes, but it is also pretty easy to see some "holes" in our coverage of the technical topics in Amateur Radio. For example, we have an occasional article about microwave equipment, but I am sure there are people who are doing some great work in that area, and I believe many readers would like to read about more of it. Likewise, we don't receive many submissions focused on VHF or UHF equipment or even operating techniques. Are you a VHF/UHF contester? What equipment have you built or adapted for your station? What operating tips might you have to offer to other hams? Your story might even inspire some new devotees to your corner of our hobby.

Have you put together an Amateur TV station, or perhaps even an SSTV system for HF? We haven't covered those topics in QEX — at least not for a long time. Have you been experimenting with digital TV? Do you think our readers might be interested in the work you have been doing with Amateur satellites? I know there are Amateur Radio operators doing so many fascinating things with this great hobby, and we have readers who are interested in hearing about it. You may have put together a club presentation, or even given a talk at a local hamfest. How about adapting that to share with *QEX* readers? We would love to hear from you.

Have a great summer, with Amateur Radio!

Gene Marcus, W3PM/GM4YRE

113 Wickerberry Lane, Madison, AL 35756: **w3pm@arrl.net**

An Arduino Controlled GPS Corrected VFO

A VFO that provides 1 to 112.5 MHz signals on two independent outputs. Use it as a stand alone unit or with a GPS receiver to improve frequency accuracy. UTC and six digit grid square locations are also displayed in the GPS Mode

This project began with the purchase of an Si5351A clock generator breakout board for less than $8 from Adafruit Industries. Designed as a substitute for crystal oscillator clocks, it features three output ports for frequencies between 8 kHz and 160 MHz. Although the board is specified for a wider bandwidth, this project is limited to 1 through 112.5 MHz.

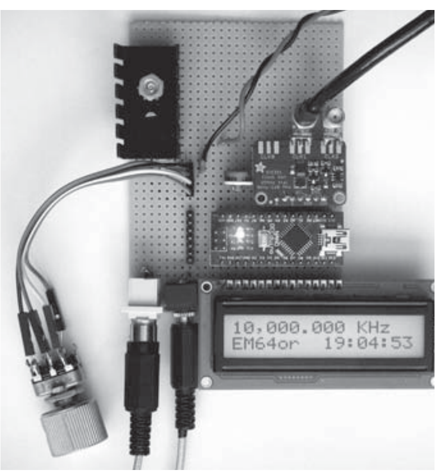


Figure 1 — I constructed the VFO on a piece of perfboard. The heatsink shown at the top left corner of the board is for the 7805 voltage regulator. The regulator is not required for the basic non-GPS configuration of the Si5351 VFO project, if used without the display backlight. The VFO output signals connect to the CLK1 and CLK2 connectors at the edge of the Si5351 board. The Arduino Nano is between the Si5351 board and the display board. At the bottom left of the perfboard are the GPS connections. Also note the rotary encoder to the left side of the perfboard. The pushbutton switches were not included on this version of the VFO.

Figure 1 shows my project, built on a piece of perfboard. The Si5351 board is the top board on the right side of the perfboard. Just below that is the Arduino Nano board I used to control the oscillator. This version uses a rotary encoder to set the operating frequency. You can see the encoder off the left side of the board. Figure 2 shows a completed unit, packaged in a plastic project box. The Resolution, Band Select, and Reset pushbuttons are on the right, just below the rotary encoder.

The Si5351A board does have limitations. Although it is a highly capable and stable board, the output is a square wave with odd harmonic frequencies present in the output. The square wave output does make a good source for some mixers. Phase noise is also higher than other popular programmable signal sources. A quick search of the Internet will yield a wealth of data concerning the performance of the Si5351A IC. Builders are urged to consider phase noise and crosstalk limitatinos before using this IC in their project.

A simplified version of the VFO can be built without the GPS module. Figure 3 shows the circuit for this configuration. Figure 4 shows the schematic diagram for the complete circuit, with GPS module, rotary encoder and pushbuttons.

Unlike a GPS disciplined oscillator (GDO) using a phased lock loop (PLL), this project uses a GPS 1 pulse per second (pps)

Figure 2 — Here is a completed VFO project, housed in a plastic project box. The Resolution, Band Select, and Reset pushbutton controls are located just below the rotary encoder.

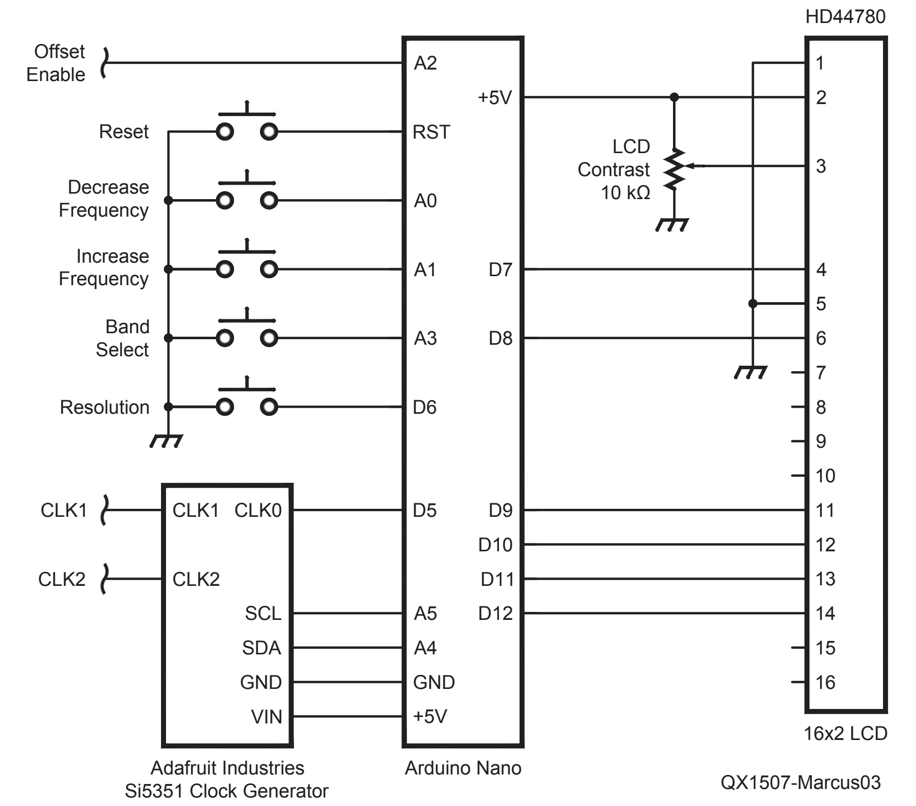


Figure 3 — This is the schematic diagram for the basic non-GPS configuration of the Si5351 VFO project. Either an Arduino Nano or Arduino Uno can be used.

output to act as a precision frequency counter gate. An Arduino Nano (or Uno) is used as a frequency counter to calculate a correction factor to use when programming the Si5351 board. Although not as accurate as a GDO, this simple method provides a variable signal source from 1 to 112.5 MHz with an uncertainty of better than 1 part in 10^7.

The Arduino provides the processing power required to calculate the frequency, control the Si5351A board, serve as a UTC timekeeper, and as an added bonus, calculate your 6 digit Maidenhead grid square locator.

Favorite operating frequencies are stored in software and are accessed by the ***Band Select*** pushbutton. Each band may be configured in a VFO only or VFO/LO combination. The Si5351A ***CLK1*** output port is the VFO and ***CLK2*** is used as an LO. The displayed frequency is arithmetically corrected when the VFO/LO configuration is used. Multiple bands can be configured in this manner. A programmable offset function allows you to use the unit for transceiver operation.

Theory of Operation

The Si5351A is based on a PLL/VCXO and high resolution MultiSynth fractional divider architecture. The Si5351A board can generate any frequency up to 150 MHz on each of its outputs. System short and long term frequency uncertainties are attributed to the onboard 25 MHz clock.

One of the three Si5351A outputs (CLKØ) is programmed to 2.5 MHz and routed to the Arduino's frequency counter input port (pin D5). The 1 pulse per second output from the GPS receiver is routed to the Arduino's interrupt 0 input port (pin D2) to act as a counter gate. The Arduino counts the 2.5 MHz input over a 40 second gate time, resulting in a 100 MHz total count. This count is used to recalculate the 25 MHz clock frequency. Total system uncertainty, including calculation resolution limitations and clock drift during counter gate time, is better than 1 part in 10 million. The VFO frequency is updated every 40 seconds, or when the frequency is changed. Typical frequency uncertainties versus time for GPS and non-GPS configurations can be seen in Figures 5 and 6.

When the unit is first turned on, the GPS processing routines are enabled to determine the correct UTC and 6 digit Maidenhead grid square location. When the software determines that valid data has been received, the GPS processing routines are disabled. At this point, frequency accuracy and time is maintained by the GPS 1 pps signal. GPS NMEA processing is turned off to eliminate processing conflicts and resultant frequency counter errors.

Excellent library routines are available on the internet to simplify Si5351A frequency programming. Instead, I chose to program the Si5351A PLL and MultiSynth functions directly without the use of library routines. The resultant code is very simple compared to other routines, but works quite nicely in this application.

I found I could easily program the Si5351A board up to 150 MHz using PLL divider techniques. Unfortunately, PLL divider techniques create glitches each time the frequency is changed. Fixed PLL frequencies using MultiSynth division provide glitch-free tuning, but the frequency range is limited to 112.5 MHz using this method.

Options

The unit may be built without the GPS receiver and used as a stand-alone VFO (refer to the software installation instructions). If used in the stand-alone mode, you can expect a drift rate of approximately 1.6 Hz / °F (2.8 Hz / °C). Operating the VFO in stand-alone mode also provides an excellent means to test the project prior to connecting the GPS related hardware.

The frequency is controlled either by the rotary encoder or the frequency up/down pushbuttons. You may want to include either the encoder or pushbuttons, or both.

Construction

Construction of the VFO unit is not critical provided adequate RF layout techniques are used. Do not use long unshielded wires for RF and GPS 1 pps connections. I first built the system using a solderless breadboard without any problems. Later, I transferred the circuit to a RadioShack perfboard.

The Si5351A board and LCD may be directly powered from the 5 V pin of the Arduino. A separate 5 V DC source is required if you use the LCD backlight and/or GPS receiver. A 7805A voltage regulator with a small heat sink works nicely. LCD backlight current requirements vary by manufacturer, so circuit details are not included here.

GPS antenna location is critical. A solid GPS signal is necessary to ensure consistent system operation. You should try to locate the GPS antenna with a clear view to the sky, away from noise sources. The GPS receiver requirements are 5 V operation, 1 pps output, and 4800 baud NMEA data output in either $GPGGA or $GPRMC format.

Software Installation and Setup

The Arduino download website (**http://arduino.cc/en/Main/Software**) outlines installation instructions for the first-time Arduino user. My Arduino Si5351A_vfo.ino file is located at: (**www.knology.net/~gmarcus/Si5351.html**). This file, current at publication time, is also available for download from the ARRL *QEX* files web page.[1]

[1]The software code for the Arduino Controlled GPS-Corrected Dual Output Si5351A VFO, current as of the publication date of this issue of *QEX*, is available for download from the ARRL *QEX* files web page. Go to **www.arrl.org/qexfiles** and look for the file **7x15_Marcus.zip**.

Important note: The serial input port (pin 0) is used for both GPS and USB serial data. Do not attempt to upload software into the Arduino Nano while receiving GPS data. Disconnect the GPS NMEA data line before uploading software.

The software will allow the displayed frequency to be arithmetically corrected when both output ports are used in the VFO/LO configuration. Multiple bands may be configured in this manner.

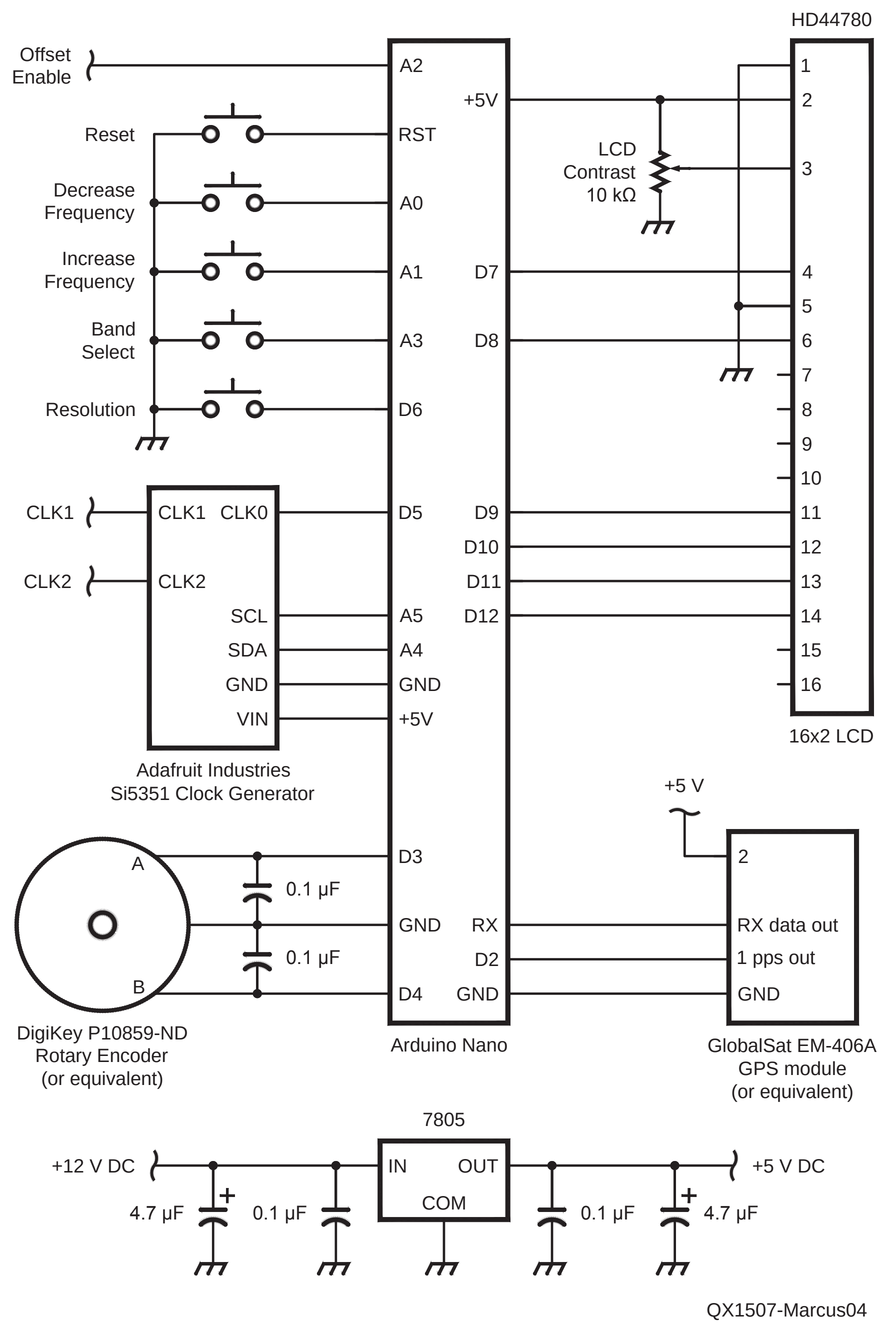


Figure 4 — The GPS corrected version of the Si5351 VFO project is shown in this schematic. The rotary encoder can be included to control the operating frequency. Either an Arduino Nano or Arduino Uno can be used.

```
const unsigned long Freq_array [] [3] = {
{ 7030000,0,0 },                      // CLK1=7.030 MHz, CLK2=0 MHz, Display=7,030.000 kHz
{ 1810000,0,0 },
{ 3560000,0,0 },
{ 7040000,0,0 },
{ 10106000,0,0 },
{ 14060000,0,0 },
{ 18096000,0,0 },
{ 21060000,0,0 },
{ 24906000,0,0 },
{ 28060000,0,0 },
{ 50060000,0,0 },
{ 5286500,8998500,1 },       // CLK1=5.2865 MHz, CLK2=8.9985 MHz, Display=14,285.000 kHz
{ 5016500,9001500,2 },       // CLK1=5.0165 MHz, CLK2=9.0015 MHz, Display=3,985.00 kHz
(0,0,0)

};
```

Near the beginning of the Arduino sketch you will find the following variable that defines the Band Select configuration:

Depressing the Band Select pushbutton will step through the programmed frequencies. Each entry follows the format "{CLK1 frequency, CLK2 Frequency, math command}." You may modify, delete, or add to the Band Select list. The last entry in the list must be (0, 0, 0). The rotary encoder or frequency up/down pushbuttons will only control the CLK1 frequency.

All frequency entries are in Hz. The math command controls how the frequency is displayed. A "1" will add CLK1 and CLK2, and a "2" will subtract CLK1 from CLK2. A "0" results in no change.

For example: {5286500,8998500,1}, will result in:

```
CLK1 output: 5.286500 MHz
CLK2 output: 8.998500 MHz
LCD display: 14.285000 MHz
```

The ***Offset*** variable controls the frequency offset. When the Arduino pin A2 is held low, the programmed offset in hertz is added or subtracted. Any positive or negative value of hertz can be used to program the frequency offset.

There are two ways to configure this project: either as a GPS corrected frequency source or as a stand-alone unit without GPS correction. Locate the variable ***GPSflag*** near the beginning of the Arduino sketch and set the variable to either "1" for GPS correction or "0" for non-GPS operation.

If you choose to use the project without GPS correction, you can enter a correction factor to allow for greater frequency accuracy. To calculate and enter this correction factor, perform the following steps:

1) Connect the VFO to a frequency counter
2) Set the VFO to 25 MHz
3) Note the measured frequency in Hz

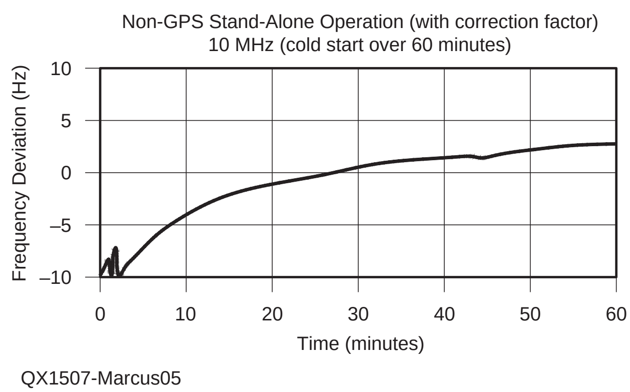


Figure 5 — This is a graph of the frequency versus time over the initial 60 minutes, for the output from the non-GPS version of the VFO project. A constant room temperature was maintained during the data collection. The drift rate is approximately 1.6 Hz / °F (2.8 Hz / °C).

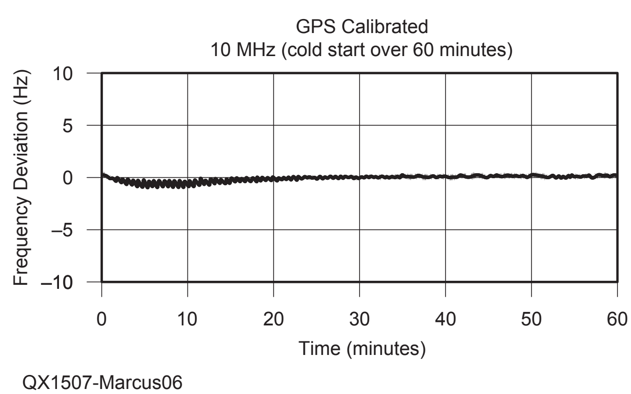


Figure 6 — Here is the graph of the frequency versus time over the initial 60 minutes for the GPS corrected VFO. A constant room temperature was maintained during this data collection. Frequency corrections to bring the frequency back to normal are shown during the early minutes of warm-up.

4) Subtract 25 MHz from the counter reading

5) Note the difference in Hz (such as –245)

6) Locate the variable ***CalFactor*** in the Arduino sketch and enter the value.

Frequency uncertainty without GPS calibration and without the calculated correction factor is normally less than 1 kHz.

Operation

When first turned on, you will see "Waiting for GPS" displayed on the LCD. See Figure 7. (If GPS correction is not used, the second line of the LCD will continually display the frequency step resolution, as shown in Figure 8.) It may take a few minutes for the GPS receiver to lock and obtain valid NMEA data. After the GPS receiver obtains valid data, the "Waiting for GPS" display will be replaced with the (UTC) time and your 6 digit grid square locator. The system is now ready for operation. See Figure 9.

Depress the resolution pushbutton to select the frequency step. When the button is depressed the selected resolution will be displayed for a few seconds on the second line of the LCD. See Figure 10. The frequency may be changed either by using the frequency up/down buttons or the rotary encoder. Depressing the Band Select pushbutton will step through the programmed frequencies.

Experimentation

This is an open source project. You are encouraged to experiment and improve upon the system operation. Simple system improvements such as using a 4 × 20 LCD to display additional data can be easily implemented. Variables containing latitude, longitude, and the number of satellites in view exist in the ***GPSprocess()*** subroutine. More complex updates such as varying the CLK2 LO frequency may also be incorporated.

Gene Marcus, W3PM/GM4YRE, was first licensed in 1963 as KN3YVP, and has held an Amateur Extra Class license since 1968. He was first licensed in Scotland as GM5AQM in 1969. He received a First Class FCC Radiotelephone license in 1977.

Gene completed an ASEE degree program at Penn State University in 1968. After a four year tour as a Cryptologic Technician with the US Navy, he began a 32 year career in the field of precision measurement equipment calibration. In retirement he enjoys experimenting with various RF and microprocessor projects, and enjoys world travel with his wife, Phyllis.

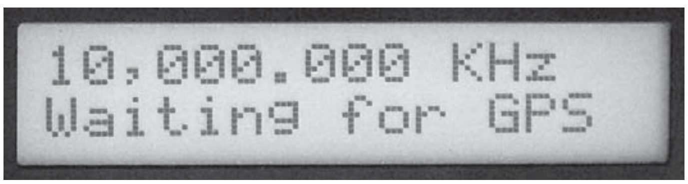


Figure 7 — When you first turn on the GPS corrected VFO, you will see the start-up frequency and the Waiting for GPS message on the display.

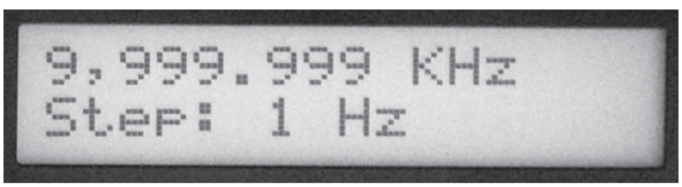


Figure 8 — If you build the non-GPS version, the initial display will show the start-up frequency and the initial frequency step size.

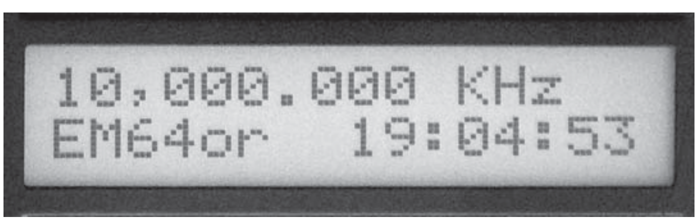


Figure 9 — When valid NEMA GPS data is received by the Arduino, the second line of the display will show your six-digit Maidenhead Grid Locator and the time, in UTC.

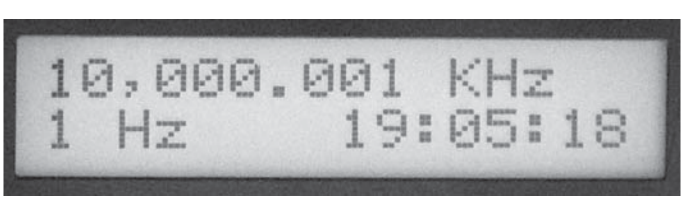


Figure 10 — When you change the frequency step size on the GPS version, the Grid Locator will be replaced by the step size on the display.

Michael Knitter, DG5MK

Weissenburger Web 5, 45701 Herten, Germany: **mkh01@t-online.de**

The DG5MK LCQ-Meter

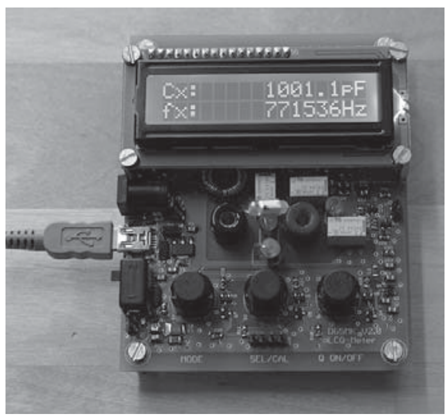
Here is a simple but effective tool to measure inductance, capacitance, and circuit Q.

Measuring resonant frequency and quality factor (*Q*) of LC tuned circuits can become quite complex, time consuming and also expensive in terms of needed measurement equipment. This article describes a very affordable hand-held measurement tool, the LCQ-Meter, which allows among other things measuring resonant frequency and quality factor of parallel tuned circuits and capacitance and inductance of involved components.

Some do-it-yourself Amateur Radio operators own measurement equipment like oscilloscopes, spectrum analyzers, network analyzers, impedance measurement equipment and much more, which would easily compete with some of the standard equipment in electronics laboratories. Nearly all of them own a simple microcontroller-based meter to measure capacitance and inductance because it is cheap, easy to handle, and has an accuracy that fits amateur needs. The original design probably goes back to AADE (Almost All Digital Electronics) around 1997, who still sells the original, very accurate L/C-Meter at an affordable price.[1]

Since then hundreds of clones have been built, finally flooding eBay from China. The funny thing is that there was no evolution at all. Most of the circuits are still based on the same old LM311 comparator oscillator, while some newer ones use microcontroller-based internal comparators, but no change in the principles.

For a recent Amateur Radio project I needed to measure inductors and capacitors, which was easily done with my L/C-Meter, but I also needed to measure some wired LC tuned circuits and was also interested in the *Q* of those circuits. An idea came up to build something new, smart and handy, which could do the job much easier with no additional effort.



The guiding principle for the development of the LCQ-Meter was to build a small piece of equipment at low cost, to meet my ham radio needs. It had to include measurement of capacitance and inductance on isolated components, like the AADE L/C Meters. It also had to include the measurement of capacitance and inductance within a wired LC tuned circuit, the resonant frequency, and as a key challenge, the unloaded *Q* of that circuit. Having small equipment and low cost in mind, there would be tradeoffs in terms of accuracy and measurement range, off course.

After some hundreds of hours of development, simulation, building, testing and measuring, this article describes the current "final" design of the LCQ-Meter.

I will start with some theoretical

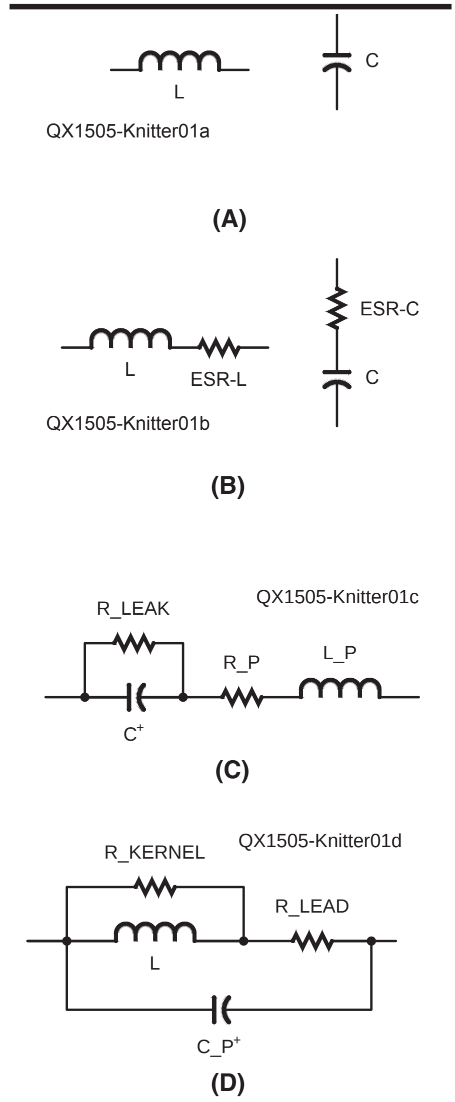


Figure 1 — Different equivalent circuits of capacitors and inductors.

[1]Notes appear on page 26.

background on tuned circuits, and especially on Q measurement, to help explain the overall design. Evaluating the advantages and disadvantages of the different measurement methods will guide us to the current implementation of the LCQ-Meter. I will avoid heavy mathematics, including complex numbers, but a basic framework is given for those readers who are interested in the background.

After looking at the theory behind the measurements, I will describe the overall implementation and software for the microcontroller. After some practical measurement examples the article closes by discussing the accuracy of the LCQ-Meter.

Characteristics of Capacitors and Inductors

Engineers design circuits by using simple electrical models to study the performance of the circuit they wish to build. The same is true for measuring capacitors and inductors.

Using the model of lumped, lossless components, a capacitor is just characterized by one single parameter, which is the capacitance in F. In the case of an inductor, the parameter of interest is inductance in H. Figure 1A shows the equivalent circuit. For many applications this level of detail is just enough.

When dealing with low radio frequencies, that model is not sufficient. Loss has to be introduced, which is done by a parallel or series resistor to convert some energy into heat. This introduces the parameter of equivalent series resistance (ESR). Figure 1B shows the corresponding equivalent circuits with a series resistor.

The magnitude of the quality factor, Q, of the components is introduced as the simple ratio between reactance and resistance at a given frequency:

$$Q_L = \frac{X_L}{R_{ESR}} = \frac{2\pi fL}{R_{ESR}} \qquad \text{[Eq 1]}$$

$$Q_C = \frac{X_C}{R_{ESR}} = \frac{1}{2\pi fCR_{ESR}} \qquad \text{[Eq 2]}$$

Even if Equations 1 and 2 show that Q is frequency dependent, that equivalent circuit still assumes that L, C and the ESR are not frequency dependent. This is a very important assumption, as we will see later that all the calculations of L and C in the original AADE design as well as in the LCQ-Meter design is based on that assumption.

Before we prove whether this assumption is correct, let's have a quick look at the equivalent circuits in Figure 1C and D. They introduce new components for the capacitor to cover parasitic effects like dielectric loss, ohmic (ideal) resistance and inductance for the connection leads, and magnetic kernel loss and ohmic resistance and capacitance for the connection leads on the inductor.

These models still do not include any frequency dependence of the connection lead resistance, however, nor the loss resistance of the magnetic kernel material. In Amateur Radio applications, this sometimes plays a key role, but will not be covered in this article (with one exception which is the parasitic capacitance of the LCQ-Meter's oscillator).

Let's go back to the equivalent circuit from Figure 1B, which is the model we will use. Figures 2, 3 and 4 show some example plots of selected inductors and capacitors

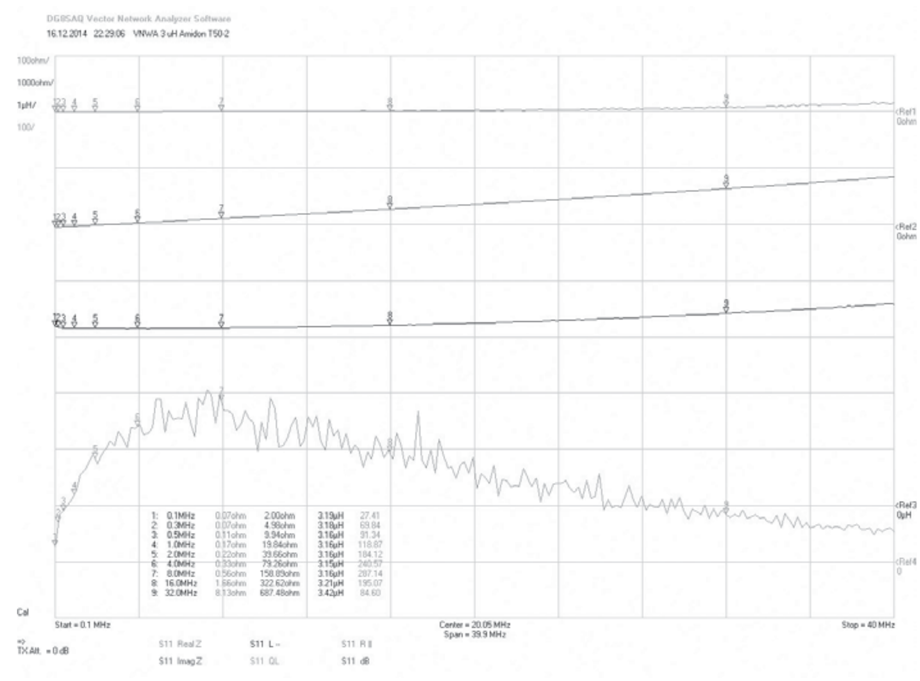


Figure 2— Measurement of a 3 μH inductor on T50-2 Amidon toroid core.

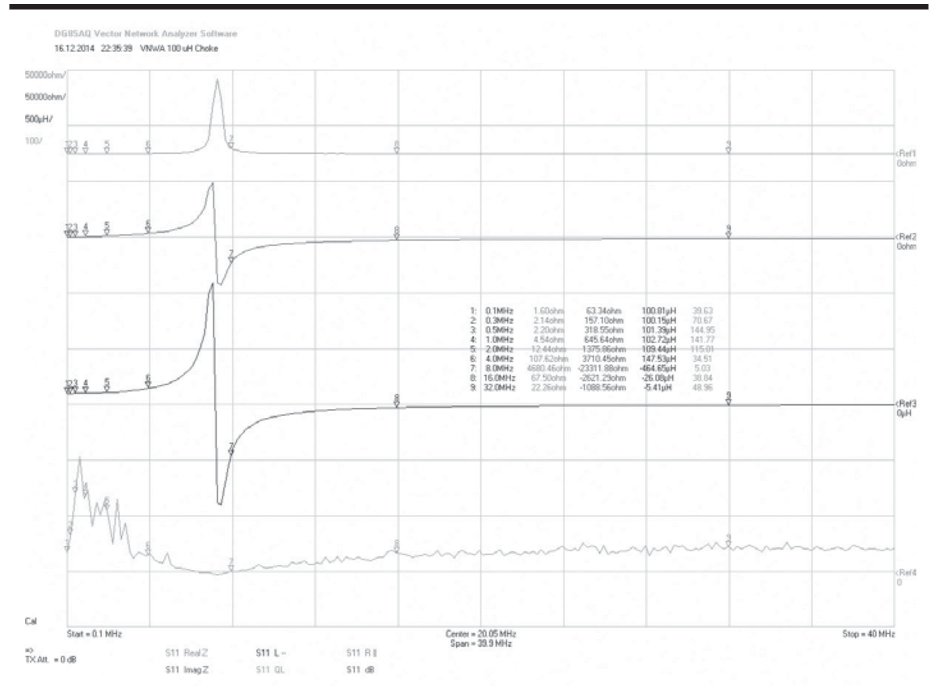


Figure 3 — Measurement of a 100 μH inductor, choke, through-hole model.

done with a vector network analyzer. The plots do show the series resistance or ESR (Real Z), the reactance (Imaginary Z), the inductance or capacitance as well as *Q*, along a wide frequency range.

The inductance of the 3 μH inductor shown in Figure 2 is relatively stable below 15 MHz. From Equation 1, we would expect a linear increase of *Q* with frequency. This is not the case, however, because of parasitic effects. So we cannot simply use a measured *Q* on one frequency to predict the *Q* on another frequency. Also, *Q* is not at all located on a smooth curve. Take Marker 7 at 8 MHz as an example, where *Q* is measured as 287. Now look at Marker 6, at 4.0 MHz, and you will see *Q* measured as 250. With high *Q* inductors the series resistance (ESR) is quite low and only a few milliohm change will impact *Q* significantly.

Figure 3 shows the same plot for a 100 μH choke. Above 2 MHz the inductance increases, going towards infinity at around 7 MHz, and then comes back as a negative inductance, which is the characteristic of a capacitor. What happens is that parasitic capacitance becomes dominant above 7 MHz, and there is a resonance of the inductance with the parasitic capacitance around 7 MHz. Even below 1 MHz the measured inductance is not constant, but the deviation is below 2%, compared to a 100 kHz measurement. We would have seen the same scenario for the inductor from Figure 2 if we go much higher in frequency.

Figure 4 shows a similar plot of a 1000 pF mica capacitor with leads. The capacitance is relatively stable as the frequency increases, but towards 16 MHz it increases by about 8% compared to a measurement at 100 kHz, and increases even more going higher in frequency. We can assume a resonance with parasitic inductance if the frequency would be much higher. Another important fact is that the measured *Q* of the capacitor is at least 1 to 2 decades higher than the *Q* of the inductors. The serial resistance (ESR) even goes below 1 mΩ, and care has to be taken to be sure the measurement method is still able to correctly measure such low resistance.

Using a high quality capacitor in a tuned circuit will reduce circuit *Q* discussions mainly to the *Q* of the involved inductor. Do not assume the same high *Q* for cheap ceramic capacitors!

We can draw some important conclusions from the measurements shown. Using a measuring technique for capacitance and inductance that assumes stability of capacitance and inductance as the frequency changes is limited in accuracy. It makes no sense to expect 0.1% accuracy for such designs.

Using too high of a measurement frequency for isolated component measurement brings in more trouble than it helps. A series of measurements shows that 0.5 MHz to 1 MHz is a good choice. Many people state that the original AADE design measures below 1 MHz because of the old LM311 comparator used, but maybe the frequency was cleverly chosen to avoid a lot of trouble with parasitic effects at higher frequencies?

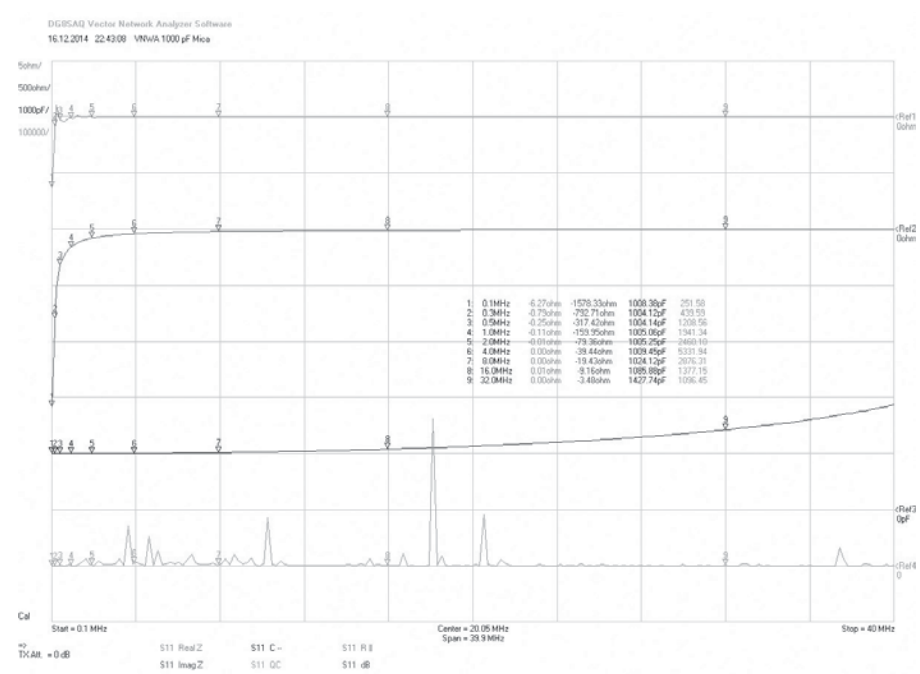


Figure 4 — Measurement of a 1000 pF Mica capacitor, through-hole model.

For Amateur Radio in the HF range, isolated measurement of inductors and capacitors to build tuned circuits doesn't make always sense. The parasitic components have a too much influence. Therefore, the tuned circuit has to be measured as one circuit, as the LCQ-Meter does.

Q does not behave as predicted using simple models, as shown in Figure 1B and by Equations 1 and 2. Instead, a measured *Q* value is for a specific frequency. As in most Amateur Radio designs, the quality factor question is about *Q* of a tuned parallel circuit at resonant frequency. It would be best to measure *Q* in that target environment.

Using very low loss capacitors in a tuned circuit will also allow showing isolated inductor *Q* with an acceptable accuracy.

Measurement of Capacitance and Inductance

There are very different ways to measure capacitance and inductance. Some examples are charge time measuring on capacitors, phase change and amplitude measurement between voltage and current but most of these methods are rather difficult to be implement within a low cost design.

An easy methodology is to measure frequency deviation by introducing the device under test (DUT) into a tuned circuit. Figure 5A shows a resonant circuit with all unknown component values except a reference capacitor, C_{ref}.

The first step is to get the values for C_{fix} and L_{fix} (Figure 5A). When C_{ref} is not connected, the resonant frequency will be given by Equation 3.

$$f_{r1} = \frac{1}{2\pi\sqrt{L_{fix}C_{fix}}} \quad \text{[Eq 3]}$$

When C_{ref} is connected, the resonant frequency will be given by Equation 4.

$$f_{r2} = \frac{1}{2\pi\sqrt{L_{fix}\left(C_{fix}+C_{ref}\right)}} \quad \text{[Eq 4]}$$

Dividing Equation 3 by Equation 4 gives a term that is not dependent on L_{fix}! Solving for C_{fix} gives Equation 5.

$$C_{fix} = C_{ref}\frac{1}{\left(\frac{f_{r1}}{f_{r2}}\right)^2 - 1} \quad \text{[Eq 5]}$$

Now as we know C_{fix}, it is easy to calculate L_{fix} from Equation 3.

$$L_{fix} = \frac{1}{C_{fix}\left(2\pi f_{r1}\right)^2} \quad \text{[Eq 6]}$$

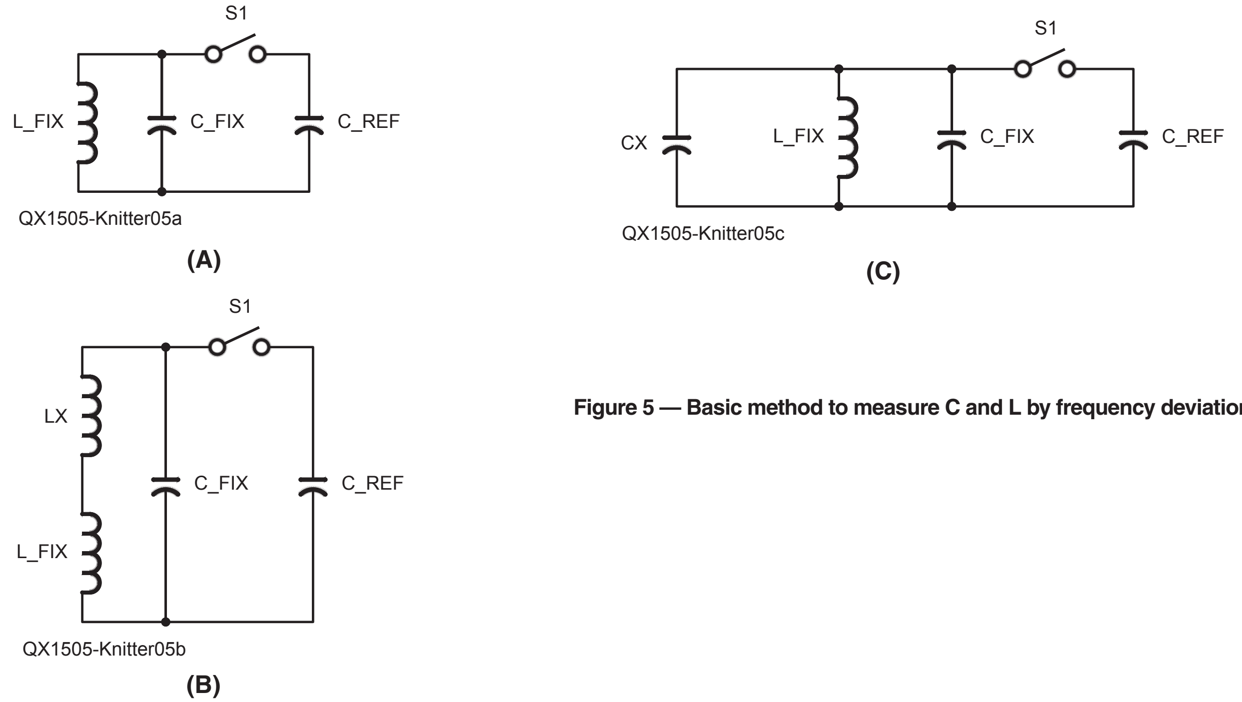


Figure 5 — Basic method to measure C and L by frequency deviation.

All components of the tuned circuit are known now, and the mathematical accuracy is only dependent on the reference capacitor. Don't spend a fortune on a 0.1% reference capacitor, though! Please see the paragraphs about capacitance stability in the "Characteristics of Capacitors and Inductors" section again!

If an unknown inductance, L_X, is added in series with L_{fix}, as shown in Figure 5B, or an unknown capacitance, C_X, is added in parallel with C_{fix}, as shown in Figure 5C, they can be easily calculated by using the same formulas if the new resonant frequency, f_x, is measured and C_{ref} is not connected:

$$C_x = \frac{1}{L_{fix}\left(2\pi f_x\right)^2} - C_{fix} \qquad \text{[Eq 7]}$$

$$L_x = \frac{1}{C_{fix}\left(2\pi f_x\right)^2} - L_{fix} \qquad \text{[Eq 8]}$$

If L_{fix} is disconnected on one side and reconnected with shorted measurement leads for future L_X measurement, or if the measurement leads are attached to C_{fix} left open for future C_X measurement, the calculation for L_{fix} and C_{fix} using Equations 5 and 6 will include any parasitic capacitance or inductance by the measurement leads. Often this activity is called calibration.

This is basically the full mathematics used in LC Meters with a microcontroller to measure the different frequencies.

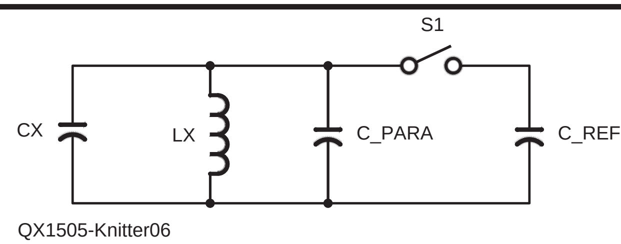


Figure 6 — Method to measure C and L in parallel in tuned circuits.

The methodology is based on the easiest equivalent circuit from Figure 5A, even without considering component losses. As we will see shortly, this is a valid method because any loss of accuracy by introducing a change of resonant frequency due to loss is rather low compared to the other effects we have seen from the VNA plots.

You may also notice that all of the models add an unknown inductance in series with L_{fix} and an unknown capacitance in parallel with C_{fix}. This insures that the new resonant frequency will always be lower than the one from calibration. To stay in the target frequency range, all designs use around 50 µH to 100 µH for L_{fix}, 500 pF to 1000 pF both for C_{fix} and C_{ref}. Again, in principle, any values can be used, but a clever choice helps to avoid any high frequency troubles as discussed earlier.

After a lot of tests and measurements, the LCQ-Meter stays with this design for isolated capacitor and inductor measurement as other choices and methodologies did not show any relevant advantage.

Measurement of Capacitance and Inductance in Tuned Circuits

The methodology to measure capacitance and inductance can be taken one step further to measure both values in parallel in a (parallel resonant) tuned circuit.

Let's assume we have a parallel tuned circuit built with L_X and C_X as shown in Figure 6. There is no C_{fix} capacitance this time, but we have a parasitic capacitance, C_{para}. C_{ref} is a known reference capacitor but of lower size (100 pF) to allow oscillation at much higher frequencies.

If C_{para} were known, we could calculate C_x and L_x directly, with C_{ref} switched in and out of the circuit. With C_{ref} switched out of the circuit, the resonant frequency is f_x, and with C_{ref} switched into the circuit, the resonant frequency is f_y.

$$C_x = C_{ref} \frac{1}{\left(\frac{f_x}{f_y}\right)^2 - 1} - C_{para} \quad \text{[Eq 9]}$$

Knowing C_x allows us to calculate L_x.

$$L_x = \frac{1}{\left(C_x + C_{para}\right)\left(2\pi f_x\right)^2} \quad \text{[Eq 10]}$$

But how do we find C_{para}, which is dependent on the oscillator and circuit board design and probably is also frequency dependent?

The solution is to measure the resonant frequency at different values of L_x, without any C_x in the circuit. In Figure 5A, substitute C_{para} for C_{fix} and L_x for L_{fix}. Then the parasitic capacitance, C_{para}, can be calculated at a given frequency. The result is a pair of C_{para} and f_x values. Doing this with very different L_x values to cover a frequency range from some 100 kHz to 30 MHz will lead to a collection of capacitance/frequency pairs.

In the LCQ-Meter design it showed up that C_{para} is non-linear with frequency change. Using some mathematics to find an interpolation function finally led to Equation 11.

$$C_{para} = k_1 - k_2 \ln(f_x) \quad \text{[Eq 11]}$$

With the current circuit board (version 2.1) and oscillator design, k_1 equals 66.0 and k_2 equals 3.2. Therefore the parasitic capacitance varies from 10 pF at higher frequencies to 25 pF at lower frequencies.

Decreasing this parasitic capacitance was the key motivation to go from circuit board version 1 to version 2. In the former layout the parasitic capacitance was more than double the value with the revised circuit board. Careful circuit board design pays off! Without any relays, the parasitic capacitance was measured at 7 to 8 pF, which is very close to the transistor datasheets and *LT SPICE* simulation.

Equation 11 allows finding the frequency dependent parasitic capacitance for the given hardware design, and therefore, using Equations 9 and 10, it allows calculating C_x and L_x from a resonant parallel tuned circuit.

This seems complicated, but it isn't. The interpolation function has to be found just once for any given hardware design. Use a spreadsheet calculation program of your choice, drop in the value pairs, create a chart, let the program calculate some interpolation functions and just look which one best fits the pairs!

A Few Comments on this Methodology

This methodology shows the net capacitance and inductance within a tuned circuit. Parasitic capacitance of the inductor will be added to the capacitance and vice versa. This is what is really needed in practice to tune a tuned circuit to a target frequency.

To arrive at the true resonant frequency for C_x in parallel with L_x this frequency has to be calculated, and cannot be measured directly because of the parasitic capacitance, C_{para}. The actual frequency is always a little bit lower. The LCQ-Meter software takes care of that variation.

Using this methodology, the measured frequencies with and without C_{ref} can become much different and can increase up to 30 MHz and higher. This can violate the assumption of frequency independence of some component values. Later on, however, we will see that this methodology delivers appropriate results.

Measurement of *Q* of Inductors and Tuned Circuits

Equations 1 and 2 show that Q depends on identifying the loss resistance, R_{ESR}, at a given frequency. The example circuit when we first stated those Equations was for components in series. In that case, the equivalent series resistance (ESR) for each component will add to create a new ESR for the circuit. In the case of a parallel tuned circuit, the total circuit ESR is the parallel combination of ESR from each component. The capacitor ESR will be quiet low, so the resulting circuit ESR is mainly made up by the inductor ESR. This Q is called unloaded Q, because there is no relevant (loading) connection to the tuned circuit that will attenuate the circuit's energy.

The net result is that we need to identify the ESR at a given frequency to calculate Q. There are many methods to do this job and most of them are covered in the literature. See, for example, the HP/Agilent *Impedance Measuring Handbook*.[2] To evaluate a selection towards implementation for a small, low cost device, the following example will be used.

We want to evaluate the Q of a 10 µH inductor at 5 MHz. Let's assume this inductor has a Q of 150 at the given frequency. If this inductor is part of a parallel tuned circuit let's also assume that the corresponding capacitor (100 pF) has a very high Q. So all measurement methods, standalone or within a tuned circuit, go back to measure the ESR of the inductor. Using Equation 1 leads to an ESR value of 2.1 Ω.

The expectation is now to measure a 10% change of Q, which means increasing Q to 165 and decreasing the ESR to 1.9 Ω. Each of the following methods should be evaluated towards this change, if it is practical to measure any change of voltage, phase, frequency, time, or whatever is used in a method to show this change of Q with a low cost and small device design. However, not all methods will be tested as they do not fit the small, low cost device requirements.

Methodologies That Require Tuning of Components

In the HP/Agilent *Impedance Measuring Handbook* and many sources in the literature and on the Internet there are descriptions of methods that require tuning of components like capacitors or resistors to either move to resonance (Resonance Method) or to compare the unknown inductor to reference components (Bridge Method). These methods will not be used because we don't want to manually or electromechanically tune any component.

Methodologies That Require Tuning of Frequency or a Separate Oscillator

With the Resonance Method it is possible to stay with fixed components and tune the frequency instead to arrive at resonance. Frequency tuning is basically also needed with more sophisticated methods like the IV Method, RF-IV Method, Network Analysis Method and Auto Balanced Bridge Method, which are all described in detail in the HP/Agilent *Impedance Measuring Handbook*

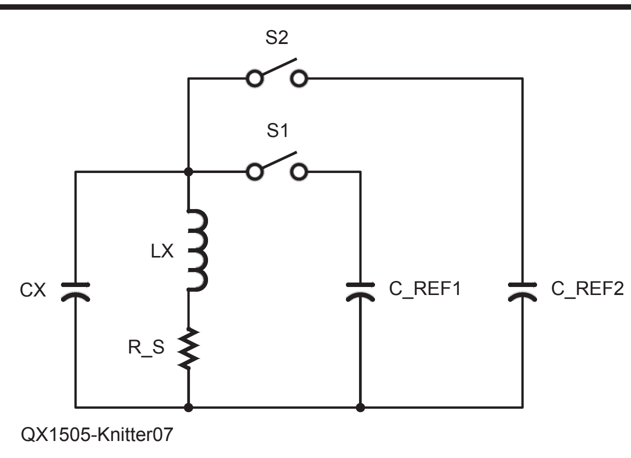


Figure 7 — Measuring *Q* by frequency deviation.

(see Note 2). They all take a separate (tunable) oscillator to inject some current through the device under test, to measure voltage and/or current.

A very practical method is described in *Experimental Methods in RF Design*, in section 7.9, Q Measurement of LC Resonators.[3] I used the parallel tuned circuit trap method later in my design work to confirm Q values for comparison of accuracy with the LCQ-Meter.

These methods do not all fulfill the low cost requirement, however. The frequency generator (DDS / SI507 / PLL) and other required parts can be quite expensive.

Measuring Q With Resonant Frequency Deviation

Most Amateur Radio operators know the equation to calculate the resonant frequency of a tuned circuit as shown in Equation 12, but this is only valid for the assumption of lossless components.

$$f_r = \frac{1}{2\pi\sqrt{LC}} = \frac{1}{2\pi}\sqrt{\frac{1}{LC}} \qquad \text{[Eq 12]}$$

If we introduce loss by including the equivalent series resistance of the components, R_{ERS}, the equation shows a slightly lower resonant frequency.

$$f_r = \frac{1}{2\pi\sqrt{LC}} = \frac{1}{2\pi}\sqrt{\frac{1}{LC} - \frac{{R_{ERS}}^2}{L^2}} \qquad \text{[Eq 13]}$$

Equation 13 would allow us to calculate R_{ERS}, and therefore Q, with Equation 1 if all other component values and also the new, lower resonant frequency can be measured.

Figure 7 shows a circuit with two well-known reference capacitors. We can calculate R_{ERS} based on three resonant frequencies obtained by switching the reference capacitors in and out of the circuit.

The mathematics behind finding C, L and also R_{ERS} is not shown here, but it is possible to quickly evaluate the frequency change from our example using Equation 13.

Without loss, f_r is 5032921 Hz from Equation 12 for the 10 µH and 100 pF components. With the additional loss of 2.1 Ω ESR included in Equation 13 leads to an f_r of 5032810 Hz. The difference is just 111 Hz. Now we know why Equation 12 does work for most cases. If Q is increased by 10% — which means reducing R_{ERS} to 1.9 Ω the new resonant frequency from Equation 13 is calculated as f_r of 5032830 Hz. This is just a 20 Hz difference for a 10% Q change!

This method can be implemented fast, and a microcontroller can measure such small frequency differences with high efficiency. This is especially true if the measurement time is extended to several seconds, but the issue is stability of the oscillator. In a laboratory environment the method worked well, but just having a small breeze of air along the oscillator components changes frequency quite heavily.

A similar approach can be taken by switching on and off known reference resistors in series with R_{ERS}. This will lead to a mathematical equation that has to be solved numerically, but the issue with the small change of frequency will remain, and therefore neither method is used here. I haven't found that type of measurement for Q in the literature anywhere, but in principle it works.

Maybe a clever ham can find a way to build a very stable oscillator — perhaps a crystal based LC oscillator — that allows measuring those small frequency differences on a constant base.

Measuring Q by Switching Tuned Circuit Design

Equation 13 is only valid for parallel tuned circuits. For serial tuned circuits there is no change in frequency introducing loss and therefore Equation 12 is still valid. This fact can be used to calculate R_{ERS} by putting L and C components into a parallel tuned circuit first, and then putting the same components into a serial tuned configuration.

Without showing the mathematics, it goes back to the same small changes in frequency as shown in the previous section. Also, there is no easy design for an oscillator that works with a parallel and serial tuned circuit without introducing changes by different parasitic components. Let's look for another method, which finally made the race.

Measuring *Q* by Analyzing the Amplitude Envelope of a Switched Off Oscillator

All methods mentioned so far work in the steady state of an oscillation. Next we will move to the time domain and measure transient behavior of a switched off oscillator with a parallel tuned circuit as the frequency relevant component.

If a parallel tuned circuit consisting of a capacitor and inductor is part of an oscillator design there will be a steady oscillation at the resonant frequency with a specific amplitude. Figure 8 shows an oscilloscope pattern from the current LCQ-Meter. The sinusoidal curve is measured across the tuned circuit. It stays at a constant amplitude until time zero on the x axis.

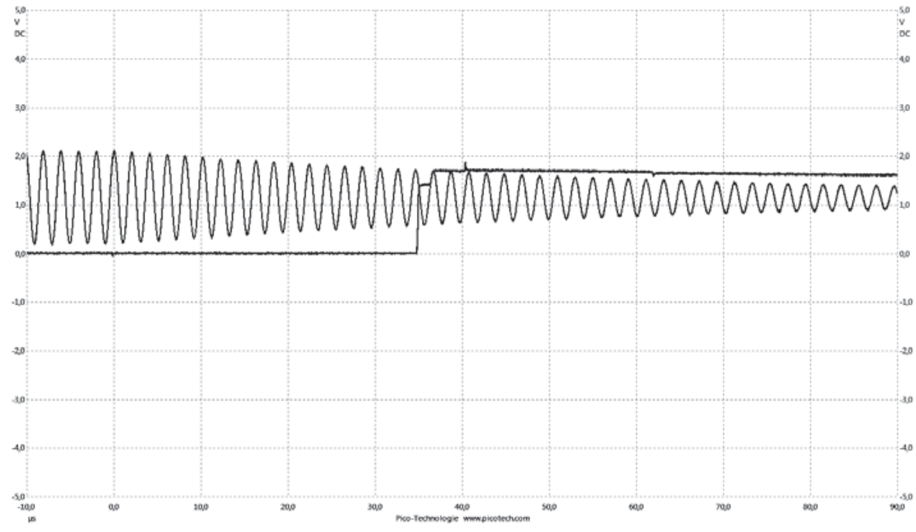


Figure 8 — Amplitude versus time of a tuned circuit that has been switched off, to measure *Q*.

At time zero the oscillator — the energy injecting part of the circuit — is turned off in a way that the tuned circuit is not relevantly attenuated. From time zero onwards, the diagram shows that the amplitude of the oscillation gets lower and lower because there is loss in the tuned circuit. The time that it takes to go from one level of amplitude to a lower level is directly related to the loss of the tuned circuit.

Explained in a different way, if we can measure the amplitude at two different times, and know the resonant frequency of the circuit, we can calculate Q.

Equation 14 gives the voltage across the tuned circuit.

$$u(t) = U_0 e^{-\frac{t}{\tau}} \cos\left(2\pi f_r t\right) \qquad \text{[Eq 14]}$$

where:

$\tau = 2L / R_{ERS}$

The formula contains the swinging part at the resonant frequency, f_r, with the cosine term, the maximum amplitude as U_0, and the envelope describing part with the exponential function. All of this assumes that loss is concentrated in the inductor ESR resistance, R_{ERS}. Obviously the envelope level at a given time is depending on R_{ERS} and therefore on Q. This makes practical sense, because we would expect the swing to die faster if there is more loss.

Let's assume we can measure the envelope with some kind of peak detector at any time. Therefore we can skip the cosine term. The easiest approach to arrive at the desired R_{ERS} is to measure the envelope at time zero, which is nothing the amplitude U_0, and measure the envelope at a second time, t_1.

$$u_e(t_0) = U_0 \quad \text{[Eq 15]}$$

$$u_e(t_1) = U_0\, e^{-\frac{t_1}{\tau}} \quad \text{[Eq 16]}$$

Dividing Equation 16 by Equation 15, and solving for R_{ERS} leads to Equation 17.

$$R_s = 2L\,\frac{\ln\left(\frac{u(-t_1)}{u(t_0)}\right)}{t_1} \quad \text{[Eq 17]}$$

The last step is to use Equation 1 with Equation 17 to solve for Q.

$$Q = \pi f_r \frac{t_1}{\ln\left(\frac{u(-t_1)}{u(t_0)}\right)} \quad \text{[Eq 18]}$$

The amazing thing is that Q does not depend on any values of the components. These values come in by the resonant frequency, which can very easily be measured by a microcontroller. The challenge is to measure the envelope voltage at a given time. Again, a microcontroller is the ideal way doing dedicated actions at dedicated times. The only remaining challenge has to do with the envelope detector, but there will be a solution to that problem, too. Figure 8 includes a real measurement of the envelope by a peak detector (the nearly flat line).

A very important fact in regards to Q measurement is that this method gets more accurate as Q gets higher, because the decay time will increase with higher Q. This is a big advantage compared to other methods, which become less accurate with higher Q!

Another question is what resolution for voltage measurement do we need to find small changes in Q like the 10% increase from our earlier example.

Taking our tuned circuit made of the 100 pF capacitor, 10 µH inductor, and 2.1 Ω resistor with an amplitude of 1 V at time zero, Equation 16 predicts an envelope voltage of 0.349 V after 10 microseconds. If Q is increased by 10%, which relates to a new ESR resistance of 1.9 Ω, the voltage will be 0.387 V. This is an 11% voltage change within a resolution that can easily be measured by a 10 bit ADC included in a microcontroller.

The net result is that even if this Q measurement method looks a little bit strange at first, it is the best fit for acceptable accuracy in a small, low-cost device.

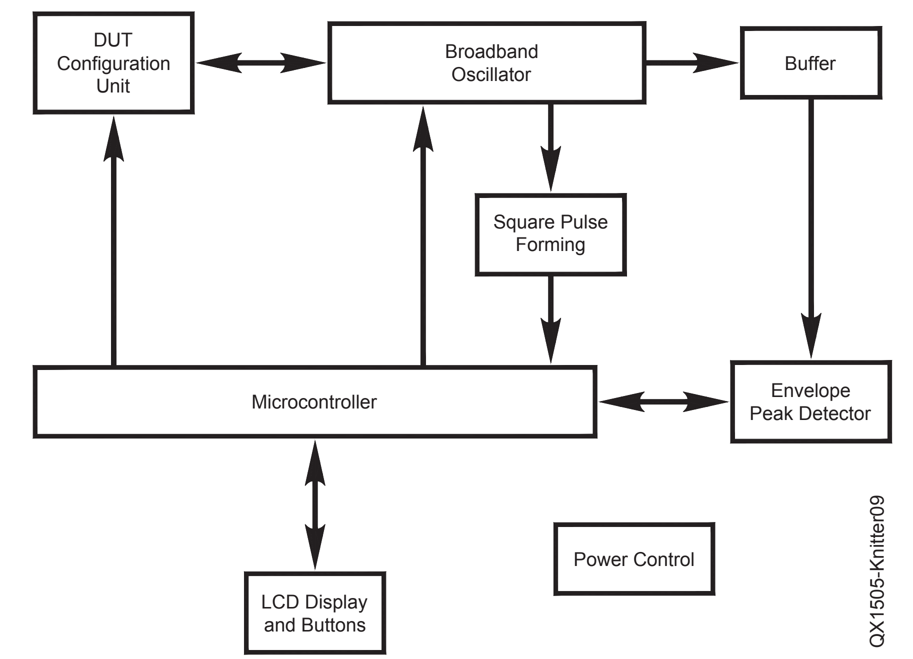


Figure 9 — Block diagram of the LCQ-Meter.

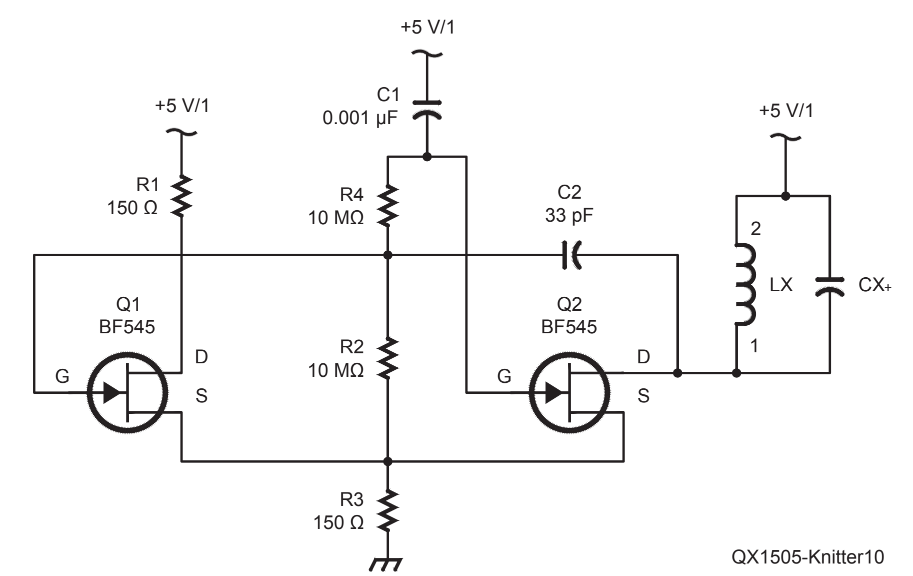


Figure 10 — Broadband amplitude stabilized oscillator.

The LCQ-Meter — A Microcontroller Based Implementation

After all that theory, let's see how this is actually implemented in the LCQ-Meter. Figure 9 shows a block diagram of the meter.

A modern affordable microcontroller from Microchip, the 16F1788 takes over all the control, display and input processing. The Device Under Test Configuration Unit is a relay switched network that allows the unit to combine internal fixed and reference capacitors, an internal inductor and the DUT L, C or LC tuned circuit in different serial and parallel configurations for the different measurement modes. Finally in each configuration the result is an LC parallel tuned circuit that acts as the frequency describing element for the Broadband Oscillator.

The Broadband Oscillator can be switched on and off by the microcontroller. There is also a capability to kick the oscillator to ensure that it starts. The oscillator is described in detail later.

For frequency measurement (internal and external) a Square Pulse Former converts the signal from the oscillator into a clean square wave. This is needed as the microcontroller can only count clean square pulses reliably.

For *Q* measurement, the oscillator will be periodically switched off. To attenuate the tuned circuit the least amount possible during envelope measurement, a high efficiency buffer is placed before the Envelope Peak Detector. The Envelope Peak Detector can be switched on and off by the microcontroller to allow measurement of the envelope at a specific time.

There is also a little bit of effort needed for voltage regulation, bypassing, and so on. These tasks are handled by the Power Control Block. In the current version, the LCQ-Meter can be run by three different power sources, including 2 AA size batteries to make it a portable unit.

Broadband Amplitude Stabilized Oscillator

The Broadband Amplitude Stabilized Oscillator and the Peak Detector are the most challenging components of the overall design. Several requirements have to be met:

1) The oscillator has to cover a frequency range over multiple decades without component change. Measuring tuned circuits with inductors of a few millihenrys and capacitors of a few nanofarads means oscillation at as low as 10 kHz, while measuring tuned circuits with components in the microhenry and picofarad range means oscillation at 10 MHz or higher.

2) The oscillator has to oscillate with relatively high, constant amplitude to allow envelope measurement without any needed additional amplification or gain control.

3) The oscillator has to allow a parallel tuned circuit to be the only frequency relevant element.

4) The oscillator must be capable of being switched on and off without attenuating the tuned circuit element after being switched off. Also, no relevant extra current can be introduced when switching it off, to avoid spikes.

5) The oscillator has to be safe from electrostatic discharge. Attaching a capacitor, inductor or a tuned circuit as the device under test should not kill the circuit easily.

This all sounds like an HF engineer's dream, but it can be built for a few dollars!

Figure 10 shows the oscillator design, which goes back to the clever design presented in *Experimental Methods in RF Design*. See Note 3.

Basically, this is a two stage Franklin oscillator made with good available BF545A N-JFETs. Transistor Q1, on the left, works in common drain configuration, and Q2, on the right works in common gate configuration.

To understand how it works let's leave the two 10 MΩ resistors, R2 and R4, away for a moment and think about the unconnected Gate of Q1.

A signal on this gate will be amplified using an extremely low current as the amplifying unit is an FET. This NFET is a depletion type, so relative lower voltage will decrease the source drain current and, therefore, the source will move more to ground. This move is the input for Q2, which is in a common base configuration because C1 grounds the base for AC current. Lower source potential on Q2 means higher gate potential on Q2 and therefore the Q2 source drain current is increased. This higher current along L_X and C_X moves the drain of Q2 more to ground. The result is a two stage non-inverting amplifier with extremely high input and low output resistance.

Now C2 feeds back the output of Q2 to the gate of Q1. Oscillation will start, but will not run into nonlinear behavior because of the special characteristic of J-NFETs.

Back to the two resistors R4 and R2. C2 is DC connected to 5 V through L_X. With R2, it forms a low pass filter that creates a DC control voltage arriving from the amplitude oscillation at the source connection. This control voltage is also applied to Q2 by R4. This is the amplitude stabilizing part of the oscillator.

Also, the device under test is connected to the drain of Q2, which makes it more reliable than other designs where the LC circuit is attached to a gate of FETs.

Finally, if the Q2 source is disconnected from ground the drain will become a very high resistance to any other components. C2 is connected to other components by two 10 MΩ resistors and by the Q1 gate. The Q1 gate also has a very high resistance and low capacitance. The net is by disconnecting the source (or both sources) from ground it is possible to "free up" the LC tuned circuit of L_X and C_X without leaving a relevant resistance in parallel to attenuate it.

In summary, this design meets all our requirements for the oscillator at very low cost. In the current LCQ-Meter design, Version 2.1, the oscillator works against a negative supply to measure L_X and C_X against ground. See Figure 14 for the full circuit diagram.

Super Fast Comparator Based Peak Detector

The peak detector for envelope measurement is the second challenging part of the overall design. There are four important requirements for this part of the circuit.

1) The peak detector has to ramp up to the input voltage very fast. Testing Equations 17 and 18 with some *Q* values shows that measurements have to be done in a few microseconds. That means the ramp up time must be less than 1 µs.

2) The peak detector has to be capable of being switched on and off to allow measurement at a specific time.

3) The peak detector has to store the momentary voltage long enough to be evaluated by the microcontroller, which will take several microseconds.

4) The peak detector has to make this measurement with suitable accuracy and stability.

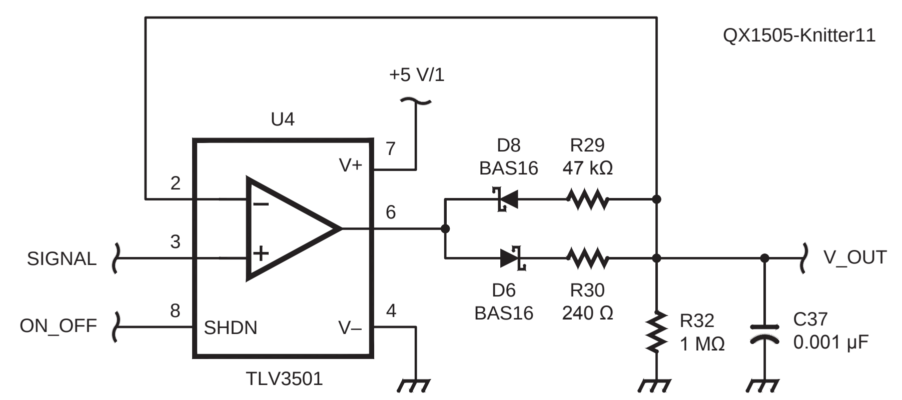


Figure 11 — Comparator based peak detector.

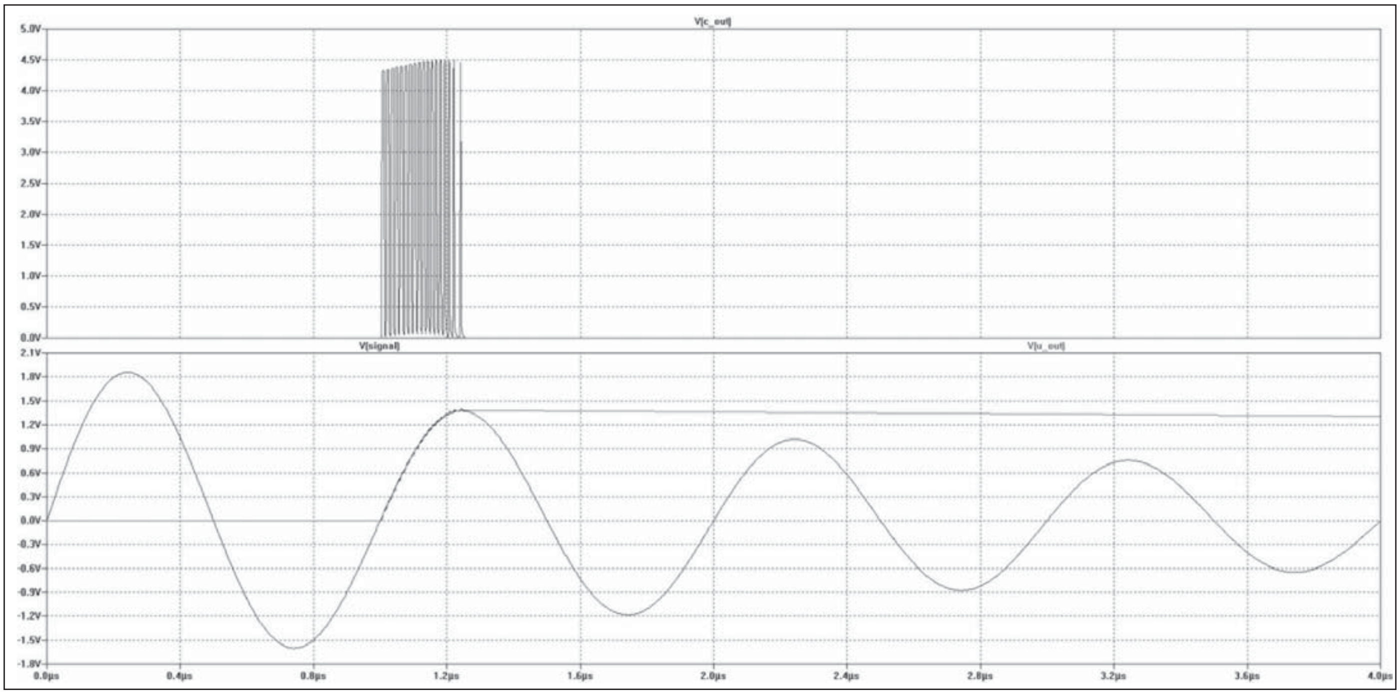


Figure 12 — *LTspice* simulation of peak detector with a 1 MHz signal.

After a lot of research in literature and on the Internet I did not find any existing design that will fit to those requirements. Most designs rely on operational amplifiers, and some on comparators, but all are much too slow. The fastest design achieved a settling time in milliseconds. It can be done, however.

Figure 11 shows a "charge pump" peak detector design based on the ultrafast 4.5 ns rail-to-rail comparator, TLV3501 from Texas Instruments.

This comparator has the big advantage of providing a shutdown input, which allows the comparator to switch on and off within 100 ns. Furthermore, it is a modern rail-to-rail type, which allows measuring from ground up to the power supply level.

Here is how it works. Basically any positive voltage difference above a few millivolts between the positive and negative input of the comparator will result in a positive output (close to 5 V) on pin 6. A current through the fast diode D6 and R30 will quickly charge the holding capacitor, C37.

Because the negative input of the comparator is connected to that capacitor voltage level (closed loop) the charge will stop when the voltage level at C37 reaches the positive input level (the signal level). If that input level is reached, the comparator will toggle and the output on pin 6 will become negative (ground level). This will now discharge the holding capacitor slowly via R29 and D8.

If the signal level still increases or again increases another fast charge cycle will occur. If the signal level decreases, C37 will slowly be discharged again via R29 and D8. The net result is that the voltage on C37 will follow the positive signal peak level based on ultra fast charge cycles and much slower discharge cycles.

You may ask why D8 and R29 are needed at all, or why is a (slower) discharge needed? This is because the comparator will overshoot due to switching delay time and needed hysteresis to prevent wild oscillation. In a real world situation, the output voltage will go above the signal level and will stay there if no discharge takes place. This is also the reason why R30 cannot go much lower. R30 together with C37 form a low pass filter, which mainly prevents that overshoot.

Only if we very carefully select R30 and C37, can we omit the discharge path of R29 and D8. In the current LCQ-Meter design (V2.1) this is the case, but pads for D8 and R29 are still on the circuit board to allow further experimentation. Charge and discharge cycles can be widely adjusted by choosing different values for R30 and R29.

R32 is needed to discharge C37 after a measurement, when the comparator is shut down, which means the output goes to high resistance. Please note not to use Schottky diodes in this design, because their backward capacity is much higher, which results in slower switching times and more overshoot.

Looking back at Figure 8, the straight, flat line shows the voltage level across C37 after switching on the comparator.

The current design, with R30 at 240 Ω allows a ramp up time of less than 500 ns. Someone could argue to reduce C37 to a lower value, to achieve faster ramp up times, but this is not possible because the microcontroller charges its internal hold capacitor for the ADC converter via a relatively high resistance. See the data sheet for the 16F1788. Also, please remember that C37 is part of a low pass filter.

Figure 12 shows an *LTspice* plot for this speed optimized design (as shown in Figure 11) with an input signal at 1 MHz (damped sine wave in the lower section of the plot), the voltage across the charging capacitor C37 (the line that is a zero and then follows the rising second peak of the sine wave, and then holds the peak value) and also the output voltage on the comparator pin (the trace in the upper section of the plot).

The peak detector follows the signal until maximum, which is the envelope at that time. If the peak voltage is measured not too far away from the recent peak, the digitized voltage will quiet closely represent the peak voltage of the envelope at that time. A closer look to the comparator output voltage confirms that it is a series of very fast charge cycles.

The same plot with an input signal at 10 MHz is shown in Figure 13. You can clearly see that the peak detector charges the holding capacitor during the maxima of the input signal towards the envelope voltage. It takes about 500 ns to charge the capacitor to the peak envelope voltage. This rise time has to be incorporated into the calculation of the envelope voltage at a defined time.

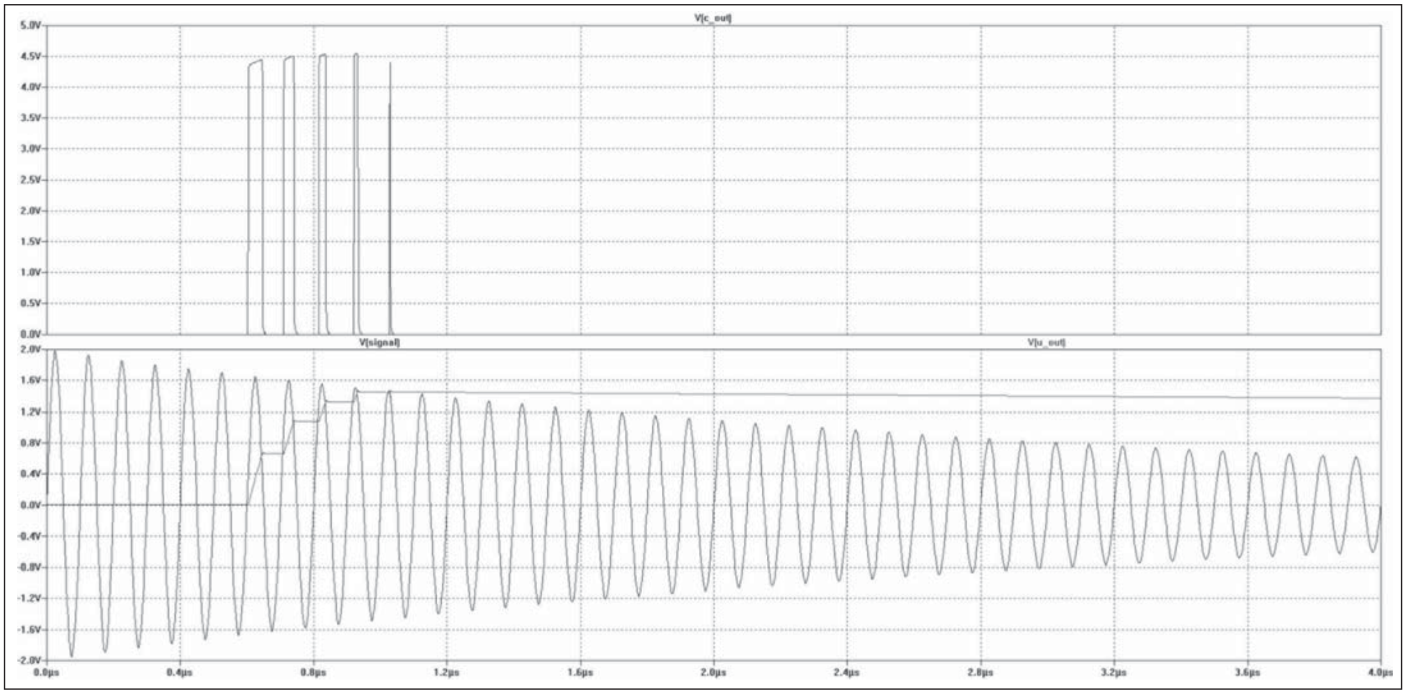


Figure 13 — *LTSpice* simulation of peak detector with a 10 MHz signal.

Putting It All Together

After discussing the details on the key components there are not many secrets left on the rest of the design. Figure 14 shows the complete circuit diagram.

The top section of Part C of the schematic shows the DUT Configuration Unit. It looks complex, but essentially it is about switching each needed tuned circuit configuration with three relays. Analog switches cannot be used here as they have too much influence on the measurement in terms of residual resistance and stray capacitance. If you don't need to measure L and C, but just tuned circuits, you don't need all of the relays. You just need one relay to switch the 100 pF reference capacitor on and off. The relays are controlled by the microcontroller using the three transistors (Q3, Q4, and Q5) at the bottom of Figure 14C.

At the top of Figure 14A, there are some components to allow different supply voltage inputs. The supply voltage can be 5 V through a mini USB connector, 9 V from a battery or DC supply through J1, an unregulated input of 6 to 16 V via connector JP1, 2 to 3 V (2 cells) from batteries via connector JP2.

In practice I have found that the USB supply voltage sometimes contains quite a high noise level. Therefore, all sources are converted to a clean 5 V by the TPS61222 boost converter. In most cases, however, and especially if a battery supply is not needed, the whole circuit can be driven by regulated 5 V directly, which will save some components.

The TPS60403 charge pump (see Figure 14B) creates a clean negative 5 V supply for the oscillator. This is needed as the device under test measurement should be against ground and there are no comparable P-FET types to the N-FET BF545A easily available.

The voltage supply to all key components of the circuit is decoupled by using ferrite beads and bypass capacitors to avoid any crosstalk through the supply lines.

The bottom part of Figure 14A shows the microcontroller and the LCD. Figure 14B shows the user buttons. I used a cheap 16 letters by 2 lines liquid crystal display, but I used high quality push-button switches for better handling and reliability.

There is an option to use an alternative, high quality 16 letters by 2 lines LCD from Electronic Assembly. Along with the same or alternative push-button switches (Schurter), the bottom of the circuit board can be built to have a very flat design, to allow easy assembly into a case. For details see the bill of materials.

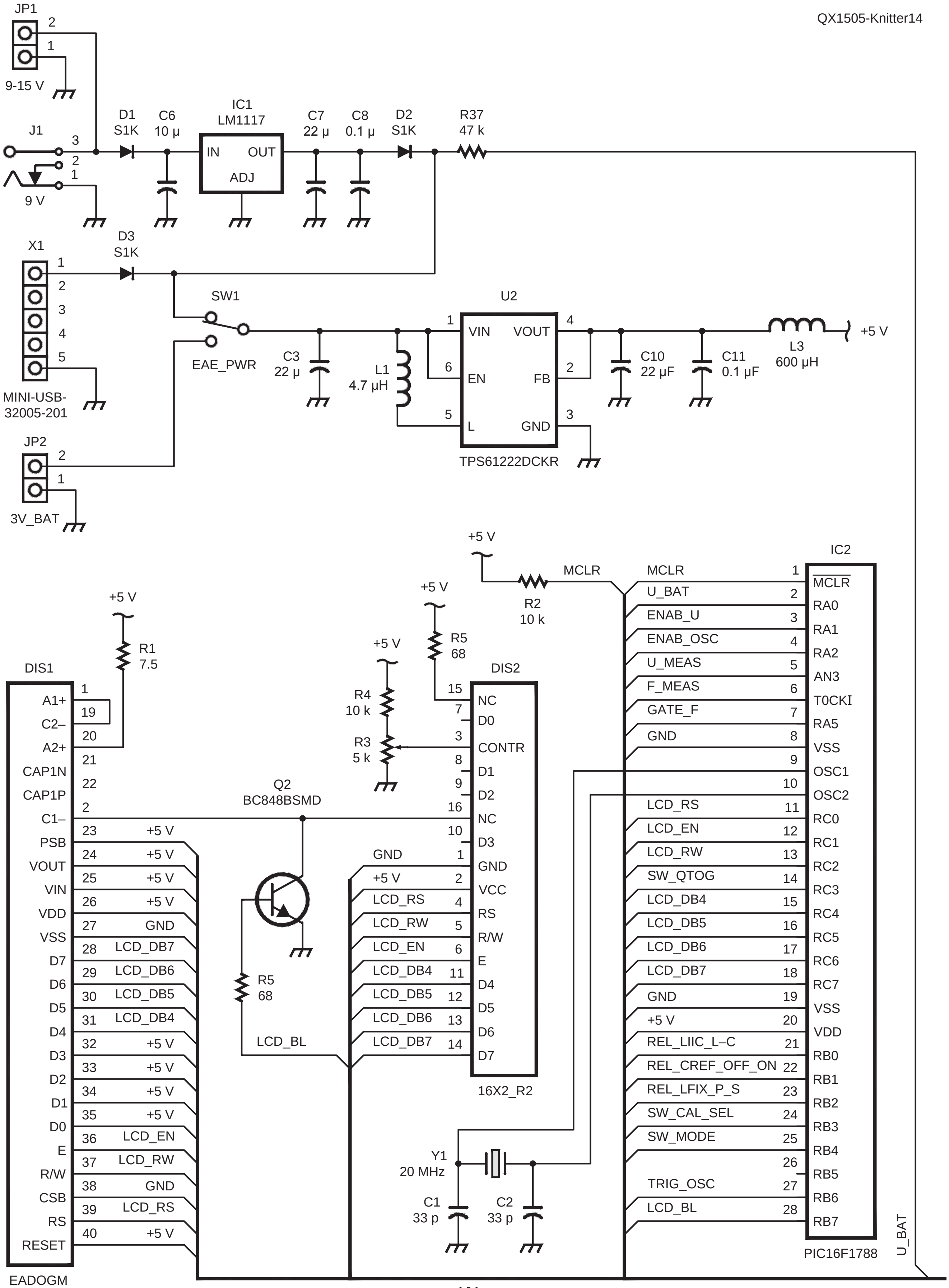
The 16F1788 microcontroller is a low cost, enhanced midrange type from Microchip. It is still an 8-bit microprocessor, but includes large Flash ROM, RAM, EEPROM and some very nice analog components like comparators, operational amplifiers and also an asynchronous prescaler, which allows frequency measurement up to at least 40 MHz.

The middle portion of Figure 14C shows the oscillator, as well as a buffer made with a dual gate MOSFET (Q10). This design is very often used in active probes because the input capacity is less than 2 pF, and the input resistance is extremely high. The buffer is followed by the peak detector (TLV3501 and associated components), which is shown at the top of Figure 14B. Except for the input to the buffer via C36, the full envelope measurement signal chain is DC coupled to minimize the required components.

There is another comparator near the center of Figure 14B, which is used to form clean square pulses for the frequency measurement. The TLV3501 is a little bit oversized here, but because the microcontroller internal comparators cannot reach 30 MHz, I chose to stay with the same comparator component that I used for the peak detector.

The circuit board layout version 2.1, before painting the ground plane, is shown in Figure 15. The size is exactly half of a Eurocard-Size. A double sided board is used, with most components as surface mount devices (SMD). I know that some hams are allergic to SMD, but it is definitely the better solution for such small designs at higher frequencies.

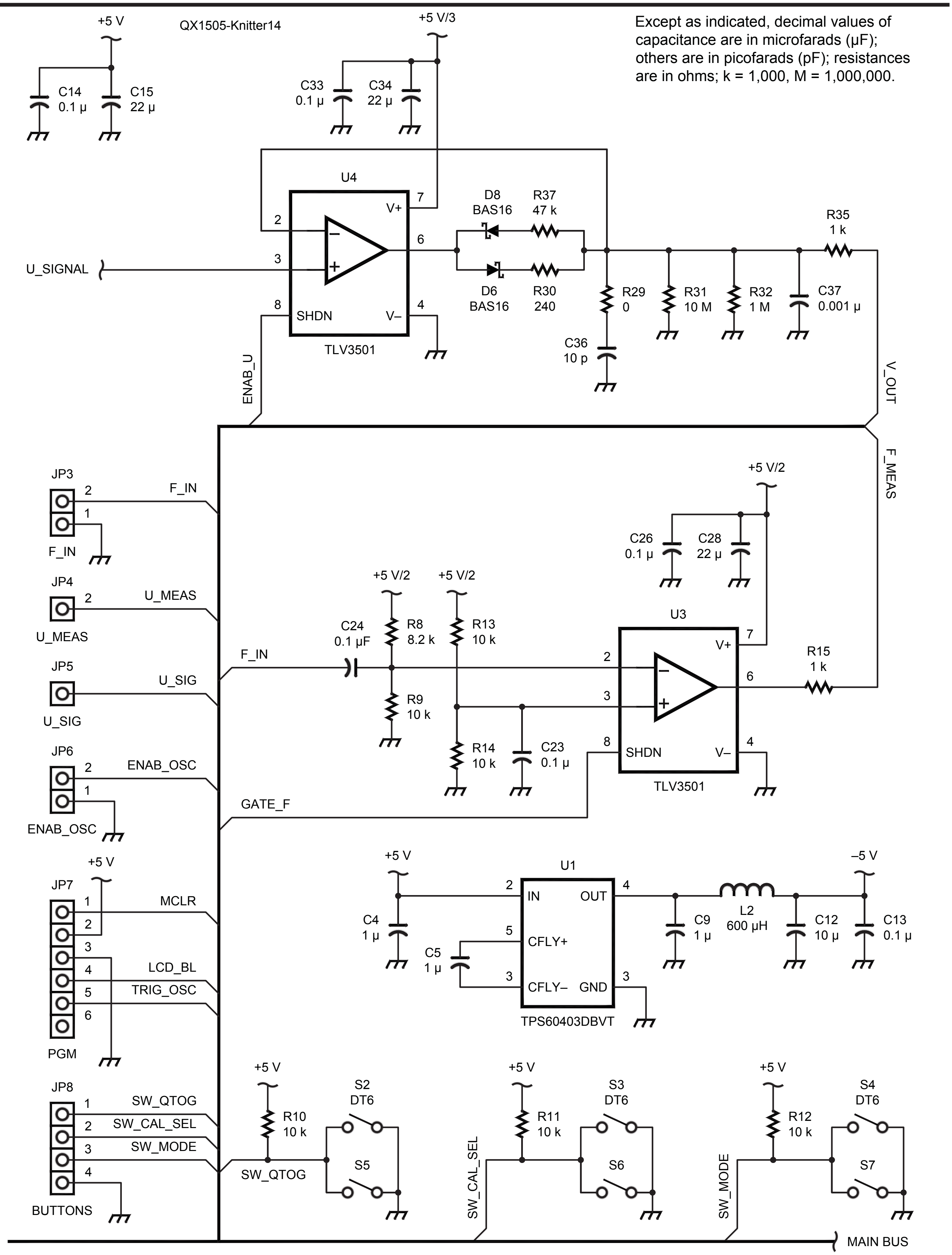
The good news is that all components, except the TLV3501 and some of the power supply components, are also available as through hole devices. So if you would like to try building the LCQ-Meter without using



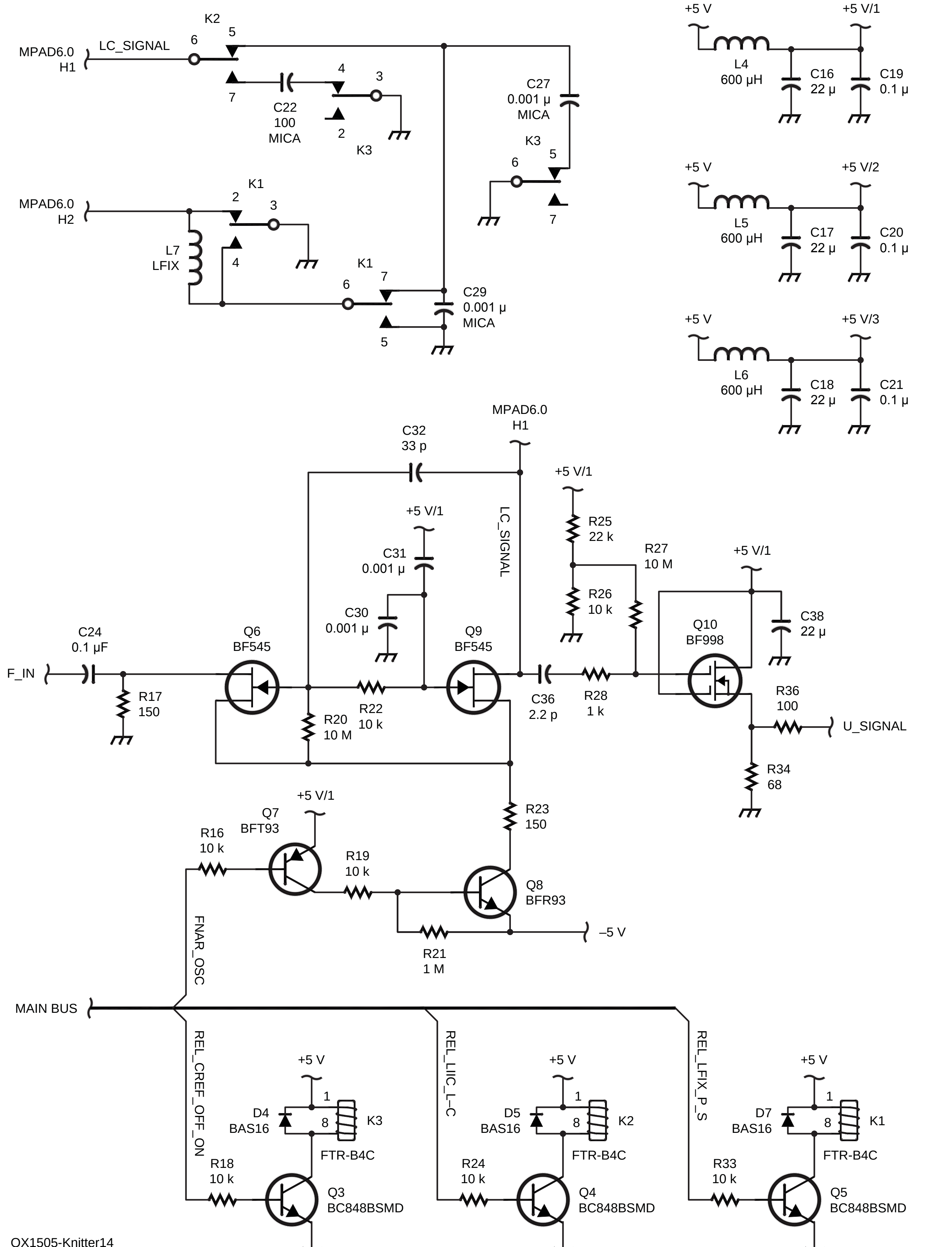
(A)

Figure 14 — This is the complete circuit diagram of the LCQ-Meter.

Except as indicated, decimal values of capacitance are in microfarads (µF); others are in picofarads (pF); resistances are in ohms; k = 1,000, M = 1,000,000.



(B)


MPAD6.0
H1
LC_SIGNAL
K2
6
5
7
C22
100
MICA
4
3
2
K3
C27
0.001 µ
MICA
K3
6
5
7
MPAD6.0
H2
K1
2
3
4
L7
LFIX
K1
6
7
5
C29
0.001 µ
MICA
+5 V
+5 V/1
L4
600 µH
C16
22 µ
C19
0.1 µ
+5 V
+5 V/2
L5
600 µH
C17
22 µ
C20
0.1 µ
+5 V
+5 V/3
L6
600 µH
C18
22 µ
C21
0.1 µ
MPAD6.0
H1
C32
33 p
+5 V/1
LC_SIGNAL
+5 V/1
C31
0.001 µ
C30
0.001 µ
R25
22 k
R27
10 M
R26
10 k
+5 V/1
C38
22 µ
Q10
BF998
C24
0.1 µF
F_IN
R17
150
Q6
BF545
Q9
BF545
R20
10 M
R22
10 k
C36
2.2 p
R28
1 k
R36
100
U_SIGNAL
R34
68
+5 V/1
Q7
BFT93
R16
10 k
R19
10 k
R23
150
Q8
BFR93
R21
1 M
–5 V
FNAR_OSC
MAIN BUS
REL_CREF_OFF_ON
REL_LIIC_L–C
REL_LFIX_P_S
+5 V
1
8
D4
BAS16
K3
FTR-B4C
R18
10 k
Q3
BC848BSMD
+5 V
1
8
D5
BAS16
K2
FTR-B4C
R24
10 k
Q4
BC848BSMD
+5 V
1
8
D7
BAS16
K1
FTR-B4C
R33
10 k
Q5
BC848BSMD
QX1505-Knitter14

(C)

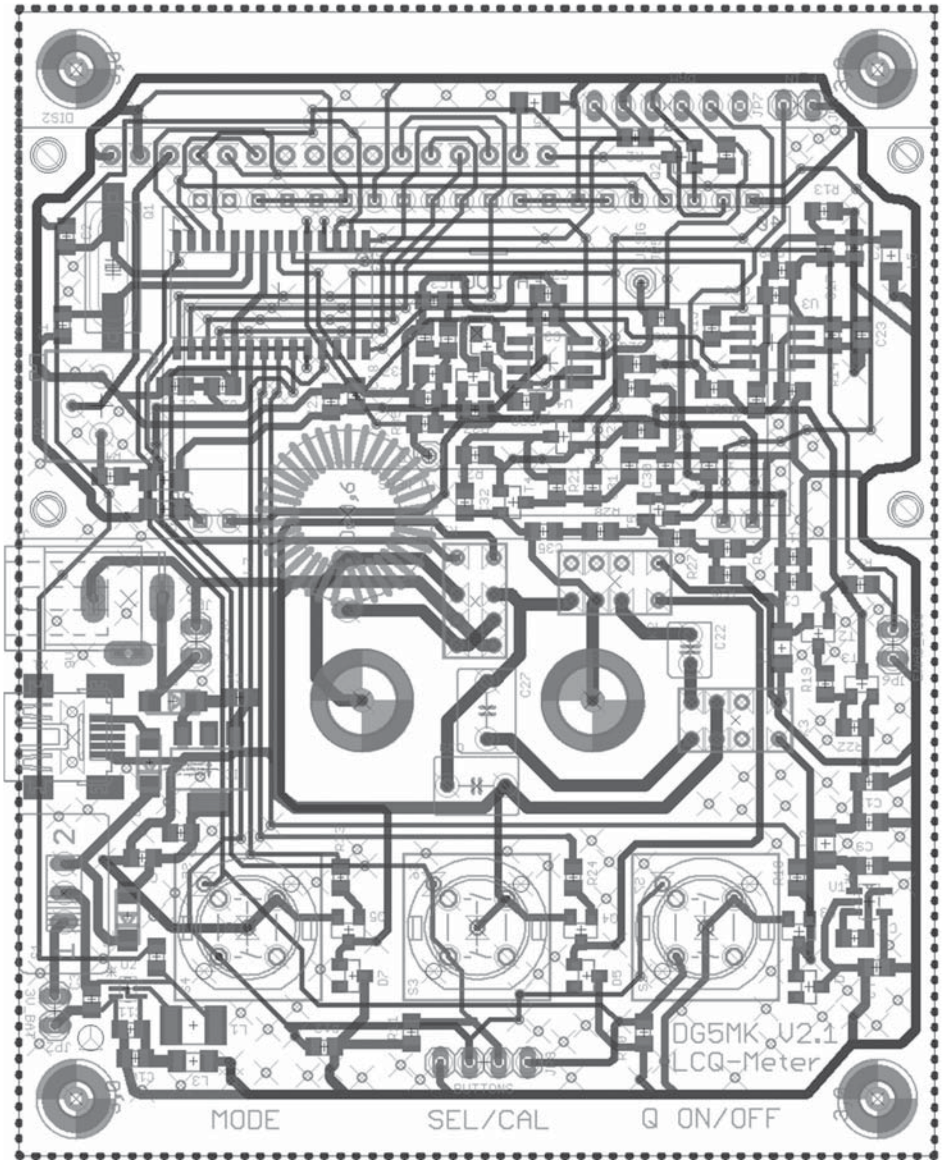


Figure 15 — This is the circuit board layout.

SMD, it can be done. I built a previous prototype with through hole devices, using the full sized Eurocard dimensions. See Figure 16.

No Life Without a Soul — The Software

Now that we know all the theoretical background and how this can be put together in hardware, we need to talk about software for the microcontroller. This is truly a topic of its own, and can become quiet complex because many of the LCQ-Meter functionalities, such as *Q* measurement, are very critical in timing. In the end, it is the software that enables all the functionalities of a complex microcontroller, so it is good to understand how it works.

The following paragraphs will just be an overview of the software in a kind of black box design. In reality the microcontroller is programmed in *C* mixed with assembler routines for critical timing. The current program fills 90% of the relatively large flash memory of the 16F1788, and makes intensive usage of interrupt processing.

As shown in Figure 17, there is a main program, which executes in an endless loop. In this loop, calculation for C, L, *Q* and other parameters, along with the corresponding display on the LCD take place depending on the different operating modes. There is also some functionality that will be invoked if flags are set by interrupts.

For example, when the user presses the calibrate button a flag will be set by the interrupt routine ISR, and the main loop can call needed functions, and reset the flag afterwards. There is more user flag processing to select operating modes, select letters, numbers, and so on.

The interrupt service routine is very lean. Just setting flags and incrementing counters for timing purposes avoids a lot of trouble with saving the context as well as decreased performance.

There are a lot of functions to do any kind of processing. This starts in initialization to set up ports correctly, timers, the prescaler, and the ADC converter, and ends in specific routines to do frequency and *Q* measurement.

Frequency measurement is done on a most accurate base. The microcontroller's prescaler is dynamically reconfigured to get the best resolution depending on the frequency. This methodology is also used for external frequency measurement.

The complex timing for envelope measurement, like oscillator shutdown and waiting for a specific time before starting the voltage measurement, is programmed in assembler because a compiled language like *C* is not fully predictable down to the required microsecond level.

Maybe the most interesting thing is the *Q* measurement cycle. Without going into too much detail here is how it is implemented.

The frequency has already been measured, so the *Q* measurement cycle is dynamically changed depending on the signal frequency. For example, imagine a slow signal at 100 kHz. Just measuring the envelope voltage a few microseconds after the oscillator is switched off will probably lead to incorrect results. It may happen that a decreasing part of a sinusoidal cycle is measured. Therefore, with low frequencies, the measurement time has to be much longer. The opposite is true for very high frequencies.

The cycle itself starts with measuring the base voltage (operating point) and the peak voltage, U_0, of the signal, and continues with a measuring loop. In this loop the oscillator is powered down to start the decrease of the envelope and each time for different time values the voltage, U_t, is measured. If the voltage is less than 50% of the voltage U_0, the cycle is stopped. So the time difference between U_0, and U_t, is dynamically defined by the signal itself. Having an optimal time, *t*, the voltage, U_t, is measured multiple times in a second loop to get an average in order to decrease the influence of noise.

To get a little bit more transparency on this *Q* measurement cycle, an oscilloscope plot taken from the LCQ-Meter is shown in Figure 18. The larger, nearly square wave pulses are the oscillator swings, the trace along with bottom, with spikes, is the voltage measured with the envelope detector. We do not see the decaying envelope because this is a very large time scale. In comparison to Figure 8, this is about the complete cycle.

Each time the oscillator is started it takes a few milliseconds until the amplitude

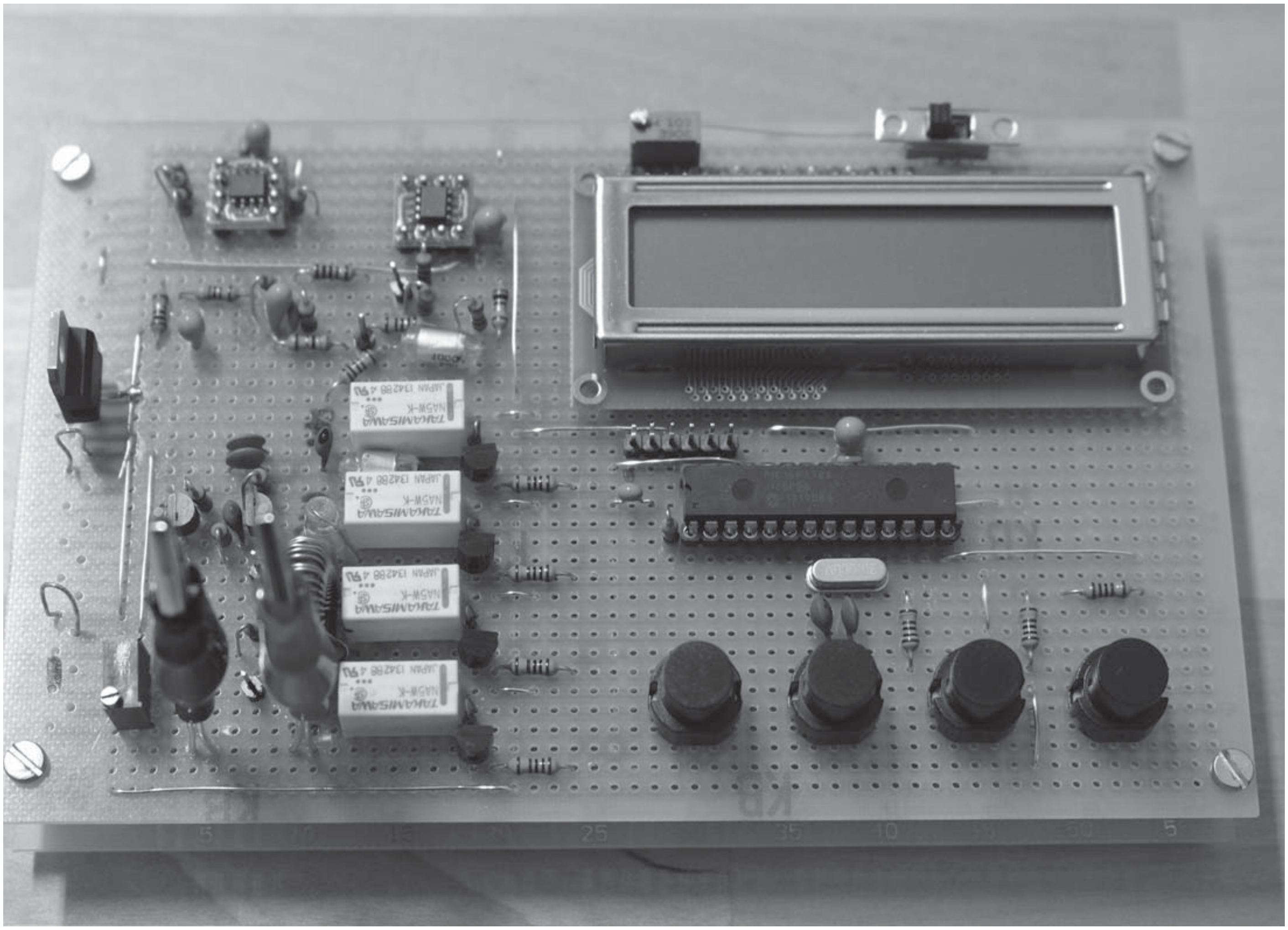


Figure 16 — This is a version of the LCQ-Meter built mostly with through-hole components.

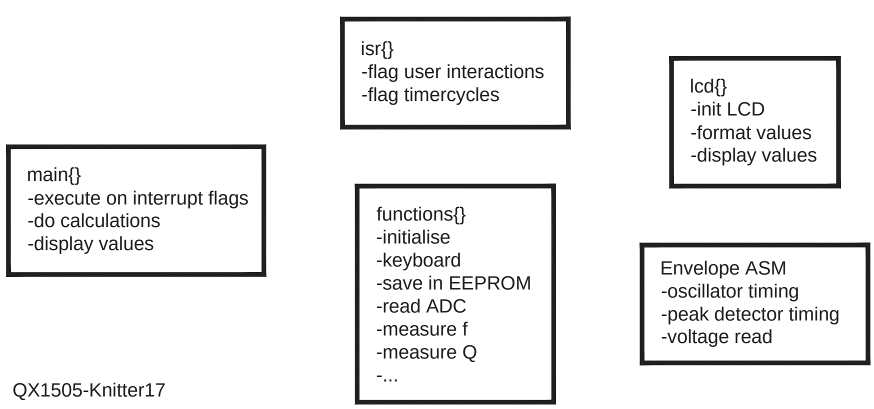


Figure 17 — This diagram shows the various software routines.

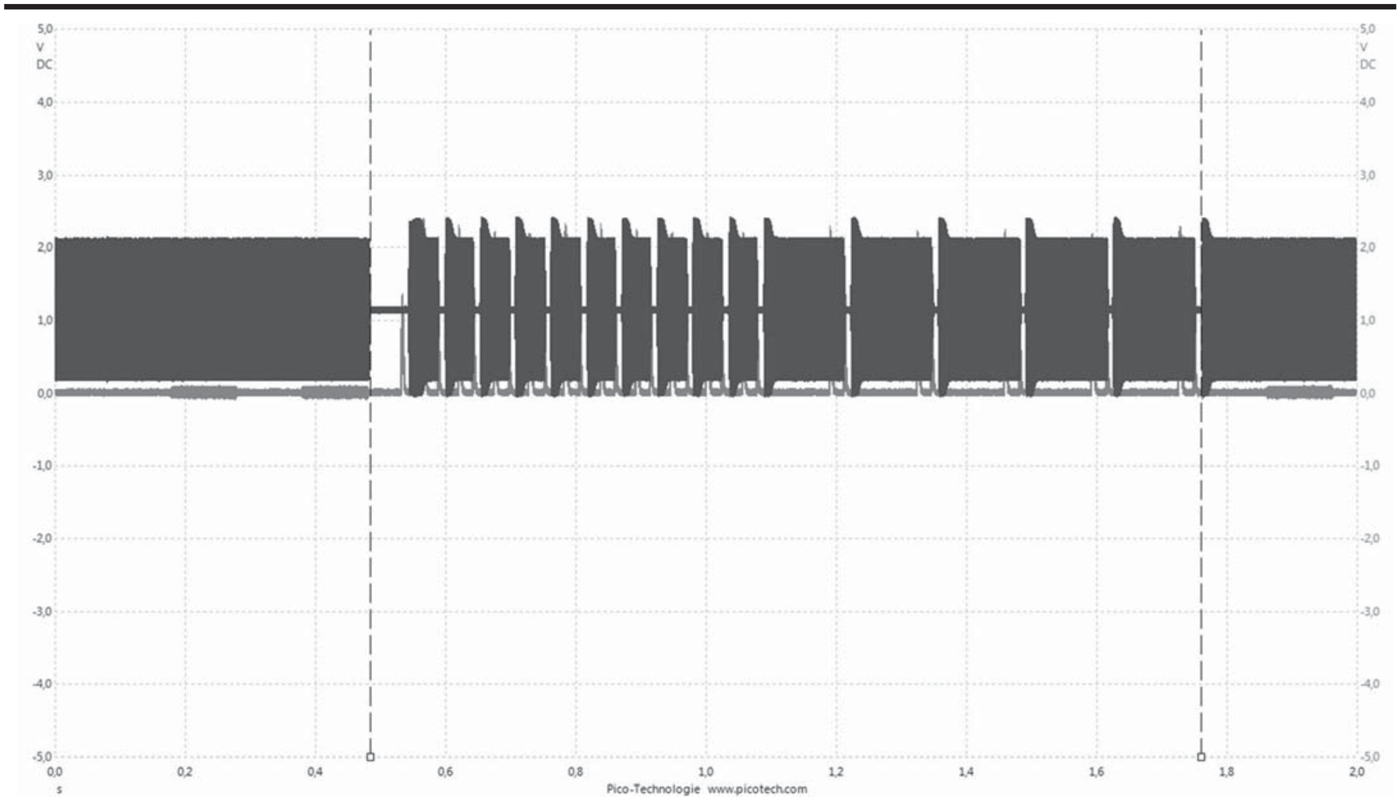


Figure 18 — LCQ-Meter *Q* measurement cycle.

stabilization is done. After that the envelope is stable until the oscillator is shut down.

Starting with the first vertical ruler, frequency is measured shortly before. This is noticeable as the small noise on the bottom line, so there is a little bit crosstalk anyway. The first spike is about measuring the DC operating point. Next are ten small packets, which is finding the right delay time, t_1, to measure U_t. Having that time, t_1, available, five longer packets follow to do a more accurate average measurement of U_{t1}. The next overall cycle starts after the second vertical ruler.

Functionality of The LCQ-Meter

Figure 19 shows the assembly diagram of the LCQ-Meter version 2.1 with controls. Most controls and terminals are self explanatory. The lower three buttons help to describe the functionality of the LCQ-Meter.

The left button is "Mode Selection." It is used to enter a setup menu, but mainly to toggle between four different operating modes.

1) C – Mode: Measure capacitor attached to the clamps; display capacitance and measurement frequency.

2) L – Mode: Measure inductor attached to the clamps; display inductance and measurement frequency.

3) L||C – Mode: Measure parallel tuned circuit; display capacitance, inductance, measurement frequency, and *Q* if enabled by the "Toggle *Q*" button, on the right.

4) F – Mode: Measure frequency on external connector.

In the setup menu it is possible to edit a 16 digit welcome message, such as your call sign, which will be displayed during the power on sequence. The "Select" button will jump from digit to digit while the "Mode Selection" and "Toggle *Q*" buttons allow moving backward and forward through a list of letters and numbers. The setup menu also includes adjusting the reference capacitor

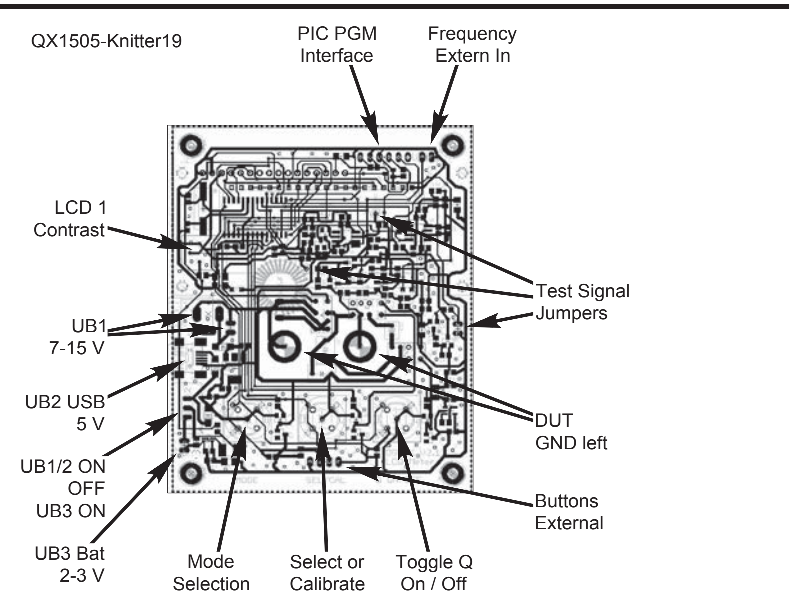


Figure 19 — LCQ-Meter controls.

Table 1
Bill of Materials

Parts	*Qty*	*Value*	*Package*	*Description*
C22	1	100 p	C025-040X050	Capacitor, Mica or Styroflex
C27, C29	2	0.001 µ	C050-050X075	Capacitor, Mica or Styroflex
C35	1	2.2 p	C0805	Capacitor, NPØ
C36	1	10 p	C0805	Capacitor, NPØ
C1, C2, C32	3	33 p	C0805	Capacitor, NPØ
C30, C31, C37	3	0.001 µ	C0805	Capacitor, NPØ
C8, C11, C13, C14, C19, C20, C21, C23, C24, C25, C26, C33	12	0.1 µ	C0805	Capacitor, X5R or X7R
C4, C5, C9	3	1 µ	C0805	Capacitor, X5R, 16 V, Murata, GRM21BR71C105KA01L
C6, C12	2	10 µ	C0805	Capacitor, X5R, 16 V, Murata, GRM21BR61C106KE15L
C3, C7, C10, C15, C16, C17, C18, C28, C34, C38	10	22 µ	C0805	Capacitor, X5R, 6 V, Murata, GRM21BR60J226ME39L
D1, D2, D3	3	S1K	DO214AC	Diode, RS1K or other
D4, D5, D6, D7, D8	5	BAS16	SOT23	Diodes
DIS1	1	EADOGM	EA_DOGM	Electronic Assembly EA DOGM162L-A
DIS2	1	16×2_R2	TUXGR_16X2_R2	LCD display 16x2 characters
H1, H2	2	BIL 20	BIL 20	Connector Hirschmann BIL 20 red, black
J1	1	9V	SPC4077	DC Power Jack HEBW 21
JP4	1	U_MEAS	1×01	PIN Header
JP5	1	U_SIG	1×01	PIN Header
JP2	1	3 V Bat	1×02	PIN Header
JP1	1	9-15 V	1×02	PIN Header
JP6	1	ENAB_OSC	1×02	PIN Header
JP3	1	F_IN	1×02	PIN Header
JP8	1	BUTTONS	1×04	PIN Header
JP7	1	PGM	1×06	PIN Header
K1, K2, K3	3	FTR-B4C	FTR-B4C	Ultraminiature Relay FTR-B4CA 4, 5 V
L2, L3, L4, L5, L6	5	600	C1206	Ferrit Bead Murata 600 Ω at 100 MHz BLM31AJ 601 SN1L
L7	1	LFIX	ED16	Inductor 20 - 100 µH on T50-2 or T50-3
L1	1	4.7 µH	L1812	Inductor, Murata, LQH3NPN4R7MJ0L, 1212 or Fastron 1212FPS-4R7X-01
Q2, Q3, Q4, Q5	4	BC848BSMD	SOT23	NPN Transistor
Q6, Q9	2	BF545A	SOT23	N-FET, A-Type!
Q8	1	BFR93	SOT23-BEC	NPN Transistror
Q7	1	BFT93	SOT23-BEC	PNP Transistror
Q10	1	BF998	SOT143	Dual Gate MOSFET
R29	1	0	R0805	Resistor
R34	1	68	R0805	Resistor
R36	1	100	R0805	Resistor
R17, R23	2	150	R0805	Resistor
R30	1	240	R0805	Resistor
R15, R28, R35	3	1 k	R0805	Resistor
R8	1	8.2 k	R0805	Resistor
R2, R4, R6, R7, R9, R10, R11, R12, R13, R14, R16, R18, R19, R22, R24, R26, R33	17	10 k	R0805	Resistor
R25	1	22 k	R0805	Resistor
R37	1	47 k	R0805	Resistor
R21, R32	2	1 M	R0805	Resistor
R20, R27, R31	3	10 M	R0805	Resistor
R1	1	7.5	R1206	Resistor, ¼ W
R5	1	68	R1206	Resistor, ¼ W
R3	1	5 k	RTRIM64P	Trimmer Resistor 64P
S2, S3, S4	3	DT6	D6R	ITT Switch D6R
S5, S6, S7	3	MTG 1/2	SCHURTER_MTG	Schurter Switch 1241.1032.7
S1	1	EAE_PWR	M251	Sliding Switch APEM SS 25339 N
U3, U4	2	TLV3501	SO-08	TLV3501
IC2	1	PIC16F1788	SO28W	Flash-Based, 8-Bit CMOS Microcontrollers, SOIC
IC1	1	LM1117	SOT223	Low drop voltage regulators 1 A
U2	1	TPS61222DCKR	SOT65P210X110-6N	LOW INPUT VOLTAGE STEP-UP Converter
U1	1	TPS60403DBVT	SOT95P280X145-5N	Unregulated 60 mA Charge pump voltage Inverter
X1	1	MINI-USB-32005-201	32005-201	Mini USB-B Connector Lumberg 2486-01
Y1	1	20 MHz	SM49	Crystal HC49-SMD
Number of components:	130			

Notes
In the current design R29, R31, R37, C36, D8 are not used and should not be assembled (peak detector)
Alternatively use DIS1 or DIS2; Alternatively use S2, S3, S4 or S5, S6, S7; If DIS1 is used R5, R4 and R3 trimmer are not needed

values and the constants for the equation to calculate the parasitic oscillator capacitance, as explained earlier. Again "Mode Selection," "Select," and "Toogle *Q*" are used to increase, decrease and confirm the values. Last option in the setup menu is factory reset, to clear all changes.

In general all changes are stored in the EEPROM and will be kept after power is turned off. This includes the mode selection but not the "Toggle *Q*" selection. This is intentionally set back to off. The best procedure is always to first attach a device under test, and then switch on *Q* measurement. Otherwise there may be confusion about some seconds delay as the software is testing for a device under test that is not there.

The next button is "Select or Calibrate." Selection was explained earlier and is straightforward for the menus. Calibration starts a new calibration cycle in C and L Mode only. It will allow eliminating any additional capacitance and inductance brought in by measurement leads, because calibration is a delta to zero calibration. That means for C Mode the measurement leads have to be open without a device under test before pressing "Calibrate." For L Mode the leads have to be shorted before pressing the button. There is no calibration possible for L‖C Mode.

After each calibration as well as after the power is turned on, the supply voltage will be displayed to warn about a low battery condition.

The last button is "Toggle *Q*" and switches *Q* measurement on and off in L‖C Mode.

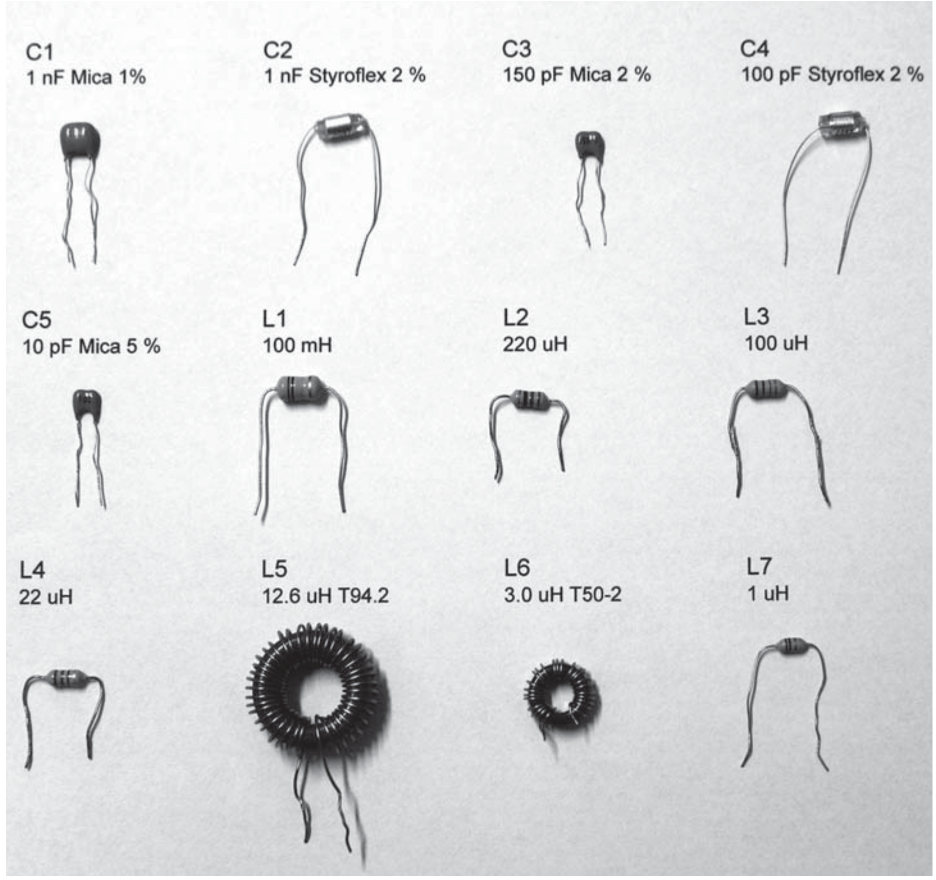


Figure 20 — This photo shows the inductors and capacitors that I used to test the LCQ-Meter.

Practical Measurements

First of all, a disconnected device under test has to be used. You cannot test a component or circuit that is part of a larger circuit. Please do not test loaded capacitors or electrolyte types.

Measurement of capacitance and inductance is done by selecting the right mode, calibrating the meter with leads open (C Mode) or shorted (L Mode) and then attaching the device under test to the clamps.

The LCQ-Meter is capable of measuring capacitance below 1 pF to larger than 0.01 µF. Very large capacitors prevent the oscillator from starting because the internal inductor, L_{fix}, is too small for that. Inductance can be measured below 1 µH to larger than 100 mH.

Measurements on parallel tuned circuits start with selecting the right operating mode and directly attaching the device under test to the clamps. The display will alternate between showing capacitance, inductance, and projected resonant frequency and *Q*. The last one has to be switched on with the "Toggle *Q*" button.

The measurement range is the same as in the other modes but care has to be taken with very small capacitors in the tuned circuit. Below 10 pF the accuracy will decrease because the parasitic capacitance is not perfectly compensated. Also, the frequency will become relatively high. The oscillator will easily cover 50 MHz, but the microcontroller prescaler will stop working somewhere between 40 MHz to 50 MHz. Also, the parasitic capacitance overall will prevent higher frequencies.

The range for *Q* measurement works well towards *Q* of 300 if the capacitance C_X is not too low, and is of a low loss type. If possible, it should be above 100 pF to reduce the influence of the parasitic capacitance, which attenuates the tuned circuit to some extent. With C_X at 1000 pF, *Q* measurements can even go higher to above 500 if you have such high quality components at hand.

Frequency measurement is straightforward. After selecting the F-Mode, the frequency of an external signal will be displayed. The measurement range depends on the prescaler and ends somewhere between 40 MHz to 50 MHz. The signal has to have a peak voltage of about 1 V, depending on R8. The input resistance is around 150 Ω. Also please note that the gate time for the measurement is quiet small, so accuracy is limited. It is just a basic additional feature that will solve many quick measurement needs.

Accuracy of The LCQ-Meter

With the design of a small, low cost device, there has to be some limitation to the measurement accuracy. The LCQ-Meter should still be able to meet the needs of many Amateur Radio operators. With the methodologies and implementation described nobody should expect 0.1% accuracy.

In practice, the goal was to achieve 5% accuracy for capacitance and inductance measurement over a wide range, and 10% to 20% accuracy for *Q* measurement.

Figure 20 shows some of the components that I used for test measurements. The components were measured with a Vector Network Analyzer with and without RF-IV methodology, for reference.

Q for inductors and/or parallel tuned circuits was measured as well with the VNA on different methodologies including the "Notch Method," with serial and parallel resonant traps as described in *Experimental Methods of RF Design* (see Note 3). The last method showed up to be the most accurate for *Q* over a wide range of measurements.

Measuring high *Q* in the range of *Q* > 300 was vulnerable to parasitic resistance and capacitance. Figure 21 shows two different front ends for the Notch measurement. The lower, little circuit board is from an HF test-set for the VNA. Even with careful calibration the highest measurable *Q* for the C1 and L5 combination (0.001 µF Mica with 12.6 µH toroid) was about *Q* = 280 using that board, while the LCQ-Meter showed a *Q* better than 300 all the time.

Doing *Q* calculations by hand on the scoped envelope plot from the LCQ-Meter, it was clear that there is no error within the LCQ-Meter and *Q* has to be higher than 280. eventually, parasitic resistance of the connectors on the test board showed up to be the bad guy. Milliohms matter here! With careful calibration and direct soldering of the device under test to the SMA connectors (the top part of Figure 21) *Q* increased to 306 both with the parallel and serial trap notch method.

Q-measurement becomes even more difficult if very low capacitance values are used, as Table 2 will prove. Table 2 shows all measurement results for L-mode, C-mode and L‖C-mode. It includes also *Q* references from both notch measurement types.

The notch measurement shows differences between both types, up to 60% if only 10 pF is used for the capacitor in the tuned circuit! This shows that measuring *Q* of parallel tuned circuits with very low capacitance could become a challenge in general.

Let's start with capacitance and inductance measurement. For singular capacitance and inductance measurement the LCQ-Meter shows good results within the expected accuracy range. For very large inductors, like L1 (100 mH) shown in Figure 20, it was not possible to create any reference value with the available VNA.

The parallel tuned circuit measurements of capacitance in parallel with inductance show good results as well. Version 2.1 of the circuit board, with much lower parasitic capacitance in the oscillator section pays off here. Even capacitance down to 10 pF can be measured with reasonable accuracy. A little bit more fine tuning on the parasitic capacitance approximation function would probably bring all of the measurements below 5% measurement error. In Amateur Radio practice this small difference in capacitance and inductance will very seldom matter.

On *Q* measurement, for practical combinations of L and C, the measurement results are better than expected if *Q* is high enough. If *Q* is much below 100, the decay of the envelope is too fast to measure correctly,

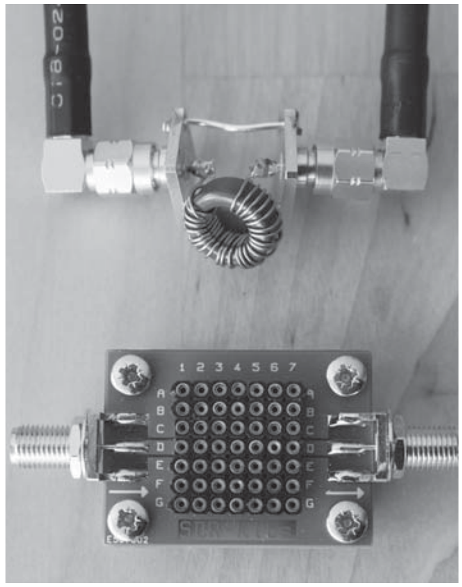


Figure 21 — The bottom photo shows a circuit board that I used to measure the *Q* of a parallel notch filter with a VNA. The results seemed too low, so I built the fixture shown in the top photo, with the components soldered directly to the SMA connectors.

however.

This means *Q* measurement will become more accurate with higher *Q* components. This is a big advantage of the LCQ-Meter compared to other measurement types. My original goal was to measure *Q* up to 10 MHz, but results are fine to some extent in the full HF range up to 30 MHz. Low *Q* and very low capacitance are challenging.

The question is whether those limitations are of any practical relevance for Amateur Radio operators? Who builds a parallel tuned circuit with a 100 µH inductor and a 10 pF capacitor? Well, you never know.

The net result is that if there is a choice for the tuned circuit capacitance (for example if measuring inductors only with a high *Q* capacitor in parallel) the tuned circuit capacitance should be as large as possible.

Another test not shown here is to introduce extra loss by adding resistors. By adding a 1 Ω or 2 Ω resistor, we can show that high *Q* circuits lose *Q* within expected results.

Summary

The LCQ-Meter proves that there is definitely much more functionality possible than currently available in the low cost small device area of amateur measureing instruments. It also demonstrates how powerful current combinations of microcontroller, software and hardware could become even for complex measurement tasks.

All of this is possible in a very affordable way. All of the components used in this project are relatively inexpensive, and have good availability. If someone does not need multiple power supply options or all functionalities, a lot of components could be eliminated.

The LCQ-Meter could become a kit or could even be made available as a ready to run device, but personally I do not have the interest and time to start a business selling components or devices. Amateur Radio is also about sharing. Therefore, I decided to make this project available to the public.

All files including Eagle circuit board files, software and so on are available on an as is base on my website for non-commercial use.[5] These files are also available for download from the ARRL *QEX* files web page, current as of the date of publication.[6]

I want to offer special thanks to DJ6EV and DD1KT for testing some of the prototypes and giving a lot of good hints and recommendations for improvement to the LCQ-Meter. I hope you enjoyed this article and gained some interesting insights and ideas.

Michael Knitter, DG5MK, works in an international computer company in sales and distribution. He earned a degree in telecommunication technology from the University of Dortmund. Besides his professional career in a very different area, he never moved away from electronics and radio communications. Michael has been a licensed Amateur Radio operator since 2006. His special areas of interest are software defined radio, digital signal processing, filter design, magnetic loop antennas, microcontrollers, C++ *and* Labview *programming. He loves to sail, as a true contrast to modern busy life.*

Notes

1. Almost All Digital Electronics L/C Meter, **www.aade.com/lcmeter.htm**
2. HP/Agilent *Impedance Measurement Handbook*, **http://cp.literature.agilent.com/litweb/pdf/5950-3000.pdf**
3. Wes Hayward, W7ZOI, Rick Campbell, KK7B, Bob Larkin, W7PUA, *Experimental Methods in RF Design*, 1st Edition, ARRL, Chapter 7, Section 7.9.
4. Julius Foit, "Broadband Amplitude-Stabilized Oscillators," *Proceedings of the 5th WSEAS International Conference on Microelectronics, Nanoelectronics, Optoelectronics*, Prague, Czech Republic, March 12-14, 2006, pp 1 – 5.
5. Michael Knitter, DG5MK, Web page, **www.dg5mk.de**
6. The files for the LCQ-Meter are available for download from the ARRL QEX files web page. Go to **www.arrl.org/qexfiles** and look for the file **7x15_Knitter.zip**. These files are current as of the publication date of this article, but will not be updated if the author creates new files. For the latest versions, check the author's website. See Note 5.

Table 2

Measurement Results With Hardware Design Version 6.3

Measurement with HW Design 2.1 and SW V 6.3 **2/15/2015**

Changed Peak Detector Design 400 nS (closed loop, BAS 16)

C - Measurement / pF	AADE Clone	LCQ-Meter	RF-IV VNWA	LCQ-Meter to RF-IV VNWA
C1 1nF Mica	1009.0	1010.5	1005.1	0.5%
C2 1nF Styroflex	1001.0	1004.0	998.0	0.6%
C3 150 pF Mica	152.2	153.2	151.5	1.1%
C4 100 pF Styroflex	100.8	101.5	100.7	0.8%
C5 10 pF Mica	10.2	10.3	10.2	1.0%

L - Measurement / uH	AADE Clone	LCQ-Meter	RF-IV VNWA	LCQ-Meter to RF-IV VNWA
L1 100 mH	115900.0	95440.0	no meas.	n/a
L2 220 uH	220.5	218.0	223.0	-2.3%
L3 100 uH	100.7	99.5	101.6	-2.0%
L4 22 uH	22.4	22.1	22.5	-1.9%
L5 12,6 uH	12.4	12.3	12.6	-2.3%
L6 3 uH	3.0	3.1	3.1	1.6%
L7 1 uH	1.1	1.1	1.1	-1.9%

L||C - Measurement	C: LCQ-Meter	L: LCQ-Meter	Q: LCQ-Meter	Frequency KHz	Q: Notch Serial TP	Q: Notch Parallel TP	Notch delta ser to par	C: LCQ-Meter to nominal	L: LCQ-Meter to nominal	Q: LCQ-Meter to Notch ser	Q: LCQ-Meter to Notch par	Q: LCQ-Meter to Notch avg
C1 1nF || L3 100 uH	1008.7	99.8	114	502	111	113	-1.3%	0.9%	-0.2%	2.4%	1.1%	1.7%
C1 1nF || L5 12,6 uH	1007.8	12.3	312	1,429	306	306	-0.1%	0.8%	-2.1%	2.1%	2.1%	2.1%
C1 1nF || L6 3 uH	1003.3	3.1	198	2,838	202	226	-10.3%	0.3%	4.3%	-2.1%	-12.2%	-7.5%
C1 1nF || L7 1 uH	999.7	1.0	9	4,939	41	45	-8.9%	0.0%	4.0%	-78.0%	-80.0%	-79.1%
C4 150pF || L3 100 uH	150.0	99.7	99	1,301	89	100	-10.8%	0.0%	-0.3%	11.4%	-0.7%	5.0%
C4 150pF || L5 12,6 uH	149.1	12.4	309	3,701	306	323	-5.1%	-0.6%	-1.6%	1.0%	-4.2%	-1.7%
C4 150pF || L6 3 uH	147.4	3.2	214	7,335	204	231	-11.8%	-1.7%	6.3%	5.1%	-7.4%	-1.5%
C4 150pF || L7 1 uH	140.6	1.1	82	12,799	56	62	-9.3%	-6.3%	10.0%	45.4%	31.8%	38.3%
C5 10pF || L4 100 uH	9.3	99.7	54	5,220	13	31	-58.2%	-7.0%	-0.3%	321.9%	76.5%	148.8%
C5 10pF || L5 12,6 uH	9.3	12.5	139	14,723	115	194	-40.8%	-7.0%	-0.7%	21.1%	-28.3%	-9.9%
C5 10pF || L6 3 uH	9.8	3.3	90	27,978	79	107	-26.8%	-2.0%	9.7%	14.6%	-16.0%	-3.1%
C5 10pF || L7 1 uH	9.7	1.2	90	46,903	39	47	-18.2%	-3.0%	18.0%	133.2%	90.7%	109.8%

Rudy Severns, N6LF

PO Box 589, Cottage Grove, OR 97424: **n6lf@arrl.net**

Radiation and Ground Loss Resistances In LF, MF and HF Verticals: Part 1

With the impending FCC announcement about the release of a new LF and a new MF band, hams will be interested in practical antennas and learning how to calculate EIRP to legally operate on those bands.

Unlike the higher bands, where the maximum transmitting power limit is stated in terms of transmitter output power, on the (soon to be released) 630 m (472 to 479 kHz) and 2200 m (135.7 to 137.8 kHz) bands, the maximum allowable power is stated in terms of the effective isotropic radiated power (EIRP) from the antenna. On 630 m the maximum EIRP allowed is 5 W, which for the short verticals likely to be used at 475 kHz, translates to a radiated power (P_r) of 1.7 W. (For more information on EIRP, see the sidebar.)

This raises the question, "How do we determine P_r?" As shown in the sidebar, the standard professional approach has been to measure the field strength at a point some distance from the antenna and then calculate EIRP. That's fine for the pros, but for most amateurs, that method won't be practical. There are other ways we might go about it, however. For example, if we can measure the current at the feed point (I_o) and if we know the radiation resistance (R_r) referenced to the feed point, we can find the radiated power from Equation 1.

$$P_r = I_o^2 \times R_r \qquad \text{[Eq 1]}$$

An alternative would be to measure the feed point resistance (R_i) and the input power (P_i) and then calculate P_r using Equation 2.

$$P_r = (R_r / R_i) \times P_i \qquad \text{[Eq 2]}$$

We can measure quantities like I_o, P_i, and R_i, but there is no way to measure R_r directly.

Feed Point Equivalent Circuit Model

Figure 1 shows the traditional equivalent circuit used to represent the resistive part of an antenna's feed point impedance (R_i) when describing what happens to the input power, P_i. The radiation resistance, R_r, represents the radiated power.

$$P_r = I_o^2 \times R_r \qquad \text{[Eq 3]}$$

where:
I_o is the current at the feed point in rms amperes.

The power lost in the soil close to the antenna is represented as R_g. The sum of other ohmic losses such as conductor loss, insulator leakage, and so on is represented as R_L. The input resistance at the feed point is assumed to be the sum of these resistances.

$$R_i = R_r + R_g + R_L \qquad \text{[Eq 4]}$$

Determining P_L is reasonably straightforward, but P_g is trickier. In the following discussion I will be ignoring R_L. In other words, we will assume lossless conductors. This is not because these losses are unimportant but the interest here is in R_r and R_g, and how they vary with frequency, ground system design and soil characteristics. P_L is certainly a worthy subject, but we will save that for another day.

The traditional assumption has been that R_r for a vertical over real ground is the same as it would be for the same antenna over perfect ground. The value we measure for R_i is assumed to be the sum of the R_r for perfect ground and additional loss terms that result from ground and other loss elements. I've certainly gone along with the conventional thinking, but over the years I've become skeptical after seeing experimental and modeling results and calculations that didn't fit. I've come to the conclusion that at HF at least, R_r for a given vertical over real soil, is not the same value for the same antenna over perfect ground.

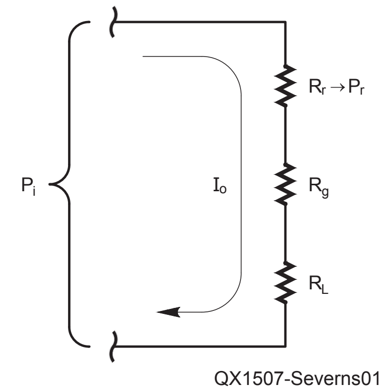


Figure 1 — This is a typical equivalent circuit for an antenna feed point resistance.

The following discussion focuses on the concept illustrated in Figure 1, with R_L = 0. The discussion will show that at HF (1.8 MHz and higher frequencies), R_r

differs significantly from the value over ideal ground. At LF (137 kHz) and MF (472 to 479 kHz), however, the variation of R_r from the ideal value is much smaller, which is very helpful for determining P_r.

To make this article easier to read I've placed almost all the mathematics and the many supporting technical details in an extensive set of Appendices.

Appendix A — Shows how to calculate R_r using the Poynting vector.

Appendix B — Gives a review of soil characteristics.

Appendix C — Describes the E and H fields and power integration.

Appendix D — Covers other miscellaneous bits.

Pushing material into appendices makes life much easier for the casual reader, but provides the gory details for those who want them. These appendices are available on my web site: **www.antennasbyn6lf.com** and are also available for download from the ARRL *QEX* files web page. Go to **www.arrl.org/qexfiles** and look for the file **7x15_Severns.zip**.[1]

R_r For A Lossless Antenna

We need to be careful with our use of the term "radiation resistance." A definition of R_r associated with a lossless antenna in free space, can be found in almost any antenna book. A typical example is given in *Radio Engineers' Handbook* by Frederick Terman:[2]

"*The radiation resistance referred to a certain point in an antenna system is the resistance which, inserted at that point with the assumed current* I_o *flowing, would dissipate the same energy as is actually radiated from the antenna system. Thus:*

$$\text{Radiation resistance} = \frac{\text{radiated power}}{I_o^2}$$

Although this radiation resistance is a purely fictitious quantity, the antenna acts as though such a resistance were present, because the loss of energy by radiation is equivalent to a like amount of energy dissipated in a resistance. It is necessary in defining radiation resistance to refer it to some particular point in the antenna system, since the resistance must be such that the square of the current times radiation resistance will equal the radiated power, and the current will be different at different points in the antenna. This point of reference is ordinarily taken as a current loop, although in the case of a vertical antenna with the lower end grounded, the grounded end is often used as a reference point."

Discussions of R_r for the lossless case are common but I've not seen a discussion of R_r where the effect of near-field losses are considered. In his book, *Antennas*, Kraus does tease us with a comment:[3]

"*The radiation resistance R_r is not associated with any resistance in the antenna proper but is a resistance coupled from the antenna **and its environment** to the antenna terminals.*"

The bold type is mine! The implication that the environment around the antenna plays a role is important but unfortunately Kraus does not seem to have expanded on this observation.

Calculation of R_r and R_g

As pointed out earlier if you know I_o and P_r, you can calculate R_r. A standard way to calculate the total radiated power is to sum

EIRP and Radiated Power, P_r, From Verticals

On 630 m the maximum allowable power is stated in terms of effective isotropic radiated power (EIRP), which is not the same as the radiated power ($P_r = R_r \times I_o^2$, where I_o is the rms current). It is important to understand the difference. As shown in Figure SB1, an isotropic radiator is one that radiates uniformly in all directions. The power density, P_{di}, is the same in all directions at a given radius. If you place a short monopole over a perfect ground plane, for the same P_r, the power density at the same radius will be greater by a factor of 3 (+4.77 dB). The factor of 3 occurs because the power density is doubled (+3 dB) by going from free space to the perfect ground plane, and there is a further increase of 1.5 × (+1.77 dB) because of the directivity of the short monopole.

To achieve the same P_d at the same radius, if we excite the isotropic antenna with $P_r = 5$ W, we can only excite the monopole with $P_r = 1.7$ W.

To determine the power density (P_d) in the wave front, we can make a field strength ($|E_z|$) measurement at some distance r from the antenna.

$$P_d = \frac{|E_z|^2}{377} \approx \frac{|E_z|^2}{120\,\pi}\left[\frac{W}{m^2}\right] \qquad \text{[Eq SB1]}$$

Note, E_z is in V/m and 377 Ω represents the impedance of free space. Implicit in Equation SB1 is the assumption that the measurement of E_z has been taken far enough from the antenna to be in the far field, where $|E_z| / |H_y| \approx 377$ Ω. At 630 m, you need to be at least 5 λ away, or about 3 km, and 5 km would be better.

Assuming P_d is constant over a sphere with radius r (in meters) you can multiply P_d by the area of the sphere to obtain EIRP.

$$EIRP \quad \frac{r\,|E\,|}{60}\left[W\right] \qquad \text{[Eq SB2]}$$

The point is that while we are allowed an EIRP = 5 W, the allowed P_r is about 1.7 W!

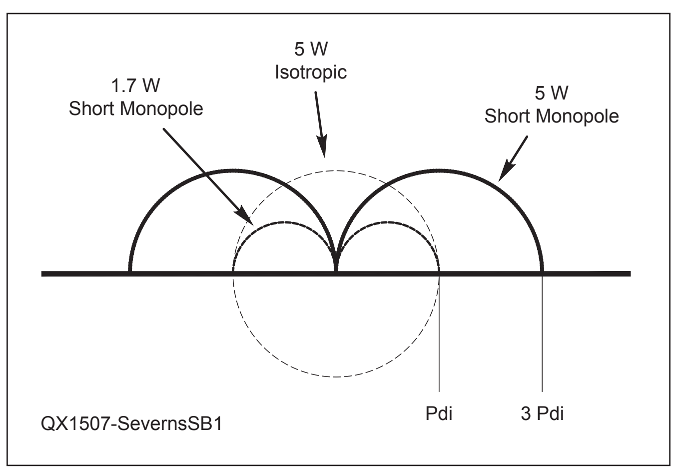


Figure SB1 — Radiation power density at the same radius from an isotropic radiator in free space and a short monopole over perfect ground.

[1]Notes appear on page 34.

(integrate) the power density (in W/m^2) over a hypothetical closed surface surrounding the antenna. For lossless free space calculations the enclosing surface can be anywhere from right at the surface of the antenna to a sphere with a very large radius (large in terms of wavelengths). For P_r calculations, a large radius has the advantage of reducing the field equations to their far-field form, which greatly simplifies the math. This is fine for lossless free space or over perfect ground, where near-field or far-field values give the same answer. When we add a lossy ground surface in close proximity to the antenna, however, things get more complicated. Note that the terms near-field, Fresnel, and far-field are carefully defined in Appendix C.

Take for example a vertical ½ λ dipole with the bottom a short distance above lossy soil. You could create a closed surface that surrounds the antenna but does not intersect ground, and then calculate the net power flow through that surface. When you do this you find the R_i provided by *EZNEC* (my primary modeling software) will be the same as the R_r calculated from the power passing through the surface. Technically, this is R_r by the free space definition, since the antenna is lossless, as is the space within the enclosing surface, but that's not how we usually think of the relationship between R_i and R_r. The conventional point of view is that the near-field of the antenna induces losses in the soil, which we assign to R_g, separate from R_r, as indicated in Figure 1. The power absorbed in the soil near the antenna is not considered to be "radiated" power although clearly it is being supplied from the antenna. When we run a model on *NEC* or make a direct measurement of the feed point impedance of an actual antenna, we get a value for R_i from Equation 5.

$$R_i = R_r + R_g \qquad \text{[Eq 5]}$$

Can we separate R_r from R_g, and if so, how? Assuming we're going to use *NEC* modeling, we could simply use the average gain calculation (G_a). The problem with G_a is that it includes all the ground losses, near and far-field, ground wave, reflections, and so on. For verticals, G_a gives a realistic, if depressing estimate of the power radiated for sky wave communications, but the far-field loss is not usually included in R_g. Typically, R_g represents only the losses due to the reactive near-field interaction with the soil. In the case of a ¼ λ ground based vertical for example, that would be the ground losses out to ≈ ½ λ (see Appendix C). Instead of using G_a we can have *NEC* give us the amplitudes and phases of the *E* and *H* fields on the surface of a cylinder, which intersects the ground surface as indicated in Figure 2.

The power density is integrated over the

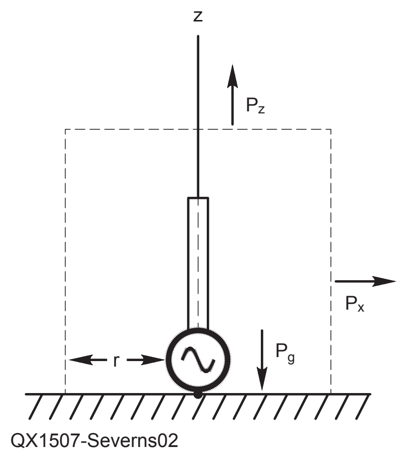


Figure 2 — We can use *NEC* modeling to calculate the *E* and *H* fields on a cylindrical surface enclosing a ground mounted vertical.

surface of the cylinder (P_x) and over the surface of the disc (P_z) that forms the top of the cylinder, giving us P_r directly. Instead of integrating the power over the surface of the cylinder we could sum the power passing through the soil interface at the bottom of the cylinder, which gives P_g directly. From either P_r or P_g we can calculate R_r using Equation 6.

$$R_r = \frac{P_r}{I_o^2} = \frac{\left(P_i - P_g\right)}{I_o^2} \qquad \text{[Eq 6]}$$

Of course this is more complicated than simply using G_a! It turns out, however, that if you're moderately clever in your choice of surface and field components, it can be quite practical to calculate the values using a spreadsheet like Microsoft *EXCEL*. The mathematical details are in Appendix A. Because the fields near a vertical are sums of decaying exponentials (1/r, 1/r^2, 1/r^3) the boundaries between the field regions are not sharply defined, the choice for the cylinder or disc radius (r) is somewhat arbitrary. The rather messy details of the choice of integration surface radius are discussed in Appendix C.

R_r and R_g for a ½ λ Vertical Dipole

For simplicity, I began this study using a resonant vertical ½ λ dipole like that shown in Figure 3, with the bottom of the antenna placed 1 m above ground. The analysis was done at several frequencies, two of which are reported here — 475 kHz and 7.2 MHz. Note the frequencies are a factor of ≈ 16× apart. In a later section, I give an example at 1.8 MHz. The antennas heights (*h*) were adjusted for resonance over perfect ground and that height was retained for modeling

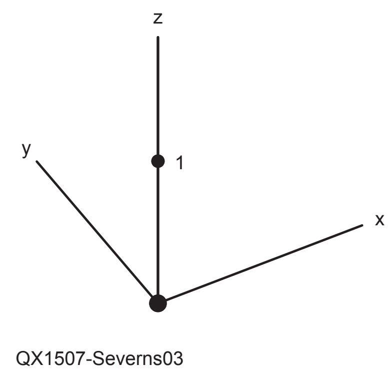


Figure 3 — This model shows a ½ λ vertical dipole, with the bottom of the antenna 1 m above ground.

over real soil.

Figures 4 and 5 show the variation in R_i at 7.2 MHz and 475 kHz for a wide range of soil conductivity (σ) and permittivity (ε_r, relative dielectric constant). The notation "J =" on the Figures indicates the height of the bottom of the antenna above ground.

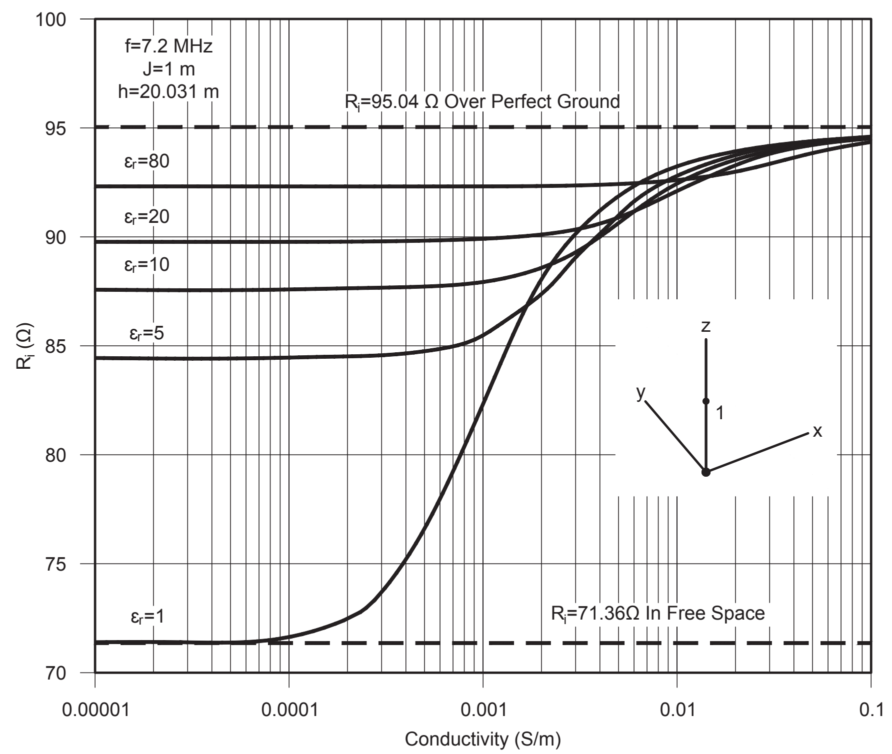
As we would expect, in free space $R_r \approx 72\ \Omega$ and over perfect ground Rr ≈ 95 – 100 Ω for these antennas. Over real ground R_i varies dramatically with both soil characteristics and frequency. One point is obvious:

R_i is not a combination of R_r over perfect ground and some R_g!

On 40 m, values for R_i over real soils are all lower than the perfect ground case, but the values on 630 m vary from well below the perfect ground case to slightly above. In both cases, as ground conductivity increases, R_i converges on the perfect ground case as one would expect. For very low conductivities, we can see that ε_r has a profound influence on R_i, but its effect is greatly reduced for high conductivities. Note that at 475 kHz for σ ≧ 0.0001 S/m, R_i rapidly converges on the perfect ground value, and the effect of ε_r is minimal. On the other hand, at 40 m the jump in R_i doesn't occur until σ ≧ 0.003 S/m, that's more than an order of magnitude higher than 475 kHz. It would appear that at 475 kHz the value for ε_r doesn't matter much over most common soils, but at 7.2 MHz it has a major influence for some typical values of σ. What's going on here?

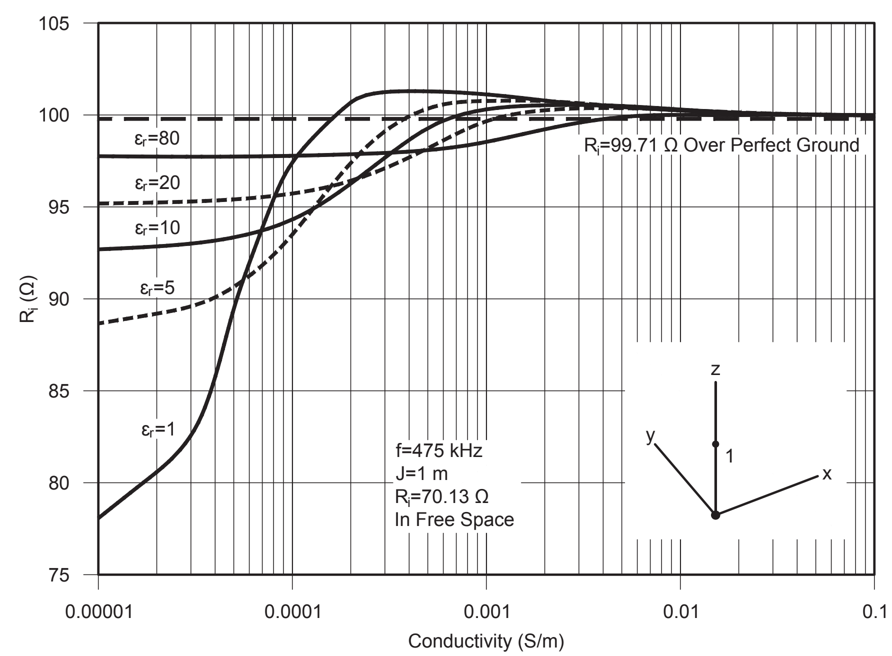
Soil Characteristics

It is important to understand that the characteristics of a given soil will vary with frequency. The following is a brief overview. You can find a much more detailed discussion in Appendix B. Figures 6 and 7 are examples of σ and ε_r for a typical soil over a frequency range from 100 Hz to 100 MHz. These graphs



QX1507-Severns04

Figure 4 — Here is a graph of R_i versus ground conductivity for a ½ λ vertical dipole at 7.2 MHz.



QX1507-Severns05

Figure 5 — This graph shows R_i versus ground conductivity for a ½ λ vertical dipole at 475 kHz.

were generated using data excerpted from *Antennas in Matter* by King and Smith.[5] In this example, at 100 Hz $\sigma \approx 0.09$ S/m and that value is relatively constant up to 1 MHz, beyond which σ increases rapidly. The behavior of the relative dielectric constant (ε_r) is just the opposite, decreasing with frequency until about 10 MHz and then leveling out. We can combine σ and ε_r by using the loss tangent (*D*).

$$D = \tan\delta = \frac{\sigma_e}{2\pi f \varepsilon_e} \quad \text{[Eq 7]}$$

where:

$\varepsilon_e = \varepsilon_o\,\varepsilon_{er}$ = effective permittivity or dielectric constant (in farads/m)

ε_o = permittivity of a vacuum = 8.854×10^{-12} farads/m.

For a good insulator, *D*<<1 and for a good conductor, *D*>>1. For most soils at HF 0.1<D<10, but it is often close to 1.

We can combine the data in Figures 6 and 7 into a graph for *D*, as shown in Figure 8.

Figure 8 shows that something interesting happens when we go from HF down to MF. At HF, *D* is usually not far from 1, but at MF, *D* is usually much higher. This implies that the soil characteristics are dominated by conductivity. Figures 4 and 5 show that at MF, conductivity becomes the dominant influence at much lower conductivities than at HF. This explains some of the features of Figures 4 and 5.

Relationships Between *D*, R_r and R_g

The role of the loss tangent, *D*, is worth exploring a bit further. Figure 4 showed the variation in R_i as ε_r and conductivity were varied. In a similar way we can examine the variation in R_r and R_g over the same range of variables as shown in Figure 9, which is a graph of R_i, R_r, and R_g with $\varepsilon_r = 10$ for the 40 m ½ λ vertical. On the chart there is a vertical dashed line corresponding to values of σ where $D = 1$ for $\varepsilon_r = 10$ ($\sigma \approx 0.004$ S/m in this example). Something interesting happens in the region around the point where the loss tangent equals one.

A very prominent feature of Figure 9 is that R_r and R_g are *not* constant as we vary σ. The value for R_g (which represents ground loss) peaks near $D = 1$, which is what dielectric theory predicts for the maximum dissipation point. We can take one further step with the data in Figure 9, and graph the ratio R_r / R_i (which is the radiation efficiency) as shown in Figure 10. The minimum efficiency (≈0.66) occurs at $\sigma \approx 0.0025$ S/m.

This graph emphasizes the effect of the loss tangent on ground loss.

Acknowledgements

I want to express my appreciation to Steve

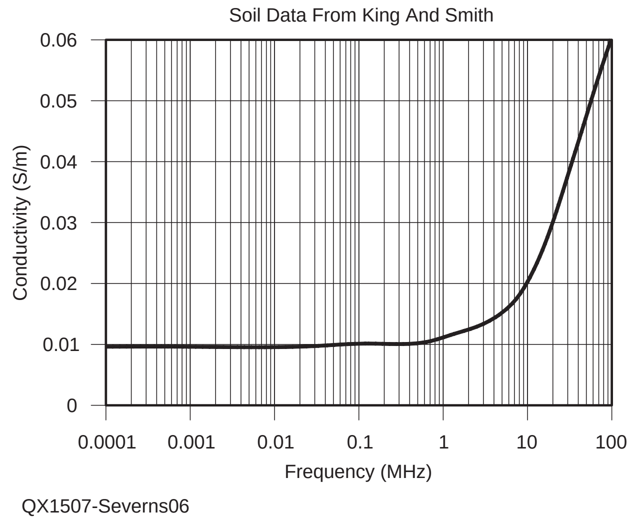


Figure 6 — This graph gives an example of how soil conductivity varies with frequency.

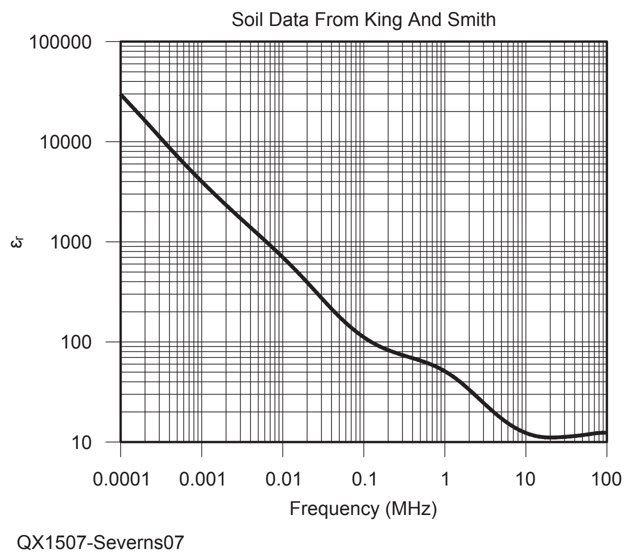


Figure 7 — This graph shows soil permittivity variation with frequency.

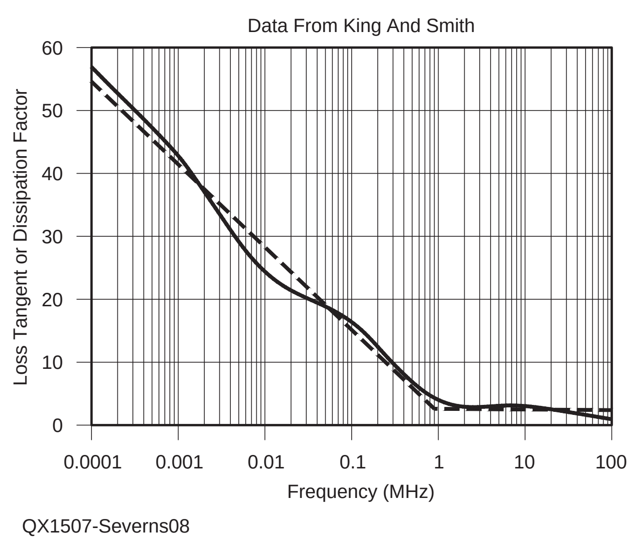
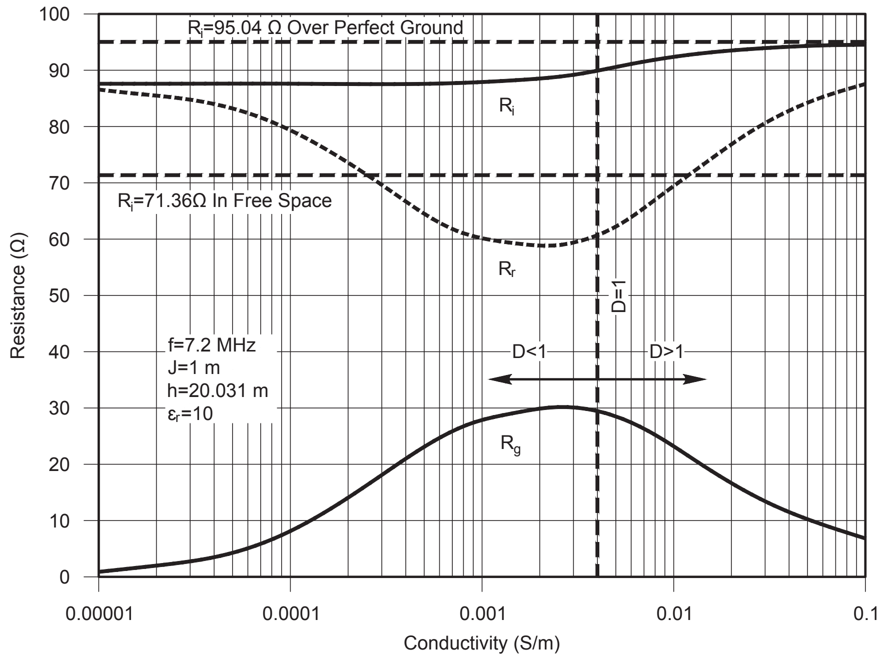
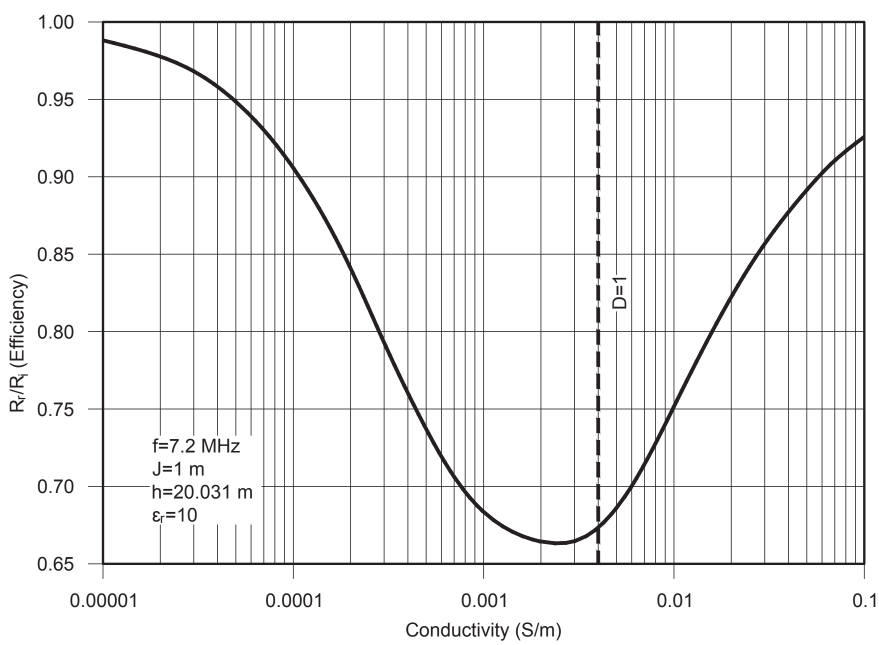


Figure 8 — Here is a graph of the loss tangent associated with the soil in Figures 6 and 7.



QX1507-Severns09

Figure 9 — Variations in R_i, R_r, and R_g with ε_r = 10.



QX1507-Severns10

Figure 10 — Here we see the variation of radiation efficiency with ε_r= 10.

Stearns, K6OIK, for his very helpful review of this article. He put in a lot of effort and I've incorporated many of his suggestions in the main article and in the Appendices. I also appreciate the comments from Dean Straw, N6BV, and Al Christman, K3LC. All of the modeling employed a prototype version of Roy Lewallen's (W7EL) *EZNEC Pro/4* modeling software (see Note 4) that implements *NEC* 4.2, and Dan MaGuire's (AC6LA) *AutoEZ*, which is an *EXCEL* spreadsheet that interacts with *EZNEC* to greatly expand the modeling options. Without these wonderful tools this study would not have been practical and I strongly recommend both programs.

Rudy Severns, N6LF, was first licensed as WN7WAG in 1954 and has held an Amateur Extra class license since 1959. He is a consultant in the design of power electronics, magnetic components and power conversion equipment. Rudy holds a BSE degree from the University of California at Los Angeles. He is the author of three books, more than 90 technical papers and a past editor of QEX. *Rudy is an ARRL Life Member and an IEEE Life Fellow.*

Notes

[1]The Appendices and other files associated with this article are available for downloading from the ARRL *QEX* files web page. Go to **www.arrl.org/qexfiles** and look for the file **7x15_Severns.zip**.

[2]Frederick Terman, *Radio Engineers' Handbook*, McGraw-Hill, 1943.

[3]John Kraus, *Antennas*, McGraw-Hill, 1988, second edition.

[4]Roy Lewallen, W7EL, *EZNEC pro/4*, **www.eznec.com**.

[5]King and Smith, *Antennas in Matter*, MIT Press, 1981, p 399, Section 6.8.

[6]J. Wait, R. Collin and F. Zucker, *Antenna Theory*, Chap 23, Inter-University Electronics Series (New York: McGraw-Hill, 1969), Vol 7, pp 414 – 424.

[7]Dan McGuire, AC6LA, *AutoEZ*, **www.ac6la.com/autoez.html**.

Jim Johnson, WA6OZJ

5036 Rolling Meadows Rd, Rolling Hills, CA 90274: **wa6ozj@earthlink.net**

A Digital Milliohm Meter

Don't have a milliohm meter yet? Here is a project that just might fill that void on your test bench.

There have been two recent articles in *QEX* about milliohm meters, which I read with interest since I've often wished I had such an instrument.[1, 2] The second one really struck a nerve, since it was much simpler than the first one. It used an LM317 type "adjustable" regulator rigged as a constant current source. It also used an analog panel meter with a new scale, however, and tearing a meter apart and putting on a new scale did not appeal to me. It was then I had a BFO (Blinding Flash of the Obvious) — "Why not use a digital panel meter (DPM)?" In this application it is much easier to read than an analog meter.

Digital panel meters are similar to your handy digital multimeter, except they don't have any range switching. DPMs come in several flavors. They usually have either 3½ or 4½ digits (displays values up to 1.999 or 1.9999). There are two power systems to choose from; the 5 V powered unit can share a common ground with the meter leads, or the 9 V powered unit, in which the power and meter leads must be isolated. In my case, I wanted to use a single 9 V battery, so I had to include an LM7805 three terminal regulator to supply the 5 V for meter power.

The circuit diagram is shown in Figure 1. As in Don Dorward's meter, I chose a single range (0 to 1.999 Ω) to simplify things. The current regulator (LM317T) is set to 100 mA, and the DPM is set to 200 mV. This way, with 100 mA flowing through the test resistance, the meter reads directly in mΩ. I used a push-button switch to turn the power on during a test, thus saving battery life. After the button is pushed, it takes a couple of seconds for the meter to come alive and make the reading. The 9 V battery should last quite a long time before the drop-out voltage of either of the two regulators is reached.

There is a variety of digital panel meters available on the surplus market in the range of $10 to $20.[3, 4] The rest of the parts could come from a modest "junk box." I enclosed everything in a plastic project box to keep it all together, as shown in Figure 2. This photo shows me measuring a 4 inch length of #22 AWG copper wire, with my thumb on the push-button switch. The meter is reading a value of 9 mΩ; more on this later.

Figure 3 shows an inside view of the box. The one I found had slots in it to hold circuit boards, so I cut a piece of "perfboard" to fit the box and mounted everything on that. At the top of the photo is the back of the DPM, in the middle is the battery holder, and at the bottom is the perfboard and components. Note the trim-pot on the board. I drilled a hole in the back cover to access the trim-pot. The current regulator needs about a 13 Ω resistor to get the 100 mA current. I used two 33 Ω fixed resistors, and a 200 Ω, 10-turn trimmer potentiometer all in parallel. One turn on the trimmer changes the current by about 1 mA.

With the leads shorted, I get a reading of 6 mΩ, but it should read "0." I tracked this down (see the discussion in the next to the last paragraph), but it may be easier to just subtract "6" from the readings. So in Figure 2, the actual resistance value should be 3 mΩ.

As for calibration, the only adjustment is the 100 ma current. The DPM calibration is fixed. A good milliammeter can be

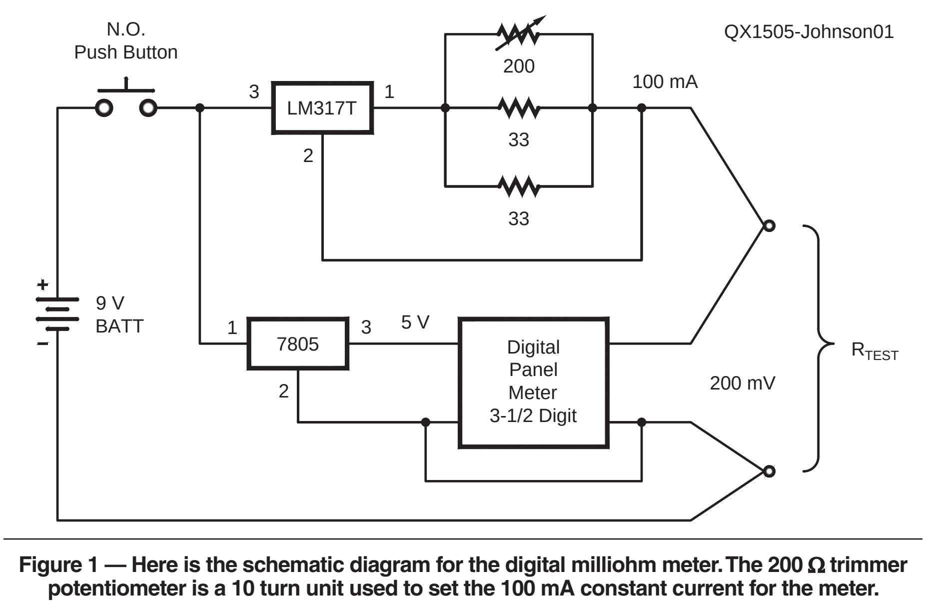


Figure 1 — Here is the schematic diagram for the digital milliohm meter. The 200 Ω trimmer potentiometer is a 10 turn unit used to set the 100 mA constant current for the meter.

[1]Notes appear on page 36.

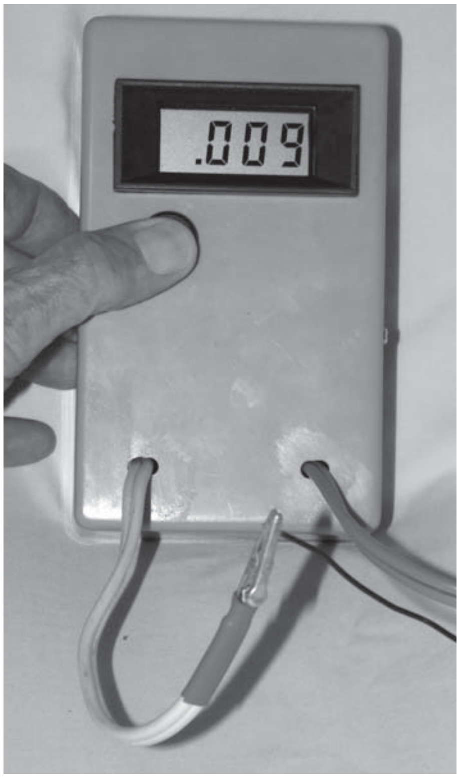


Figure 2 — The author is measuring the resistance of a 4 inch length of #22 AWG copper wire. His thumb is holding down the push-button switch to make the measurement.

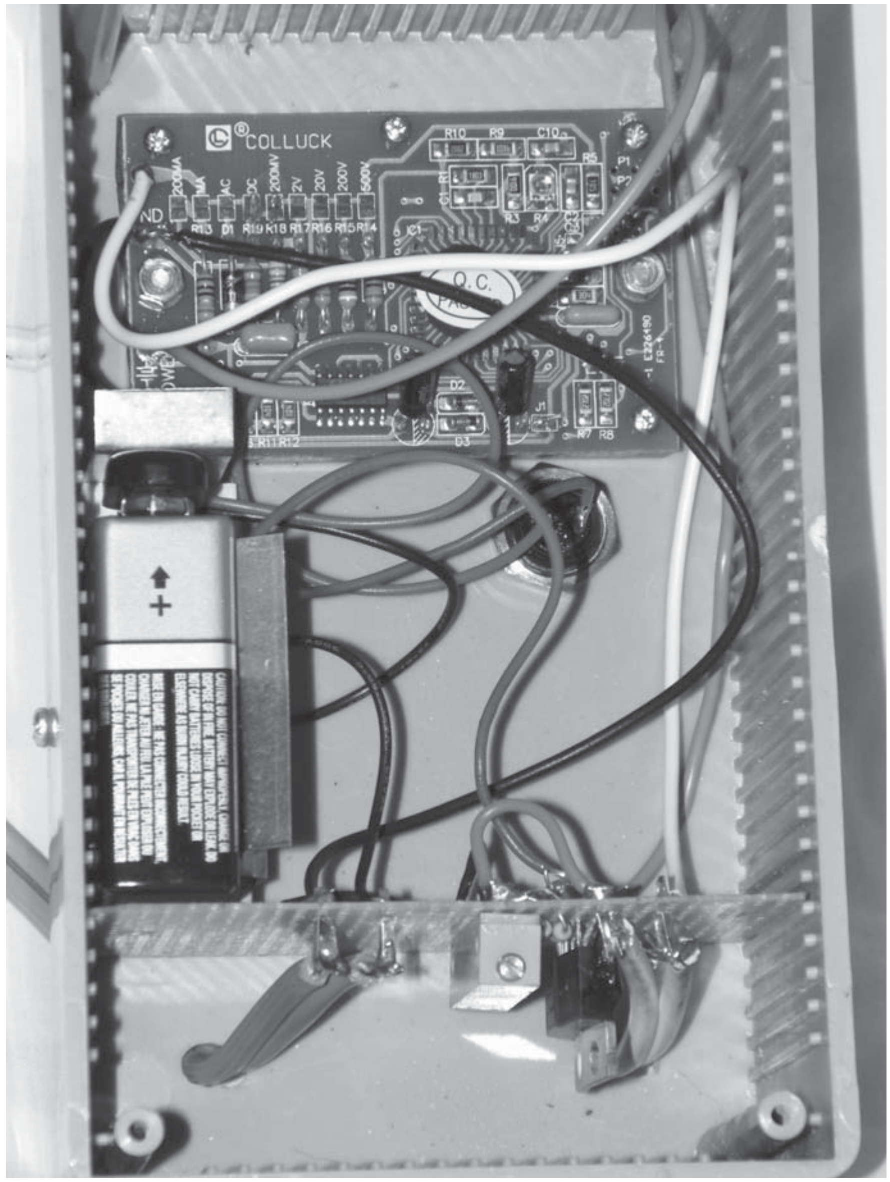


Figure 3 — This is a view inside the project case. You can see the circuit board at the top. The 9 V battery and push-button switch are in the middle section. Near the bottom of the box you can see the piece of perfboard that was cut to slide between ridges on the sides of the box. The 10 turn 200 Ω trimmer potentiometer is visible at the center of the perfboard. A hole in the back panel of the box allows adjustment to set the 100 mA meter current. The test leads pass through a hole in the bottom left corner of the box

inserted between the test leads for setting the current, or you can take a length of copper wire, measure the diameter and compute the resistance, then set the meter reading to that value using the current adjustment. Calibration is a very deep subject, so remember the guy with one clock knows what time it is, while the guy with two clocks is never quite sure.

After investigating the "zero bias" problem, I decided that it was the result of the resistance of the plated steel mini-alligator clips I was using. I tested solid copper regular and mini alligator clips and found that didn't help very much. I considered rewiring the clips so that the "sense leads" were soldered to the very tips of the clips (the current supply leads don't matter), but after all this work, there would still leave the basic contact resistance between the test clips and the part under test. After all this I still think it best (and simpler) to just read the "zero offset" before you start and then subtract that value from the test reading. Adding a "zero offset" feature to the meter would overly complicate a beautifully simple machine.

There are several other meter options. For example, if you reduce the test current from 100 mA to 10 mA, the meter reads full scale 19.99 (20 Ω), and the minimum reading will be 10 mΩ. If you use a 4 ½ digit meter and use the 10 mA test current, the full scale meter reading will be 19.999 Ω like before, but the minimum reading will remain 1 mΩ.

Another great use for the DPM is for a lead–acid battery monitor. Set to the 19.99 range it will measure battery voltage to 0.01 V. Knowing that a battery voltage of 12.75 V is fully charged and a voltage of 12.25 V is virtually depleted, you can instantly assess battery status.

Jim Johnson, WA6OZJ, has been continuously licensed since 1959. He is an ARRL Member. Jim received a BSEE degree from the University of Colorado in 1957. He retired from the aerospace industry with 50 years of experience with systems engineering. That experience included designing, testing, planning, and launching large rockets and satellites (mostly unmanned). This is his first article for an Amateur Radio publication, which he wrote with the hope that others might find the ideas useful.

Notes

[1]Steve Whiteside, N1PON, "A Linear Scale Milliohm Meter," *QEX*, Jul/Aug 2012, pp 33 – 38.

[2]Don Dorward, VA3DDN, "A Linear Scale Milliohm Meter; Another Look," *QEX*, Jul/Aug 2014, pp 23 – 26.

[3]Digital panel meters and other components for this project are available from Marlin P. Jones & Associates, Inc; **www.mpja.com**.

[4]Another source for digital panel meters and other components for this project is Jameco Electronics; **www.jameco.com**.

Scotty Cowling, WA2DFI

PO Box 26843, Tempe, AZ 85285: **scotty@tonks.com**

Hands-On SDR

Sharing Radios on the Network

I would like to thank those of you that took the time to drop me an e-mail and let me know that my introduction to FPGA programming for SDR was useful. In spite of the positive feedback, I am going to shift gears this installment. FPGA aficionados do not despair; I will return to the world of *Verilog* in my next column. As Monty Python would say, "...and now for something completely different".[1]

In this column, I will show you how to listen to various web-based SDRs around the world. I will then walk you through the procedure to set up your own server and share your radio with others, both on your local network and on the Internet. I am going to limit my discussion to receive only. Transmitting is possible, but in my opinion, it is not quite ready for prime time. It is much easier for many "listeners" to listen to the same digitized and down-converted data from the antenna than it is to figure out a way to combine the audio from many "talkers" into one stream of data, and up-convert it for transmission. Well, maybe it is not *that* hard to do, but this feature isn't readily available in any of the software that I could find on line. This leads me to define the difference between software ***controlled*** SDR and software ***shared*** SDR. A software controlled SDR is a radio intended to be used by one operator. A software shared SDR is intended to be used by more than one operator. A shared SDR may or may not be capable of supporting simultaneous users; a controlled SDR does not even need this feature. See the sidebar, Networks, Servers and Clients.

[1]Notes appear on page 42

What Do I Need?

As usual, you probably want to know the answers to a few questions: "What do I need to know?" and "What equipment do I need?"

As a bare minimum, you need nothing more than an Internet connected computer running a browser. Since this is a *Software* Defined Radio column, and you need a computer to run the software, it is not much of a leap of faith to assume that you already meet the minimum criteria! All of the software that I use in this month's column is available for free download. You must be familiar with installing and running *Windows* applications to take advantage of this free software. To get the most out of the final section on the *QtRadio* server, you must be able to install and run Ubuntu *Linux* and its applications.

You do not need any radio hardware if you do not intend to set up a server; you will be using other people's radio hardware instead. To run a **RemoteSDR** server, you will need an RFSPACE radio (such as an SDR-IQ). To run a **ghpsdr3-alex** server, you will need one of the many supported radios.[2] Even if you do not have any radio hardware and just choose to follow along with the text, you can still get in on the remote radio action and have lots of fun!

Listen to Someone Else's Radio

The simplest way to get your feet wet in Internet shared radios is to just point your

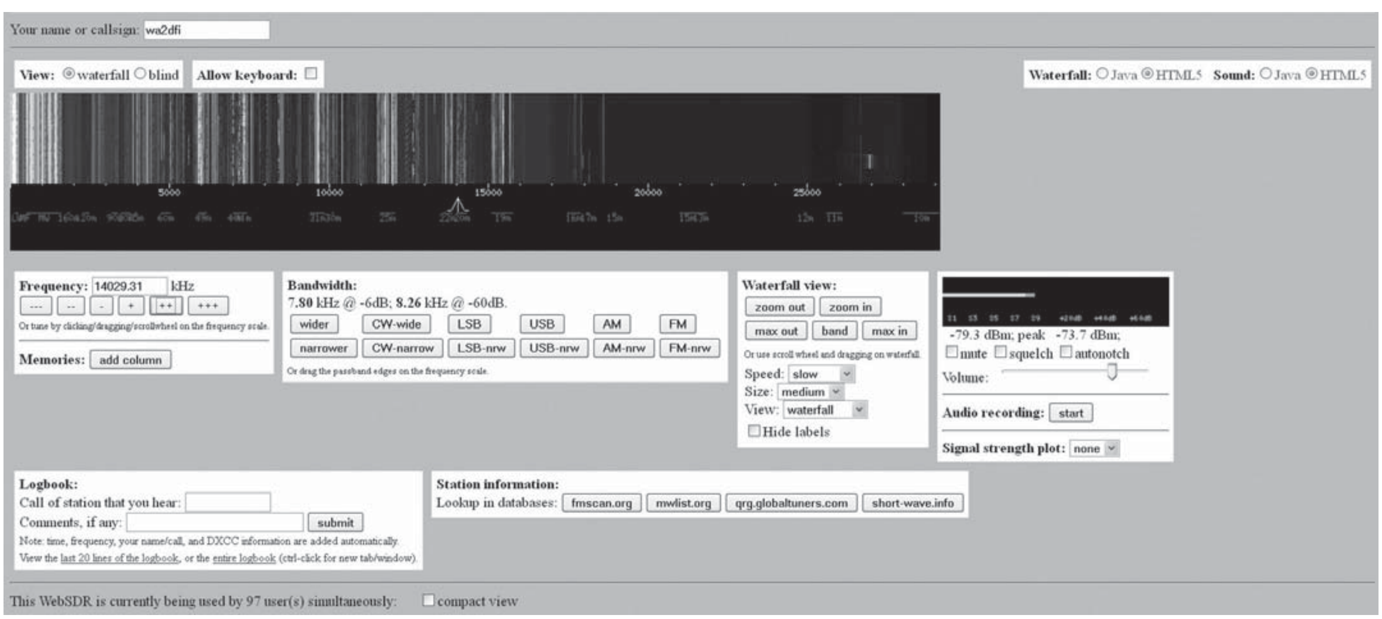


Figure 1 — You can listen to the University of Twente browser-based WebSDR at: websdr.ewi.utwente.nl:8901.

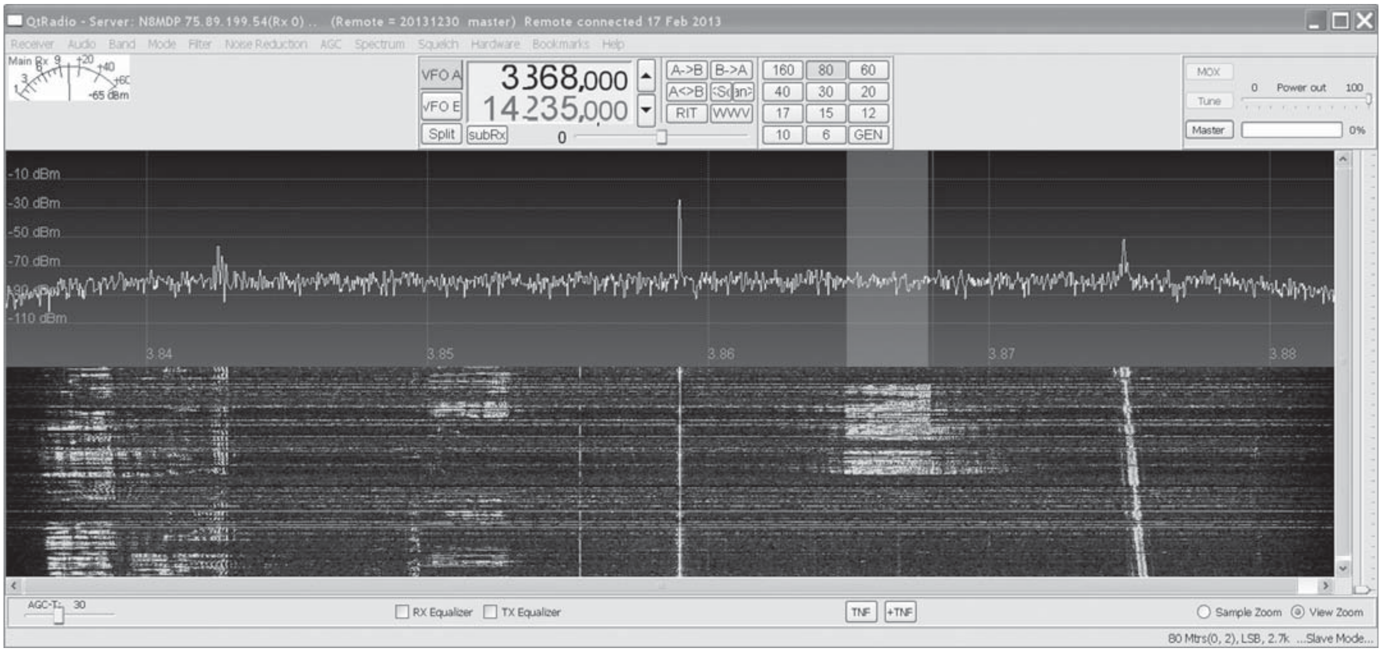


Figure 2 — This is a screenshot of the *Windows QtRadio* Client.

browser here: **websdr.ewi.utwente.nl:8901**. This is the University of Twente WebSDR in The Netherlands. WebSDR is a wide-band, multi client web-based SDR that was originally set up in 2008, making it the oldest on-line SDR. See Figure 1. Their hardware has progressed through several increasingly sophisticated iterations, and their web site contains a wealth of historical and technical information. Best of all, WebSDR does not require you to install any software on your computer in order to listen!

Now let's move on to something that is a bit more involved to set up, namely installing client software on your device. Notice that I said *device* instead of *computer*. The software we will use next is available for many devices, such as Android phones and tablets, computers and tablets running Mac *OSX*, *Linux* and *Windows*. On phones and tablets, the client software is called an *app*, short for *application*. The client for *Windows* is called *QtRadio*, and you can download a pre-built runtime version from the *QtRadio* website.[3] The Android app is called *glSDR* and is available for free from the Google play store.[4] The source files for *Linux* and Mac *OSX* are available, but you will have to compile them yourself. We will get to this later. For *Windows*, simply download the .zip file and extract it to a new directory. Run the **QtRadio.exe** program in the main directory by double-clicking it; there is no installation required. To make it easier to access, you can create a shortcut and place it on your desktop.

Once *QtRadio* is running (Figure 2), click on the **Receiver** menu in the upper left corner and select **Quick Server List**. A new window will open (Figure 3) showing all of

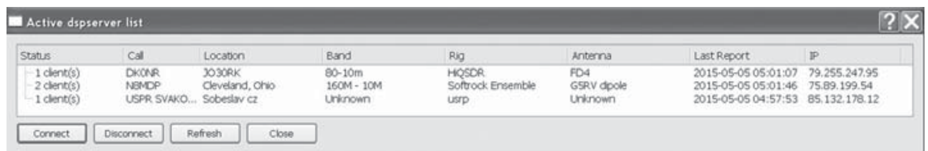


Figure 3 — Here is a screenshot of the *QtRadio* active dspserver list.

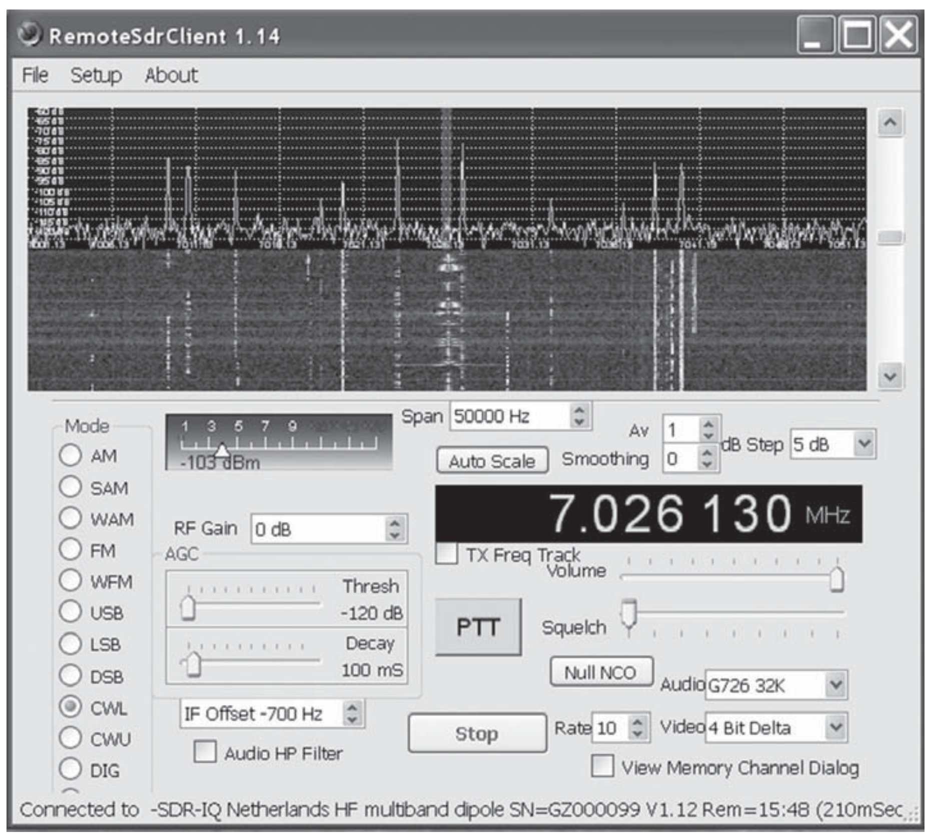


Figure 4 — A screenshot of the *RemoteSDR* Client connected to an SDR-IQ in The Netherlands.

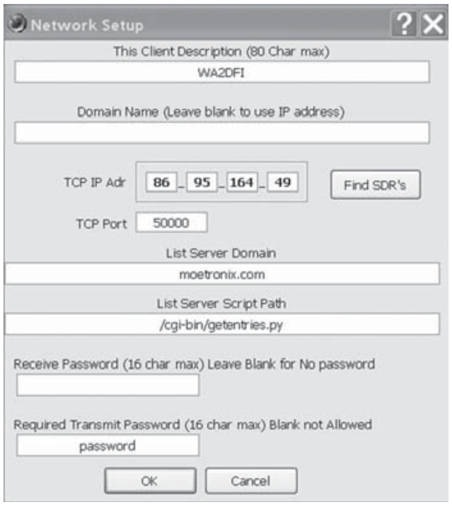


Figure 5 — This is the *RemoteSDR* Client Network Setup screen.

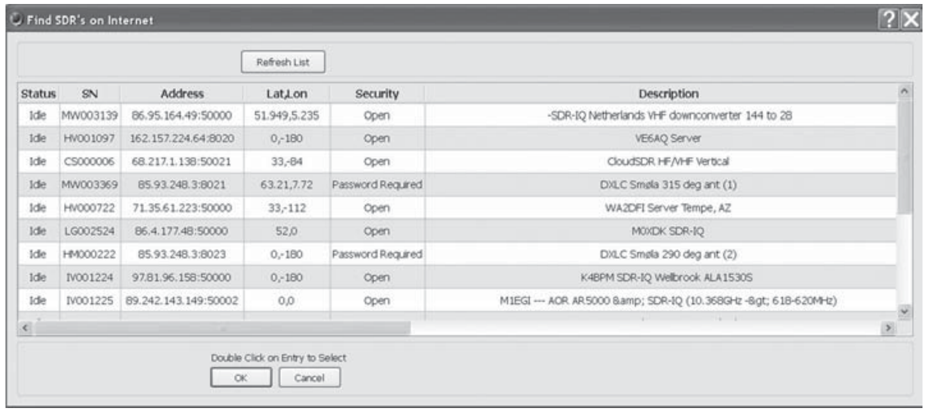


Figure 6 — This is the *RemoteSDR* Client "Find SDR's" screen.

the available dspservers. Double click on one that has 0 clients (highlighted in green) and you are listening on that radio! If you select a radio that is in use, you will not be able to control it, but you will instead hear whatever the master user (the one who is controlling the radio) hears. You only get to control the radio if you were first in line to select it.

RemoteSDR: Easy as Cake

Mangled idioms aside, I would like to segue into the server application, but the *QtRadio* server only runs under *Linux* and is a bit complex to set up. Let's take a look instead at somewhat more hardware limited server/client software that runs on *Windows* but is very easy to install. The limitation is that it requires an RFSPACE radio to act as a server. Even if you do not have an SDR-IQ or SDR-14 receiver from RFSPACE, installing this software will be a useful way to familiarize yourself with simple SDR server/client systems.

The *RemoteSDR* software is available as a free download, and it installs both the server and client programs in one operation.[5] Download the Quick Start Guide (**QuickStartGuide107.pdf**) and the server/client software installer (**RemoteSdrClientWinSetup_114.exe**). Run the installer and select all the default options. This will install both the client and the server on your *Windows* computer. You will see two new icons on your desktop: one for the server and one for the client. Go ahead and launch the client (Figure 4), click on the **Setup** menu and select **Network** (Figure 5). Enter a description for your client and then click the **Find SDR's** button. (Yes, I know that should be plural rather than possessive, but that's the way the programmers typed it!) Figure 6 shows a listing of the SDRs that *RemoteSDR* found. Double click on your selection, click **OK** and then click on the **Start** button. You are now in control of the remote radio!

Now that we can listen to someone else's radio, let's set up the server to let others listen to our radio. Unless you have an RFSPACE radio such as an SDR-IQ, you will not be able to do this, but let's cover it quickly before moving on to a more complex server that supports many different hardware platforms.

When you installed the *RemoteSDR* client application, remember that you also installed the *RemoteSDR* server application. Find its icon on your desktop and open it (Figure 7). Click on **Setup** and select **Server Setup** (Figure 8). All you need to enter here is your SDR description and your location in the **Latitude** and **Longitude** boxes and click **OK**. Figure 9 is a block diagram of the *RemoteSDR* client and server system.

Now open the *RemoteSDR* client and go to **Network Setup** and **Find SDR's**. You should now see your own SDR listed among the others. Try double clicking your own radio, then **OK** and **Start**. Did it work? It probably did not (mine didn't). Why didn't it work? Go back to the client Network Setup screen and look at the TCP address. This is the IP address that was published by the server software. It is the IP address that your ISP provided to your modem and will not be accessible from the local network side that your PC is connected to. You must use a local address, such as 127.0.0.1 (localhost) or the actual IP address of your PC on the local network. Look at the *RemoteSDR* server window for these addresses. Pick one and type it into the client Network Setup screen's **TCP IP Adr** boxes. Then click **OK** and **Start** and you will be listening to your own radio!

In order for others to listen to your radio on the Internet, you have to do one more thing: forward the data from your Internet IP address port 50000 to your local network port 50000. See Figure 10 for an explanation of why this needs to be done. You must set up a route in your home router to make this connection, so that the external clients (on the Internet) can connect to your server on your local network. Note that your radio will appear in the list of Internet SDRs even without port forwarding, but no one outside your local network will be able to connect to your radio and listen until you set up port forwarding. For a more

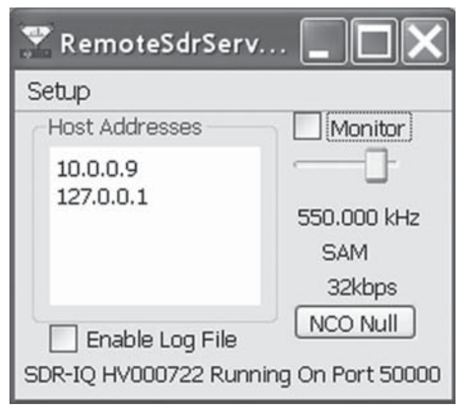


Figure 7 — Here is the *RemoteSDR* Server screen.

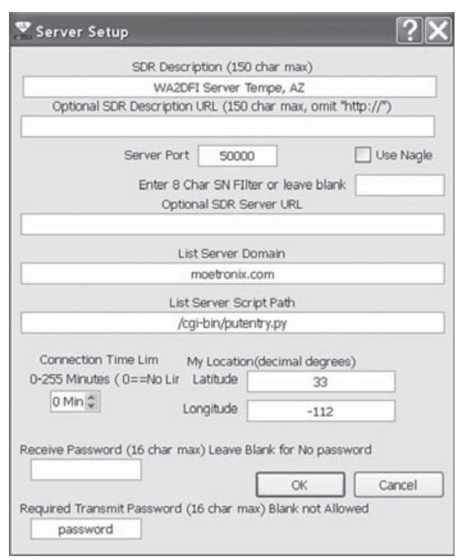


Figure 8 — The *RemoteSDR* Server Setup screen allows you to configure your server.

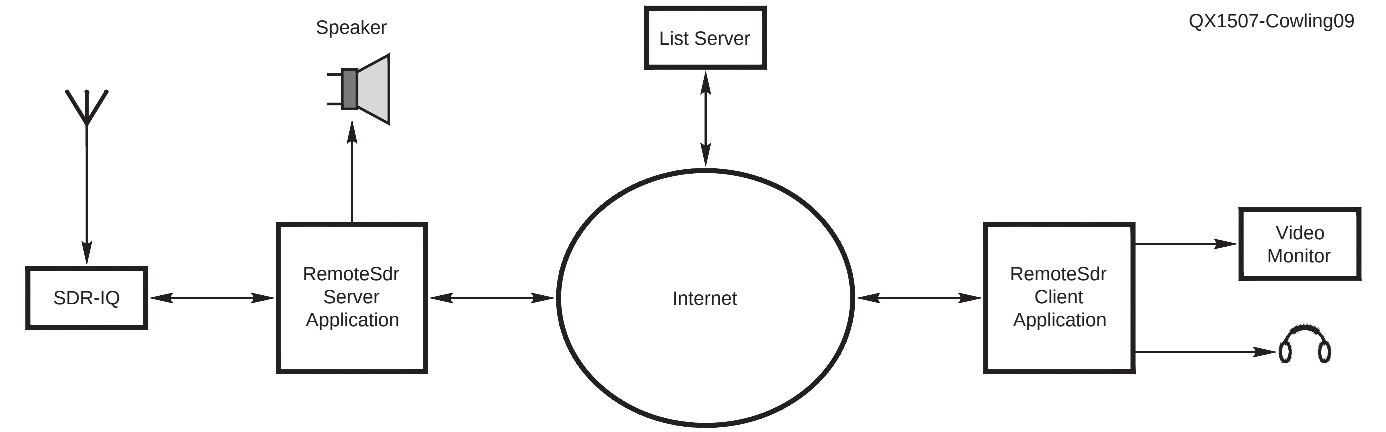


Figure 9 — ***RemoteSDR*** **system block diagram (adapted from the graphic on the RFSPACE website).**

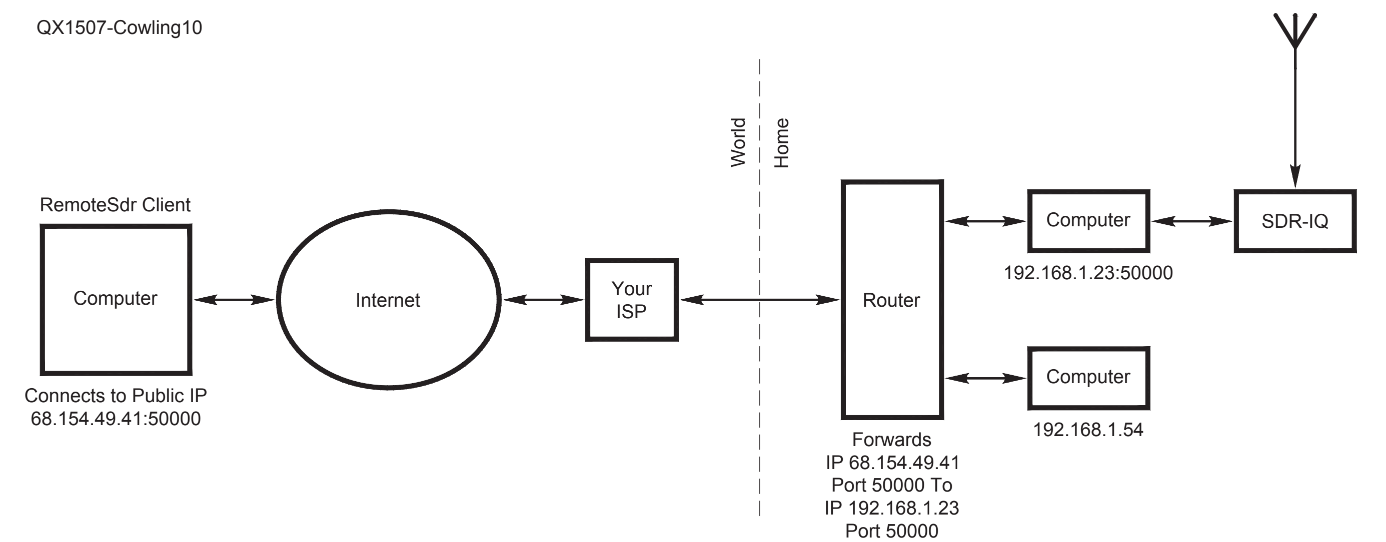


Figure 10 — ***RemoteSDR*** **system port forwarding diagram (adapted from the graphic on the RFSPACE website)**

detailed discussion, see section 3 of the *RemoteSDR* Quick Start Guide.

And Now for Something Completely Different (again)…

Actually, our final exercise is a tough one. It is really more of the same as what we have already done, but with different hardware and under a different operating system. Let's set up a ghpsdr3-alex server under Ubuntu 14.04 *Linux*. We have already run the *QtRadio* client software to listen to other servers on the Internet. The obvious next step is to set up a server of our own. The good news is that the ghpsdr3-alex server software supports many different hardware platforms (see Note 2). The not-so-good news is that it runs only under *Linux* and is not trivial to set up. We have never let that stop us before, however, so let's dive in!

Ghpsdr3-alex Server Linux Setup

Thanks to Dan Babcock, N4XWE, we have a 10-step guide to install and run ghpsdr3-alex on a computer running Ubuntu 14.04 LTS 32-bit Linux. This will install all three parts of the software. The **Server** talks to the hardware and the **dspserver** does the heavy lifting and serves up receive data to the ***QtRadio*** receiver client (see Figure 11).

Step 1: This step will make sure that your system is up to date and has the compiler tools installed. Make sure that the universe repository is enabled (it is enabled by default). Open a terminal window and run the following commands.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install make
$ sudo apt-get install gcc
$ sudo apt-get install g++
$ sudo apt-get install autoconf
$ sudo apt-get install automake
$ sudo apt-get install autotools-dev
$ sudo apt-get install libtool
$ sudo apt-get install git
$ sudo apt-get install subversion
```

Step 2: Use the following command 20 times to install the 20 prerequisite packages listed by name in Figure 12. Be sure to substitute the name of each package from that list for *package* in this command.

$ sudo apt-get install ***package***

Step 3: Install qt5. Visit **qt.io/download-**

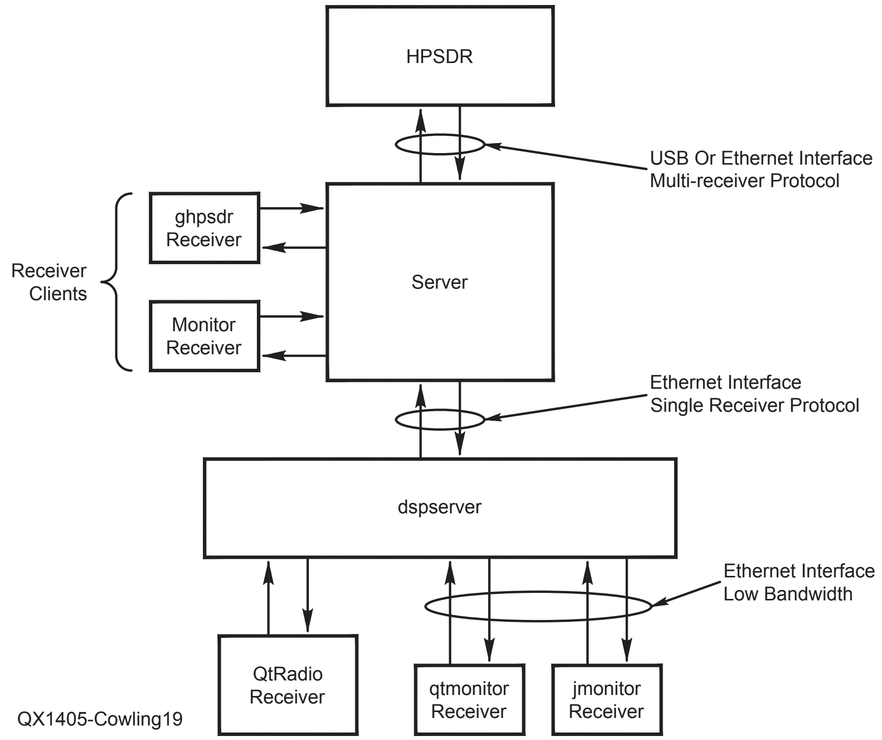


Figure 11 — This is the ghpsdr3-alex system diagram.

cmake
freeglut3-dev
gcc-multilib
libconfig8-dev
libevent-dev
libfftw3-dev
libglu1-mesa-dev
libpulse-dev
libortp-dev
libqt4-opengl-dev
libsamplerate0-dev
libspeexdsp-dev
libssl-dev
libusb-0.1-4
libusb-1.0-0-dev
libusb-dev
libxcb-composite0-dev
portaudio19-dev
qtmobility-dev
xdg-utils

Figure 12 — This list shows the prerequisite software packages to be installed in Step 2.

open-source and download the current package for your system. Do not use the recommended one; go to the **Offline Installers** page and select the version that matches your system (for example, **Qt 5.4.1 for Linux 32-bit**). Change the file permissions to allow execution, and execute it. Find where the **qmake** file is installed using the output from the locate command. Substitute this path for the example path in the export command below. The path may vary slightly if you are using 64-bit *Linux*.

$ chmod +x ***downloaded_file_name***
$ sudo ./***downloaded_file_name***
$ sudo updatedb
$ locate /bin/qmake
$ export PATH=/opt/Qt5.4.1/5.4/gcc/
bin:$PATH

Step 4: Verify that path points to the qt5 version that you installed in Step 3 above. (The qmake version is not important.)

$ qmake -v

The result should be something like below showing the Qt version that you installed in Step 3. The path may vary slightly if you are using 64-bit *Linux*.

QMake version 3.0
Using Qt version 5.4.1 in /opt/
Qt5.4.1/5.4/gcc/bin

Step 5: Install the codec2 low bit rate CODEC for audio encoding and decoding.

$ cd
$ mkdir src
$ cd src
$ svn co https://svn.code.sf.net/p/
freetel/code/codec2 codec2
$ cd codec2
$ mkdir build
$ cd build
$ cmake ../
$ sudo make
$ sudo make install
$ sudo ldconfig

Step 6: Move the codec2 header files to the correct place.

$ cd /usr/local/include/codec2
$ sudo cp * ..

Step 7: Move the codec2 libraries to the correct place. The directory in the first command is for 32-bit Ubuntu. For 64-bit Ubuntu it will likely be /usr/local/lib/x86_64-linux-gnu instead.

$ cd /usr/local/lib/i386-linux-gnu
$ sudo cp * ..

Step 8: Install ghpsdr3-alex from the Git repository.

$ cd
$ cd src
$ git clone git://github.com/
alexlee188/ghpsdr3-alex
$ cd ghpsdr3-alex
$ git checkout master

Step 9: Compile the code.

$ autoreconf -i
$ ./configure
$ make -j4 all
$ sudo make install

Step 10: Run the server, dspserver and client software. You will have to run each of these in a separate window. To open a new window, type <crtl><alt>T. Before you run **hpsdr-server**, make sure that your openHPSDR, Hermes or SDRstick hardware is connected to the network and powered up. If you are running other SDR hardware, use the appropriate hardware server and command-line options in place of **hpsdr-server**.

$ hpsdr-server --metis --samplerate
384000
$ dspserver --hpsdr --lo 0
--nocorrectiq
$ QtRadio

When you run the second line in step 10 for the first time, the software will create a new **dspserver.conf** template file for you to fill in with important things like your call sign, location, band, rig and antenna. This information will be listed in the web database that you see when you bring up the **Quick Server List** in *QtRadio*. Edit this with your favorite text editor to reflect your station information. When you run the server for the second time (after you have edited the **dspserver.conf** file) you will be prompted to create a key and self-signed certificate. The three commands to do this are listed in the prompt, but I have reproduced them below to make it a bit easier for you after they have scrolled off your screen. Run these three commands and answer any questions you are asked.

```
$ openssl genrsa –out pkey 2048
$ openssl req –new –key pkey –out cert.req
$ openssl x509 –req –days 365 –in cert.req –signkey pkey –out cert
```

To test your setup, click on **Receiver** in the *QtRadio* toolbar and select **Configure** (see Figure 13). In the **Server** tab, enter 127.0.0.1 and click **Add Host** (or just pick 127.0.0.1 from the drop-down list). Click Close, then click on **Receiver** again, but this time select **Connect**. A dialog box may pop up with additional settings, depending on the hardware that you are running. You can make your choices and dismiss the dialog box, or you can just drag it out of the way for now. Meanwhile, you should be up and running your own shared receiver! The IP address 127.0.0.1 is called the localhost address, and connects the *QtRadio* client to the **dspserver** via a direct connection within the computer.

Note that your receiver is now shown in the list of online dspservers when you display them by clicking **Receiver** and selecting **Quick Server List**. You will notice, however, that the IP address is not right, and you cannot connect to it. There is one more thing that we need to do before others can connect to our local receiver, and that is set up port forwarding for ports 8000 and 9000. Since there are many kinds of routers, I cannot give you specifics on how to do this. On the Netgear router that I have, I set up port forwarding so that any incoming packets from the WAN (outside world) addressed to any IP address on ports 8000 or 9000 forward to the local computer on the LAN (local network) that is running dspserver (mine is at address 192.168.1.2). To make things easier, you will probably need to set this computer's IP address to a fixed value rather than use DHCP to assign it. After you set up port forwarding, the IP address that you see in the **Active dspserver** list should be the IP address of your cable modem or router, and you (and everyone else, too) should be able to connect to your receiver from anywhere on the Internet.

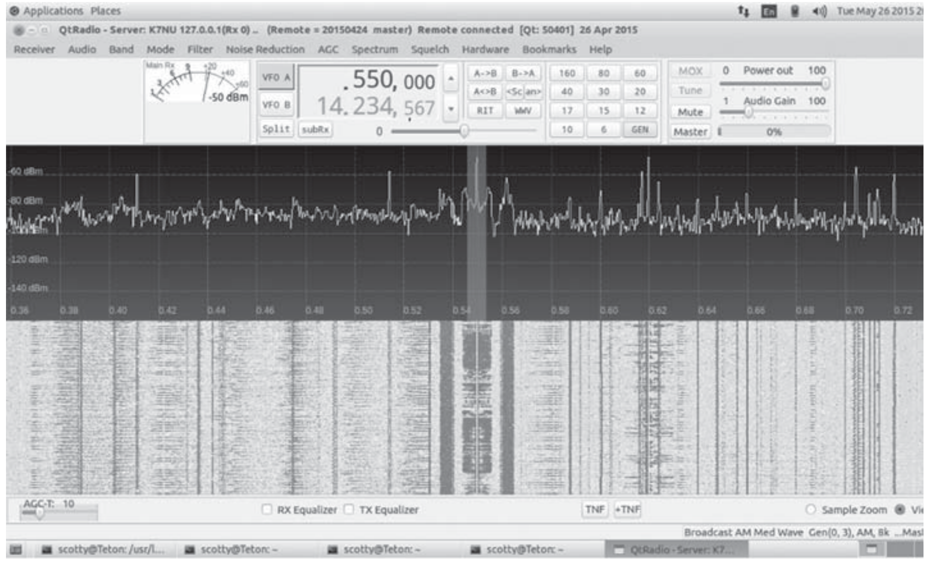


Figure 13 — This screenshot shows the *Linux QtRadio* Client.

Networks and Servers and Clients, Oh My!

While it might be obvious to networking and software gurus, the terms **server** and **client** might not be as familiar to radio experts. In simple terms, a server is a program that serves up data to other programs (called clients) that use the data in some way (display it on your screen, for example). The Internet consists of many (and I mean *billions*) of servers that deliver content to web clients. This content can be HTML web pages, MP3 music streams, MPEG4 video streams or even SDR control and audio streams.

A common example of a client is your web browser. Whether you use Firefox, Chrome, Internet Explorer or something else, this client software connects to a server on the Internet to obtain its content, which it then formats and displays on your monitor (or sends it to your speakers or to another application running on your computer). Clients can also be set-top boxes, game consoles, home theater computers, and the list goes on and on.

In our specific case, the server provides the SDR content to one or more clients. The clients may request specific data (frequency, mode, bandwidth, and so on) from the server, and (if the system is designed properly) the server responds with the appropriate data. It is this server/client architecture that enables many users to share data from one set of receiver hardware.

Back to *Verilog*

After this brief sojourn into the world of Internet shared radios (and now that you are a *Linux* and networking expert), my next column will return to the topic of *Verilog* programming for FPGAs. I will take a look at porting the open-source code for the Altera Cyclone III FPGA on the openHPSDR Hermes board to the Cyclone V FPGA on the BeMicroCVA9 development board from Arrow.[6] The result will be open-source SDR code that anyone can use as a starting point for their own customization, port to yet a different FPGA family or just become more familiar with SDR FPGA design techniques.

Notes

[1]See (**en.wikipedia.org/wiki/And_Now_for_Something_Completely_Different**) for information on the origin of the phrase.

[2]As of this writing, ghpsdr3-alex software supports the following hardware: HPSDR, SDRstick, Softrock, UHFSDR, Perseus, SDR-IQ, HiQSDR, USRP and RTL-SDR DVB-T dongle.

[3]Download the *QtRadio* client application from here: **napan.com/ve9gj/QtRadio_Windows_Master_2014-09-02.zip**

[4]The Android glSDR app is available here: **code.google.com/p/sdr-widget/downloads/detail?name=glSDR32.apk&can=2&q=**

[5]The *RemoteSDR* software and Quick Start Guide are available on this page: **sourceforge.net/projects/remotesdrclient/files**

[6]The BeMicroCVA9 should be available from Arrow Electronics by the time you read this: **parts.arrow.com/item/detail/arrow-development-tools/bemicrocva9**

Upcoming Conferences

Central States VHF Society

Denver, Colorado
July 23 – 26, 2015
Denver Marriott Westminster
7000 Church Ranch Boulevard
Westminster, CO 80021
Reservation Phone: 720-887-1177

Reservation Website: **www.marriott.com/hotels/travel/denmw-denver-marriott-westminster/?app=resvlink**

Our sponsor this year is Rocky Mountain Ham Radio (**www.rmham.org/wordpress/**). The conference will feature the traditional activities, including a banquet, luncheons and hospitality suites, technical programs, noise figure measurement, antenna range, and Rover vehicle show and tell. We have a wide variety of activities available along the Front Range of Colorado and will be offering a choice of side trips designed to entertain the entire family. Operating opportunities under consideration include microwave operating from local mountain tops and the chance to score a microwave VUCC in a weekend! A Sunday VHF 102 introduction geared to newcomers to weak signal operation on the VHF+ bands will be promoted locally and designed to encourage younger hams to get involved in DXing and contesting.

Rooms will be $109 per night and this rate will be available 3 days prior to and three days after the conference for those wishing to take advantage of this opportunity to explore the many wonders the Colorado Front Range has to offer. The Conference website includes a link to the Hotel registration page. Please use the host hotel as it makes it possible for us to bring the conference to you at the cost we've negotiated.

At this time we would like to encourage any and all amateurs interested in presenting papers and programs on subjects that would be of interest to the CSVHFS conference weak signal attendees to contact our Program Chairman John Maxwell, WØVG at **w0vg@arrl.net**.

The 34th Annual ARRL and TAPR Digital Communications Conference

Chicago, Illinois
October 9 – 11, 2015
DoubleTree by Hilton Chicago - Arlington Heights
75 West Algonquin Road, Arlington Heights, IL 60005
Reservation Phone: 847-364-7600

Reservation Website: **doubletree.hilton.com/en/dt/groups/personalized/C/CHIARDT-DCC-20151007/index.jhtml**

Now is the time to start making plans to attend the premier technical conference of the year, the 34th Annual ARRL and TAPR Digital Communications Conference. This year's DCC will be held October 9 – 11, 2015 in Chicago, Illinois, at the DoubleTree by Hilton Chicago, in Arlington Heights, IL. Regular attendees will note that the conference is a couple of weeks later than normal this year. It is the Columbus Day Weekend.

The ARRL and TAPR Digital Communications Conference is an international forum for radio amateurs to meet, publish their work, and present new ideas and techniques. Presenters and attendees will have the opportunity to exchange ideas and learn about recent hardware and software advances, theories, experimental results, and practical applications.

Topics include, but are not limited to: Software defined radio (SDR), digital voice (D-Star, P25, WinDRM, FDMDV, G4GUO), digital satellite communications, Global Position System (GPS), precision timing, Automatic Packet Reporting System® (APRS), short messaging (a mode of APRS), Digital Signal Processing (DSP), HF digital modes, Internet interoperability with Amateur Radio networks, spread spectrum, IEEE 802.11 and other Part 15 license-exempt systems adaptable for Amateur Radio, using TCP/IP networking over Amateur Radio, mesh and peer to peer wireless networking, emergency and Homeland Defense backup digital communications, using Linux in Amateur Radio, updates on AX.25 and other wireless networking protocols and any topics that advance the Amateur Radio art.

This is a three-Day Conference (Friday, Saturday, and Sunday). Technical sessions will be presented all day Friday and Saturday. In addition there will be introductory sessions on various topics on Saturday.

Join others at the conference for a Friday evening social get together. A Saturday evening banquet features an invited speaker and concludes with award presentations and prize drawings.

The ever-popular Sunday Seminar has not been finalized yet, but is always an excellent program. This is an in-depth four-hour presentation, where attendees learn from the experts. Check the TAPR website for more information: **www.tapr.org**.

Call for Papers

Technical papers are solicited for presentation and publication in the *Digital Communications Conference Proceedings*. Annual conference proceedings are published by the ARRL. Presentation at the conference is not required for publication. Submission of papers are due by 17 August 2015 and should be submitted to: Maty Weinberg, ARRL, 225 Main Street, Newington, CT 06111, or via the Internet to **maty@arrl.org**. There are full details and specifications about how to format and submit your paper for publication on the TAPR website.

Even if you are not presenting a paper at the conference, plan to bring a project or two to display and talk about in the popular Demonstration Room, or "Play Room" as it is commonly known.