QEX

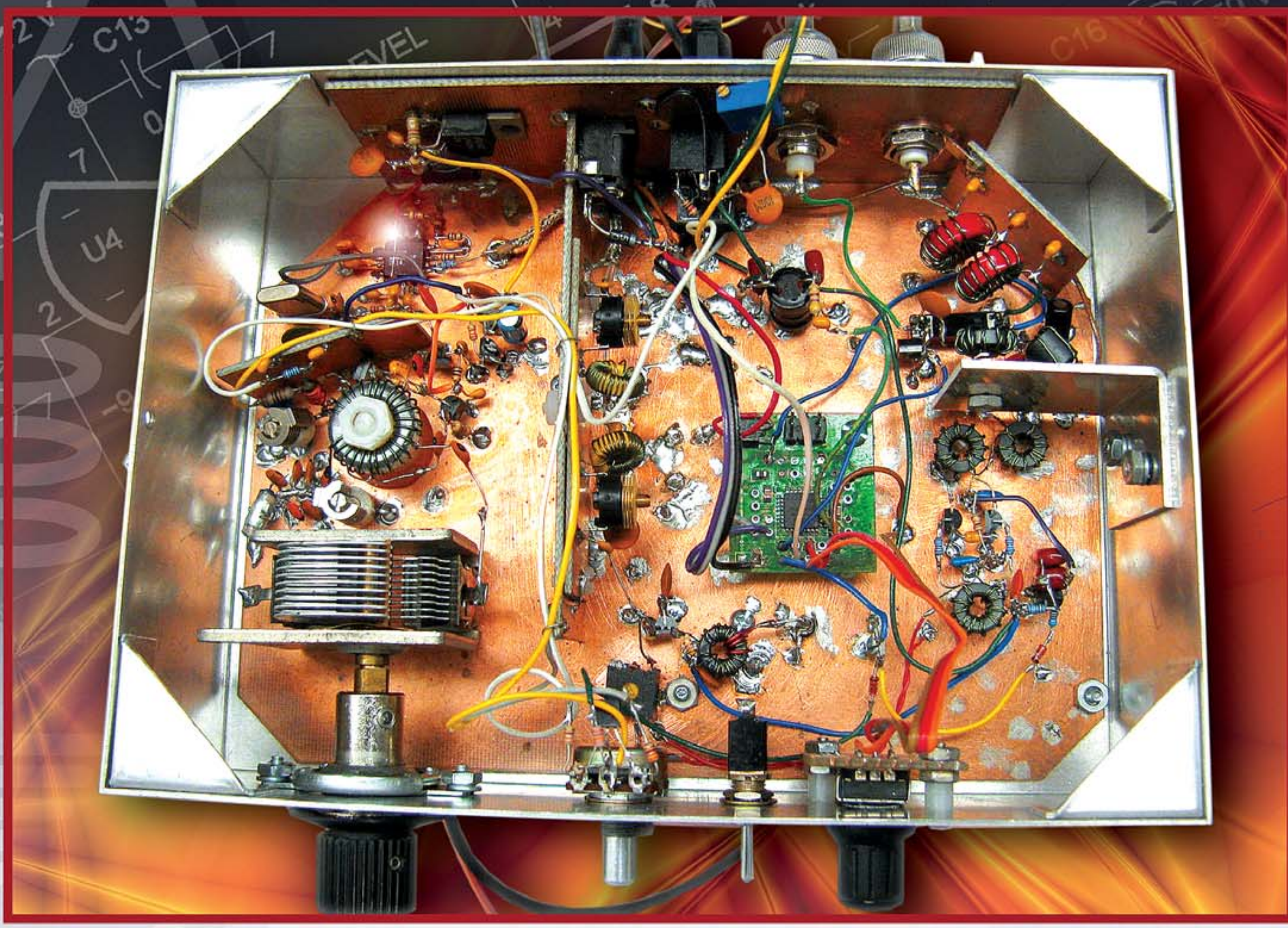
$5

March/April 2012

www.arrl.org

A Forum for Communications Experimenters

Issue No. 271



ND7PA gives us a lesson in *C* language microprocessor programming, using state machine diagrams to build "An Open Source Keyer with Programmable Control Outputs." Here, the resulting iambic keyer (the green circuit board) is built into the author's direct conversion QRP rig.

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

HF/50MHz ALL MODE TRANSCEIVER TS-590

Nothing But Performance

The TS-590S

Kenwood has essentially redefined HF performance with the TS-590S compact HF transceiver. The TS-590S RX section sports IMD (intermodulation distortion) characteristics that are on par with those "top of the line" transceivers, not to mention having the best dynamic range in its class when handling unwanted, adjacent off-frequency signals.*

- **HF-50MHz 100W**
- **Digital IF Filters**
- **Built-in Antenna Tuner**
- **Advanced DSP from the IF stage forward**
- **500Hz and 2.7KHz roofing filters included**
- **Heavy duty TX section**

- **2 Color LCD**

KENWOOD

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

Scan with your phone to download TS-590S brochure.

www.kenwoodusa.com

JQA-1205 091
ISO9001 Registered
Communications Equipment Division
Professional Systems Business Group
JVC KENWOOD Corporation

ADS#01112

* For 1.8/3.5/7/14/21 MHz Amateur bands, when receiving in CW/FSK/SSB modes, down conversion is automatically selected if the final passband is 2.7KHz or less.

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 271

Harold Kramer, WJ1B
Publisher

Larry Wolfgang, WR1B
Editor

Lori Weinberg, KB1EIB
Assistant Editor

Zack Lau, W1VT
Ray Mack, W5IFS
Contributing Editors

Production Department

Steve Ford, WB8IMY
Publications Manager

Michelle Bloom, WB1ENT
Production Supervisor

Sue Fagan, KB1OKW
Graphic Design Supervisor

David Pingree, N1NAS
Senior Technical Illustrator

Carol Michaud, KB1QAW
Technical Illustrator

Advertising Information Contact:

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

Circulation Department

Cathy Stepina, *QEX Circulation*

Offices

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

Subscription rate for 6 issues:

In the US: 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;

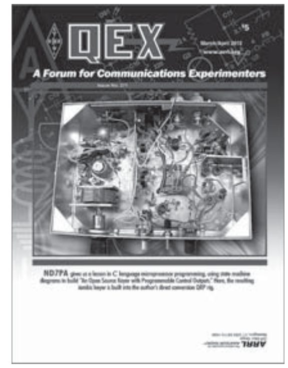
Members are asked to include their membership control number or a label from their QST when applying.

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.

March/April 2012

About the Cover

Roger Traylor, ND7PA, describes the operation of an iambic keyer with a series of state machine diagrams. Then he leads us through the steps to implement the diagrams with *C* language program sections for an ATmega48 processor to build the keyer. Roger added his keyer to a direct conversion QRP transceiver, and his design includes output controls to mute the receiver section, switch the T/R relay and key the transmitter. It even includes a sidetone output that he feeds into the receiver audio output stage.



In This Issue

Features

Index of Advertisers

The American Radio Relay League

The American Radio Relay League, Inc, is a noncommercial association of radio amateurs, organized for the promotion of interest in Amateur Radio communication and experimentation, for the establishment of networks to provide communications in the event of disasters or other emergencies, for the advancement of the radio art and of the public welfare, for the representation of the radio amateur in legislative matters, and for the maintenance of fraternalism and a high standard of conduct.

ARRL is an incorporated association without capital stock chartered under the laws of the state of Connecticut, and is an exempt organization under Section 501(c)(3) of the Internal Revenue Code of 1986. Its affairs are governed by a Board of Directors, whose voting members are elected every three years by the general membership. The officers are elected or appointed by the Directors. The League is noncommercial, and no one who could gain financially from the shaping of its affairs is eligible for membership on its Board.

"Of, by, and for the radio amateur," ARRL numbers within its ranks the vast majority of active amateurs in the nation and has a proud history of achievement as the standard-bearer in amateur affairs.

A *bona fide* interest in Amateur Radio is the only essential qualification of membership; an Amateur Radio license is not a prerequisite, although full voting membership is granted only to licensed amateurs in the US.

Membership inquiries and general correspondence should be addressed to the administrative headquarters:

ARRL
225 Main Street
Newington, CT 06111 USA
Telephone: 860-594-0200
FAX: 860-594-0259 (24-hour direct line)

Officers

President: KAY C. CRAIGIE, N3KN
570 Brush Mountain Rd, Blacksburg, VA 24060

Chief Executive Officer: DAVID SUMNER, K1ZZ

The purpose of *QEX* is to:

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

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

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

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

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

Larry Wolfgang, WR1B

Empirical Outlook

This Issue

The Jan/Feb issue of *QEX* was one of the thinnest that we have published in a long time. That is not a problem with this issue, however. As the page layouts came together, we found that we had 48 full pages, and there was not enough room to include the Letters or the Upcoming Conferences columns. Those columns will return in the May/June issue. There wasn't much new information for the Upcoming Conferences column, and most of the correspondence I had prepared for the Letters column can easily wait for the next issue.

The Letters column did not run last time, either, but there was one piece of information relating to Tom Apel's Nov/Dec 2011 article, "A New Horizontal Polarized High Gain Omni-Directional Antenna" that should have been included there. Not wanting to hold that information over again, I will pass it along to you here. Tom, K5TRA, wrote to point out that we had repeated the data from Table 1 in Table 3 of his article. The correct data for Table 3 provides the design information for 1296 MHz antenna. Here is that data:

Table 3

Helical Colinear Calculations 1296 MHz

	(Inches)	*(mm)*		
Length $\lambda/2$ =	3.20	81	0.35 λ	Elements/Turn = 3
Diameter =	2.85	72	0.31 λ	Turns = 15
Pitch =	3.46	88	0.38 λ	Total Segments = 180
Linear Total =	145.38	3693	15.94 λ	
Helix Length =	51.95	1320	5.70 λ	
Bottom =	5.91	150		
Top =	57.85	1470		

In a letter that was scheduled for this issue, Fred Brown, W6HPH, wrote to tell us that there were a couple of errors on the schematic diagram (Figure 3) of his Noise Canceller project. These were errors that occurred when we drew the schematic, and not Fred's fault. I had planned to print a corrected schematic diagram with Fred's letter, but rather than making you wait 2 more months, I will describe the problems here, and then put a PDF file of the correct schematic on the *QEX* files website. You can find the drawing at **www.arrl.org/qexfiles**, listed in the Jan/Feb 2012 section. Look for the file **1x12_Brown.pdf**. There are a pair of 1N4148 diodes connected in parallel near the noise antenna input of the circuit. Those diodes were both drawn in the same direction, but one of them has to be reversed, so that they connect anode to cathode. The bipolar transistors, Q1, Q2 and Q3 should all be shown as 2N2369A transistors. Finally, the Source lead of the 3N211 dual gate MOSFET should not connect directly to ground. That lead only connects to ground through the 220 Ω resistor and 0.01 μF capacitor combination.

In the Jan/Feb Upcoming Conferences column, we mentioned the 2012 Annual Conference of the Society of Amateur Radio Astronomers. There has been some additional information posted about that Conference on June 24-27 in Green Bank, WV. The Society has also announced their Western Regional Conference at Stanford University in California March 24 and 25, 2012. If you are interested in either of these Conferences, please visit their website at **www.radio-astronomy.org/**.

Marc Franco, N2UO, sent information about the 15th International EME conference. This year's conference will be held at Churchill College, Cambridge, UK, in August 2012. The University City of Cambridge is a world famous center of scientific learning that has played a major part in mankind's scientific advances. The conference will take place on Friday and Saturday, August 17-18, 2012. For more information, please visit the EME Conference website at: **eme2012.com**.

I believe you will find some interesting articles in this issue. We have a variety of articles planned for the coming issues, so I hope you will continue to find something of interest in every copy of *QEX* that lands in your mailbox. We are constantly looking for more good material to publish, so whether you are working on a project or have some theoretical topic you would like to explore, keep us in mind. Send your manuscripts to **qex@arrl.org**!

Michael Hightower, KF6SJ

13620 White Rock Station Rd, Poway, CA 92064; **kf6sj@arrl.net**

Simple SDR Receiver

Looking for some hardware to learn about SDR? This project may be just what you need to explore this hot topic!

This article describes the design and the theory of operation of an HF radio receiver operating in the 3.5 to 18 MHz range. The receiver architecture is based on software defined radio techniques and incorporates a Cypress PSoC CY8C3866 device that contains both analog and digital circuits, thus decreasing the receiver's component count. This part is far more than just a microcomputer; it also contains software configurable analog and digital peripherals on a single chip. Cypress calls the family a PSoC in reference to it being a programmable embedded system-on-chip. The newest series of parts, which Cypress calls the PSoC 3 family, contains a 67 MHz 8051 class microcomputer, an analog to digital converter fast enough and with enough resolution for an SDR receiver, and other valuable functions that are desirable in a receiver design.

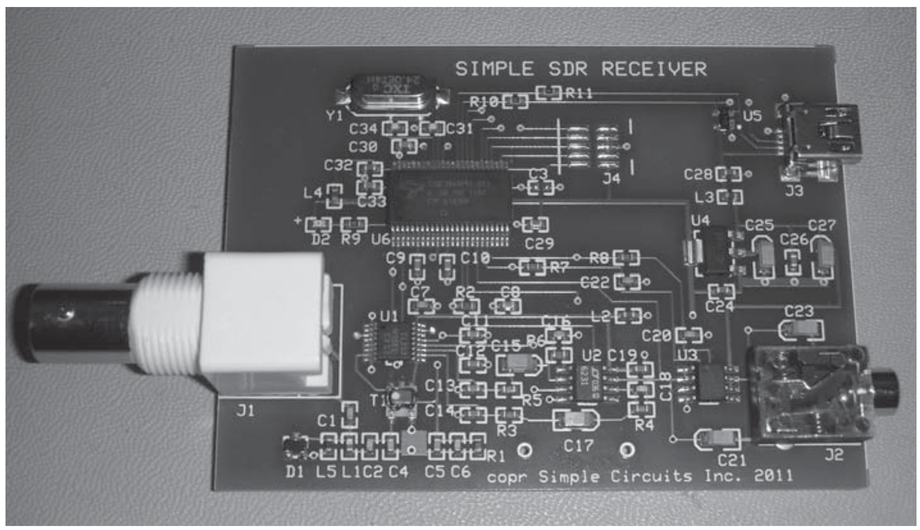


Figure 1 — The complete receiver fits on a 3.2 by 2.3 inch circuit board.

The receiver should be used for casual, conversational listening; it is not intended as a higher performance receiver for DX use. It was designed to use a minimum number of components, to be physically small and easy to operate. An LCD and controls to select the frequency and modes of operations were considered, but the design would have fewer parts and cost less if a personal computer (PC) is used for all user control. Since the PSoC has a USB port, the receiver can connect to the PC with a USB cable and take power from the PC over the USB cable, saving a power jack, and external power source. Control of the receiver is accomplished by having the receiver USB port appear as a standard com port to the PC. The *Ham Radio Deluxe* (*HRD*) program works perfectly to control this receiver. See Figure 1.

Theory of Operation

This SDR receiver is built using a quadrature sampling detector, as shown in the block diagram of Figure 2. The quadrature sampling detector is nothing more than a set of analog switches that are enabled and disabled in the particular sequence that samples the input signal four times for each cycle of the desired receive frequency. The four samples represent the 0, 90, 180, and 270° points of a sine wave. The detector output is amplified by a pair of op amp low-pass filters. After the op amps, the remaining signal processing is performed inside the PSoC microcomputer using digital processing techniques. The processing will digitize the baseband signal, remove the undesired sideband from the received signal, limit the bandwidth of the audio, and then convert the digital samples back into an analog audio signal.

At the antenna input terminals, an RF low pass filter having a 20 MHz corner frequency suppresses signals above the receiver tuning capability. Figure 3 shows the frequency response for the filter.

Referring to the schematic diagram in Figures 6 and 7, U1, a dual 1-of-4 multiplexer/demultiplexer, is used as the sampling detector. This process is similar in functionality to a local mixer, but the control is performed with digital logic switching levels.

The multiplexer is controlled with two square wave clock signals having the same frequency as the desired receive frequency. The clocks only differ in phase, one being delayed by one-fourth of the clock period. This delay represents the 90° phase shift required for an I/Q type detector.

The PSoC has an internal phase locked loop (PLL) circuit that is used to generate higher clock frequencies from a lower speed clock. The receiver makes use of this circuit to generate the sampling detector clock signals. An external 24 MHz crystal is connected to the PSoC, and is used as the reference oscillator for the PLL. The PSoC PLL only has the ability to generate frequencies that are integer multiples of the clock. An additional logic function was created and added to the programmable digital logic section of the PSoC, however, that modulates the PLL divider registers. This effectively turns the PLL into a fractional N frequency synthesizer. The "fractional" term refers to the capability of dividing the PLL feedback signal by a non-integer number, allowing for a frequency resolution as small as desired. For practical purposes, a 14 bit, sigma delta modulator constructed as a 2nd order MASH (Multi-stAge noise SHaping) was written in *Verilog*. This results in a receiver tuning frequency step size of roughly 50 Hz. When a different frequency is selected, the PSoC firmware computes a set of PLL values that will be within 50 Hz from the display frequency.

An external PLL circuit or discrete digital synthesizer chip (DDS) may generate a cleaner clock, but the advantage of using the internal PLL is obviously component cost savings.

The PLL operates in the 56 to 68 MHz range. A divider function added to the programmable digital logic section of the PSoC divides the PLL oscillator down to the desired receive frequency and shifts the phase appropriately between the two output I/Q clocks.

After the mixing process of the sampling detector, a pair of low noise op amps, U2A and U2B, amplify the base band signals. The part was chosen because of its low noise performance and the ability to operate with outputs near the ground and power rails. The op amp inputs are typically in the microvolt range. Since the op amp circuit voltage gain is on the order of 40 dB at 1 kHz, the output signals are on the order of a few hundred microvolts to a few millivolts. This circuit includes a first order low-pass filter having

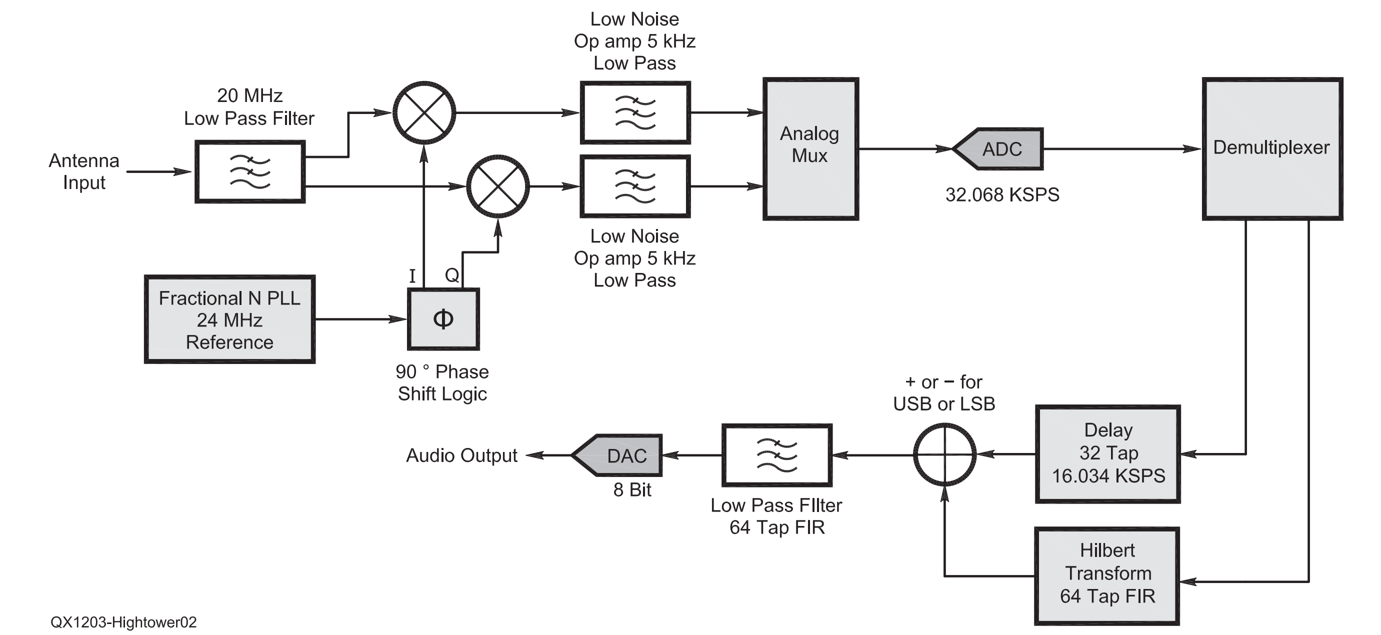


Figure 2 — Here is the Simple SDR receiver block diagram.

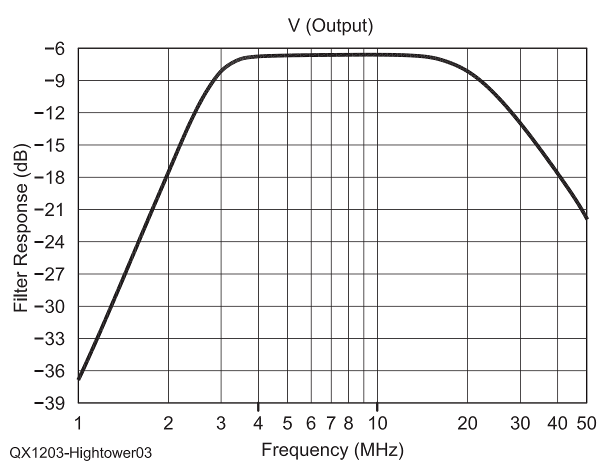


Figure 3 — This graph shows the input RF-filter response.

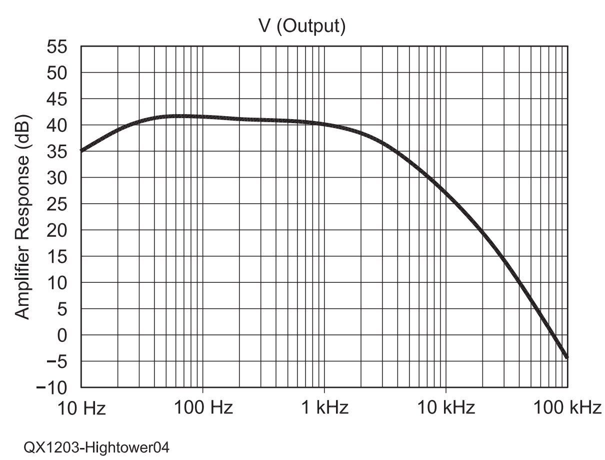


Figure 4 — The op amp circuit frequency response is shown in this graph.

a 3 kHz corner frequency. It is important to reduce the frequencies above half of the analog to digital converter sampling rate, which is called the Nyquist frequency. Otherwise, images will appear at frequencies near the sampling rate. Figure 4 shows the frequency response of the op amp circuit.

From this point, all of the signal processing is performed in the digital domain, and, specifically, inside the PSoC, U6.

The PSoC has a single analog to digital converter (ADC). Since there are two base band signals to process — the I and the Q channels — an analog multiplexer function inside the PSoC is used to switch one of the two inputs to the ADC input. It is desirable to have a high sample rate and a high number of bits, but the best tradeoff found was to use the ADC in a 14 bit mode and sampling at 32,086 samples per second. Since a sample from each input channel is necessary, the equivalent sample rate per channel is 16,043 samples per second. Therefore, the Nyquist frequency is almost 8 kHz. The op amp frequency response at 8 kHz is about 10 dB below the desired passband. This is not great, but leaves room for improvement in a future version.

The PSoC 3 family of parts has an interesting internal hardware feature they call a digital filter block, or DFB, and it consists of a 24 bit fixed point, programmable limited scope DSP engine. This is a dedicated hardware accelerator block that operates independently of the main 8051 processor. It consists of dedicated hardware than can multiply two 24 bit numbers together and add the multiplication result to a 48 bit wide accumulator register, and do this in just one system clock cycle. The process of multiplying and adding is the cornerstone of making signal filters in the digital world. This block is optimized to implement a direct form finite impulse response (FIR) filter that approaches a computation rate of one FIR tap for each clock cycle. This block is used as two independent, 64 tap digital filters.

Alternating outputs from the ADC are loaded into either a 32 sample long-delay line or one of the two digital filters. This digital filter uses a set of coefficients that form an all-pass filter having a flat magnitude response, but phase shifts all frequencies in its passband by 90°. This is called a Hilbert filter. Suppressing either the upper or lower sideband is accomplished by phase shifting the Q channel baseband data and either subtracting or adding the filter output to the delayed I channel baseband data. The delay is necessary to compensate only for the delays incurred by the processing of the Hilbert filter. The output of the addition is one of the two sidebands.

After the removal of the undesired sideband, the data stream is fed into the other half of the PSoC digital filter block configured as a low-pass filter. This filter has a steep roll off as shown in the Figure 5 for a 2 kHz filter. This is the advantage of processing in the digital domain as compared to a set of analog filters. Steep roll offs and repeatability of the filter performance over wide temperature ranges and from part to part variations are the reasons to use digital processing.

The output of the low-pass filter is fed into one of the PSoC's 8 bit digital to analog converters (DAC) that converts the data stream back into an analog signal. This signal is

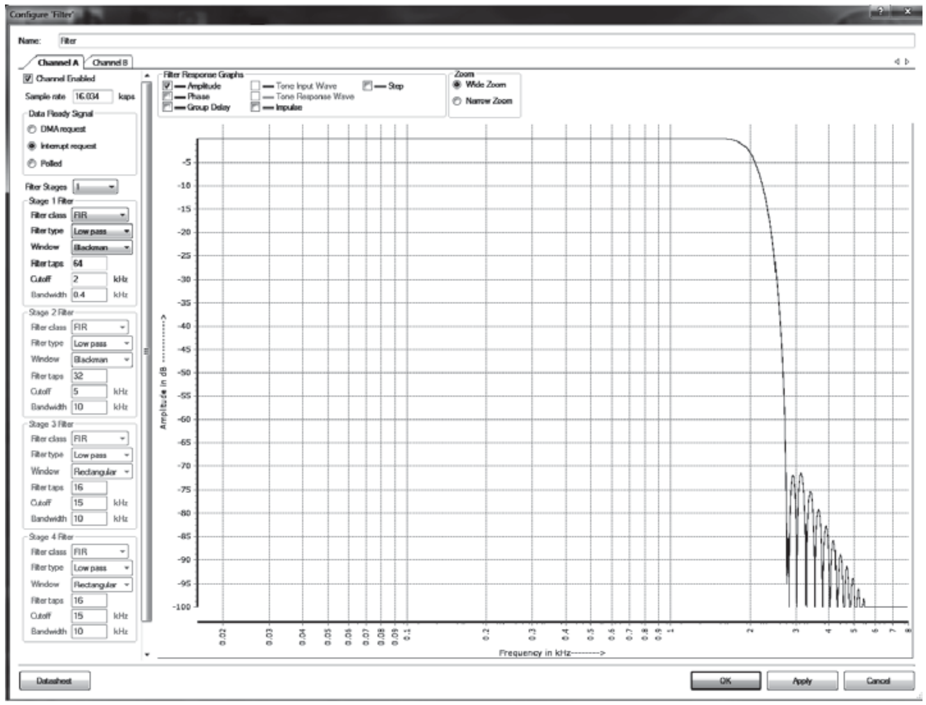


Figure 5 — The 2 kHz audio digital low-pass filter response is shown here.

Figure 6 — The Simple SDR receiver board will fit inside a Hammond Manufacturing model 1455B802 enclosure if a chassis-mount BNC connector is used instead of the on-board connector.

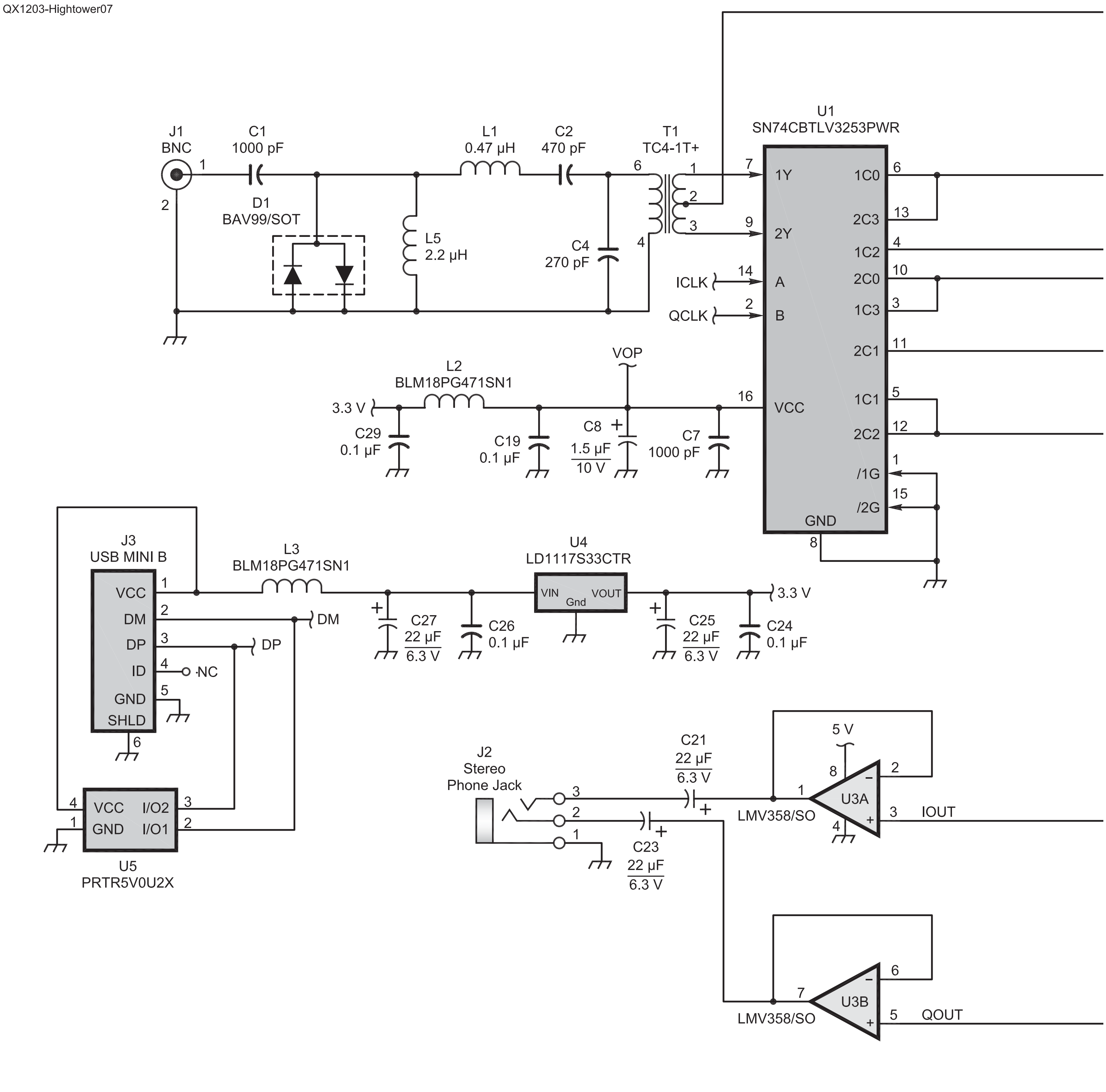
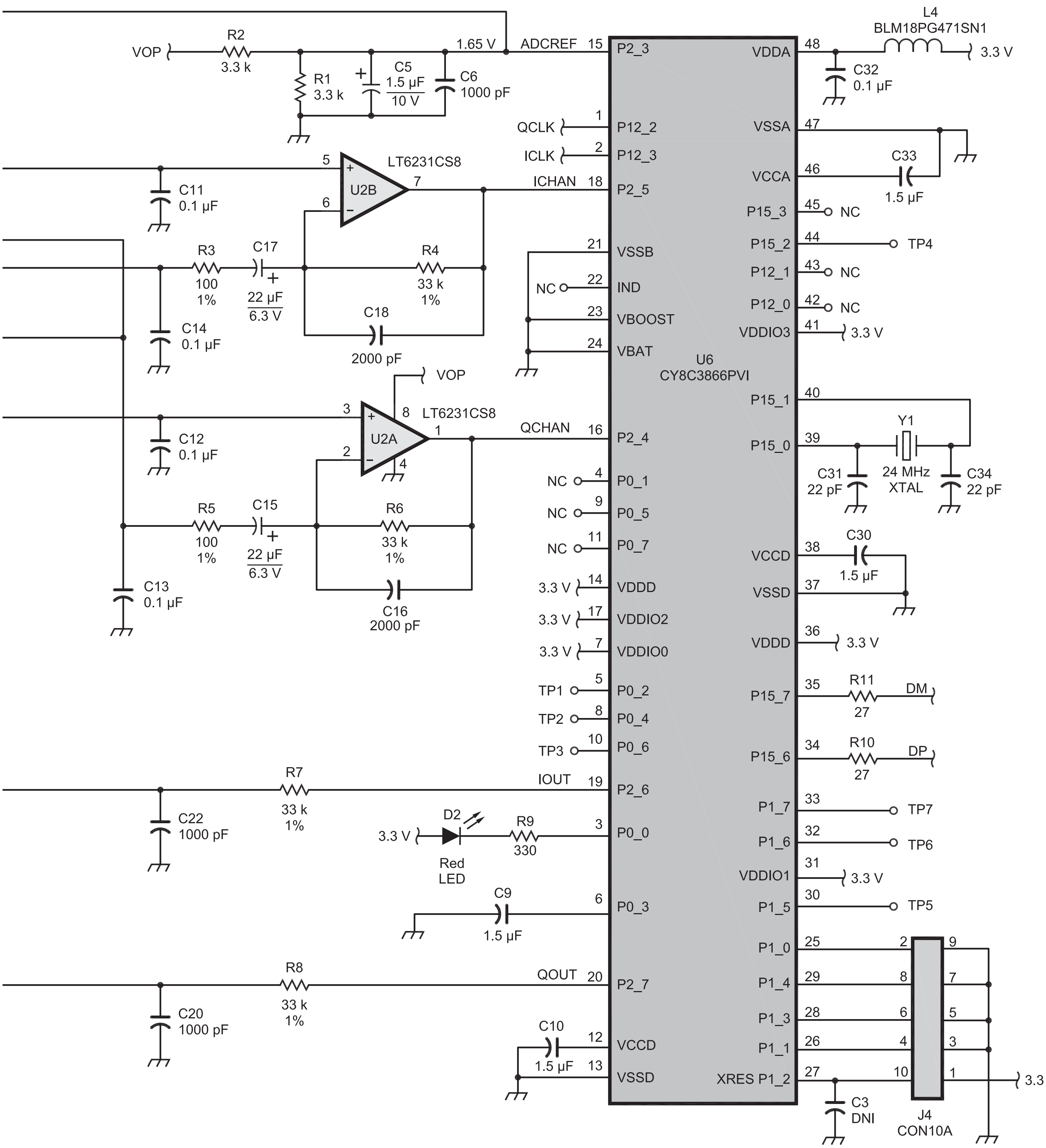


Figure 7 — Here is the schematic diagram of the Simple SDR receiver.



VOP
R2
3.3 k
R1
3.3 k
C5
1.5 µF
10 V
C6
1000 pF
1.65 V
ADCREF
15
P2_3
U6
CY8C3866PVI
L4
BLM18PG471SN1
VDDA
48
3.3 V
C32
0.1 µF
QCLK
ICLK
P12_2
P12_3
VSSA
47
LT6231CS8
U2B
ICHAN
18
P2_5
VCCA
46
C33
1.5 µF
P15_3
45
NC
C11
0.1 µF
R3
100
1%
C17
22 µF
6.3 V
R4
33 k
1%
C18
2000 pF
C14
0.1 µF
VSSB
IND
VBOOST
VBAT
P15_2
44
TP4
P12_1
43
P12_0
42
VDDIO3
41
P15_1
40
Y1
24 MHz
XTAL
P15_0
39
C31
22 pF
C34
22 pF
VOP
U2A
QCHAN
16
P2_4
C12
0.1 µF
R5
100
1%
C15
22 µF
6.3 V
R6
33 k
1%
C16
2000 pF
C13
0.1 µF
P0_1
P0_5
P0_7
VDDD
VDDIO2
VDDIO0
VCCD
38
C30
1.5 µF
VSSD
37
VDDD
36
TP1
P0_2
TP2
P0_4
TP3
P0_6
P15_7
35
R11
27
DM
P15_6
34
R10
27
DP
R7
33 k
1%
IOUT
19
P2_6
C22
1000 pF
D2
Red
LED
R9
330
P0_0
P1_7
33
TP7
P1_6
32
TP6
VDDIO1
31
C9
1.5 µF
P0_3
P1_5
30
TP5
P1_0
25
P1_4
29
R8
33 k
1%
QOUT
20
P2_7
C20
1000 pF
C10
1.5 µF
VCCD
VSSD
P1_3
28
P1_1
26
XRES P1_2
27
C3
DNI
J4
CON10A

buffered with a unity gain op amp, U3, and passed to the output connector.

Modifying and Reprogramming the Receiver

The firmware for the receiver was written and compiled within the Cypress PSoC Creator 2.0 toolset. This toolset contains the graphical design editor used to modify any of the project's internal PSoC hardware modules, the *C* compiler, and it contains tools to reprogram the PSoC flash memory with newly created firmware. This toolset is a free download from the Cypress website. (**www.cypress.com/?id=2494**)

To program a PSoC that has never been programmed, a special programming device that Cypress calls the MiniProg3 is required. Subsequent programming can be performed without using the MiniProg3, however, if the device was programmed with a firmware set that includes a special PSoC bootloader program. Then, further programming can be performed using the USB connection between the PSoC and a computer running the bootloader tool that is included in the PSoC Creator toolset.

I have made the receiver firmware available for download from the ARRL QEX files website.[1] That firmware includes the bootloader program that will communicate with and program the flash memory using the PSoC Creator bootloader tool.

For those who would like to experiment with the hardware, but prefer not to program their own devices and would rather have completed receivers, I will make assembled and programmed receiver boards available. Visit my website at **www.simplecircuits.com** for details.

Performance

The +5 V power to the receiver is supplied by the host USB device, and is typically 55 to 60 mA.

The frequency range of operation is controlled by the firmware. The current firmware limits the tuning to 3.500 MHz to 18.168 MHz. Higher frequencies are beyond the frequency range of the PSoC internal PLL circuit. Lower frequencies, such as the amateur 160 meter band, were overloaded from AM broadcast stations.

The minimum discernible signal (MDS) is approximately –117 dBm.

Installation

Before connecting the USB cable, download the file UARTUSB_CDC.zip and save it on your computer. All of the files associated with this article are available for download from the ARRL *QEX* files website. Unzip the file, put it in any folder, plug in the USB cable, and the computer will prompt you to either let it search for the driver, or allow you to specify the driver location. You will need to specify the location of where you put the file. The receiver has been tested with *Windows XP*, *Vista*, and *Windows 7*.

Note the com port number that your computer assigned to the receiver. Use that com port number in *Ham Radio Deluxe* (*HRD*). In *HRD*, add a new radio, and select Elecraft K2 as the radio type. The speed setting is not important and can be left as the default value.

Operating the Receiver

To operate the receiver, perform the following steps:

1) Connect the USB cable between a computer USB port and the receiver's USB connector, J3. Both power and control are provided by this connection. The PSoC has an internal USB full speed port. The receiver firmware implements a virtual serial communication port. When connected to a personal computer, the receiver will look like a serial com device. Using the standard CDC (communication) drivers that are built into *Windows*, the receiver can communicate with *Ham Radio Deluxe*. The receiver firmware uses the Elecraft K2 communication protocol.

2) Connect an appropriate antenna to the BNC connector, J1. The antenna should support the desired receive frequency.

3) Connect an audio amplifier or headset to the audio output connector, J2, of the receiver. This connector is a stereo connector, but both channels are fed with the same signal. Therefore, the use of either mono or stereo audio devices are acceptable. The receiver audio output will directly drive low impedance headsets. The use of good computer speakers that have a built-in amplifier and volume adjustment is ideal. There is no volume control capability in the receiver. Headsets need their own volume adjustments.

The receiver can be continuously tuned from 3.500 MHz to 18.168 MHz (80 through 17 meters).

Using *Ham Radio Deluxe* to control the radio, the receive frequency, the side band selection, and audio bandwidth can be selected.

There are four different receive filter bandwidth selections. Pressing the *HRD* Filter button produces a drop down menu with various options. FL1 is a 2.5 kHz low pass filter, FL2 is a 2.0 kHz low pass filter, FL3 is a 1.5 kHz low pass filter, and FL4 is a 1.0 kHz band pass filter. FL4 is used for CW and RTTY, and is 400 Hz wide (±200 Hz).

The attenuator button, ATT will decrease the audio volume from strong stations by several dB.

Enclosure

To keep the cost of the receiver as low as possible, the unit is not in any enclosure. The board dimensions were specifically designed to fit in a Hammond Manufacturing model 1455B802 enclosure, however. The on-board BNC connector could be replaced with a through-chassis BNC connector if you want to mount the board in a chassis.

Michael L. Hightower, KF6SJ, has been continuously licensed since 1970, and currently holds an Amateur Extra class license. Active on the HF bands, his interests include homebrew radio design and construction, and experimental communications techniques. Michael earned BSEE and MSEE degrees from the University of Illinois. He is a member of the Institute of Electrical and Electronics Engineers (IEEE), Tucson Amateur Packet Radio Corporation (TAPR) and is an ARRL Life Member.

[1]The program files for the Cypress PSoC CY8C3866 microprocessor as well as the *Windows* driver files are available for download from the ARRL *QEX* files website. Go to **www.arrl.org/qexfiles** and look for the file **3x12_Hightower.zip**.

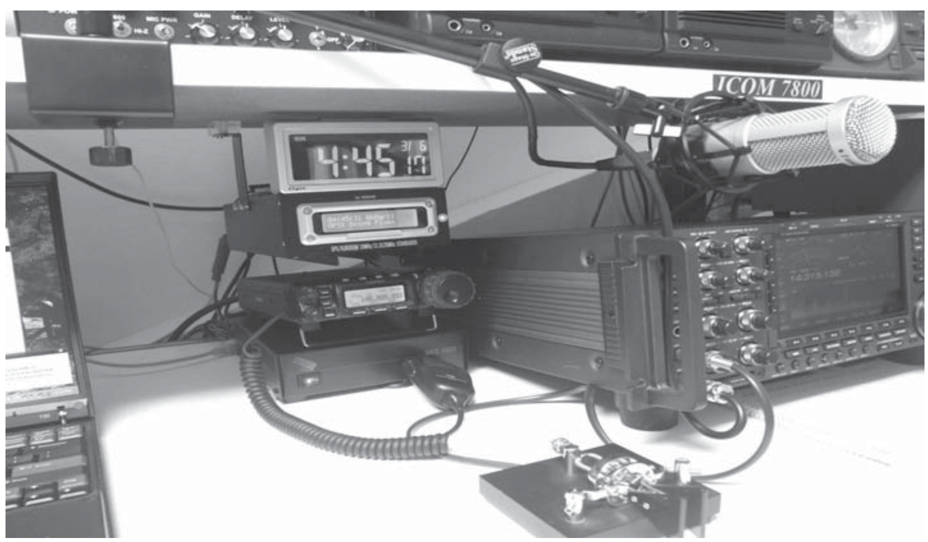
Eugeniusz A. Woloszczuk, W6EAW

494 Curie Dr, San Jose, CA 95123; **ea114w42@hotmail.com**

Stabilizing Your Transceiver Frequency Using GPS and Rubidium Reference Sources

Transceiver frequency accuracy and stability depend upon the synthesizer reference oscillator signal. The author shows how he produced better-than-designed stability for his radios.

I use either the ICOM 7800 transceiver or the Yaesu FT857D transceiver. Either way, I find that there are two problems that are annoying. First, I have to wait 15 to 30 minutes after I turn on the power for the transceiver temperature to stabilize. Second, I have to periodically calibrate the units in order to compensate for any change of crystal frequency caused by the aging of the crystal. I wanted to be free of calibration duties.



Here is a front view of the main enclosure, with the LCD Monitor for the GPS unit.

The Solution

In order to make the operation more convenient, I bought a Thunderbolt GPS disciplined clock to use as a reference oscillator for the ICOM 7800, and a rubidium programmable frequency standard FE-5680A as a reference oscillator for the Yaesu FT857D. In addition, I built a phase-locked loop (PLL) circuit to use the GPS clock to drive an Elgin 3451E alarm clock. The Elgin clock is set to UTC. It is convenient for writing down the time and date of QSOs, and it can serve as a check that the GPS is providing a correct output if its time slips relative to the GPS clock.

To protect the unit from power line outages, I am using a "Tripp-Lite" UPS with a 12 V, 22 Ah SLA battery. This will keep the circuits running for about 5 hours.

The Schematics

Figure 1 is a schematic of the Thunderbolt GPS and the components of the 10.000 MHz system. The Thunderbolt GPS disciplined clock provides a 10 MHz signal with an accuracy of about 0.01 parts per billion (ppb), thanks to frequency corrections provided by signals from eight GPS satellites orbiting overhead.

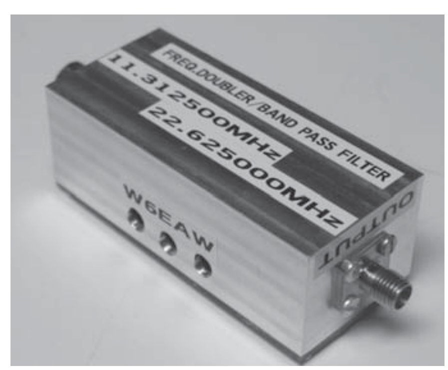


Photo A — This is an external view of the frequency doubler and band pass filter. The BNC connector connects to the 11.3125 MHz programmable Rubidium source with a coaxial cable.

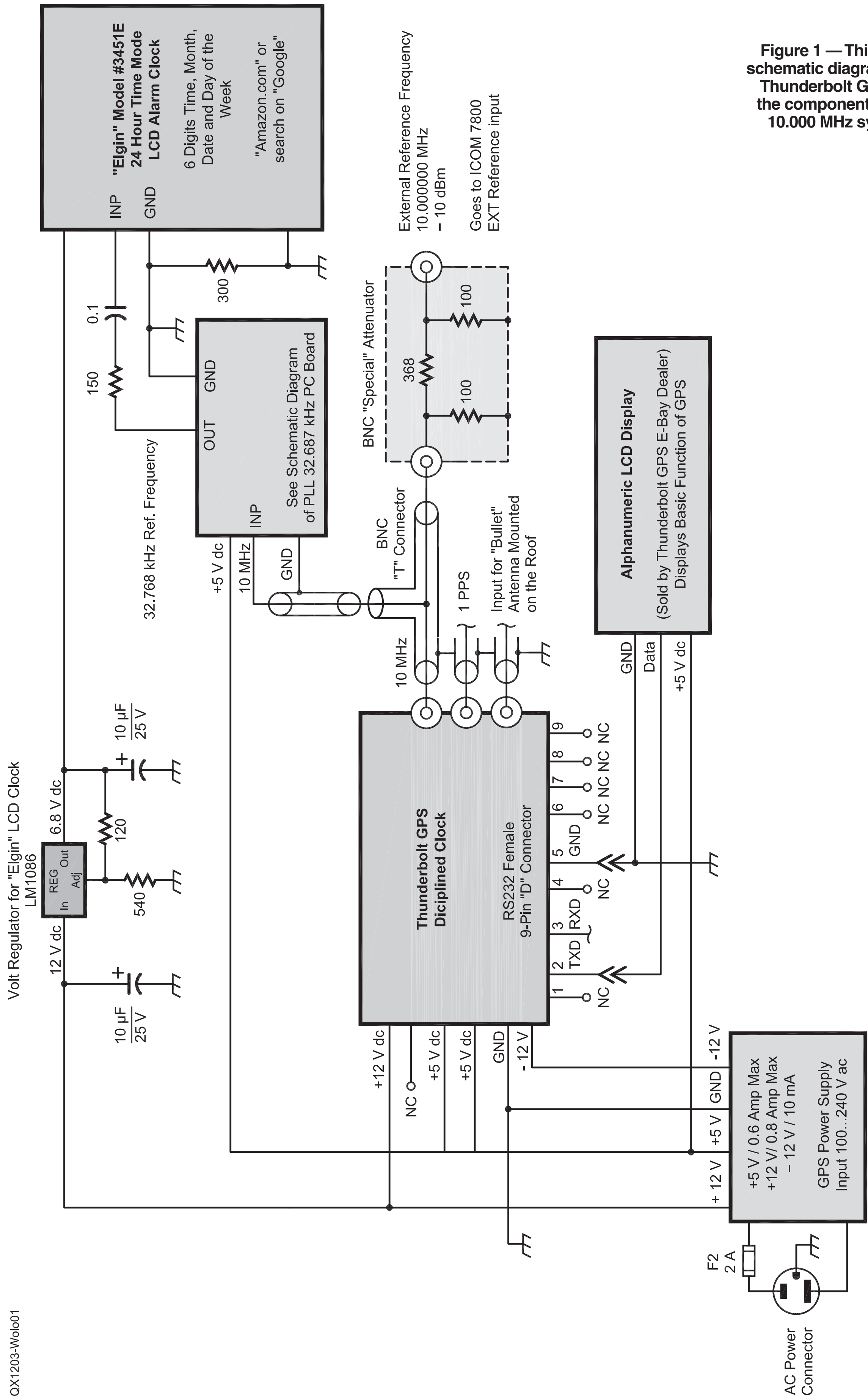


Figure 1 — This is the schematic diagram of the Thunderbolt GPS and the components of the 10.000 MHz system.

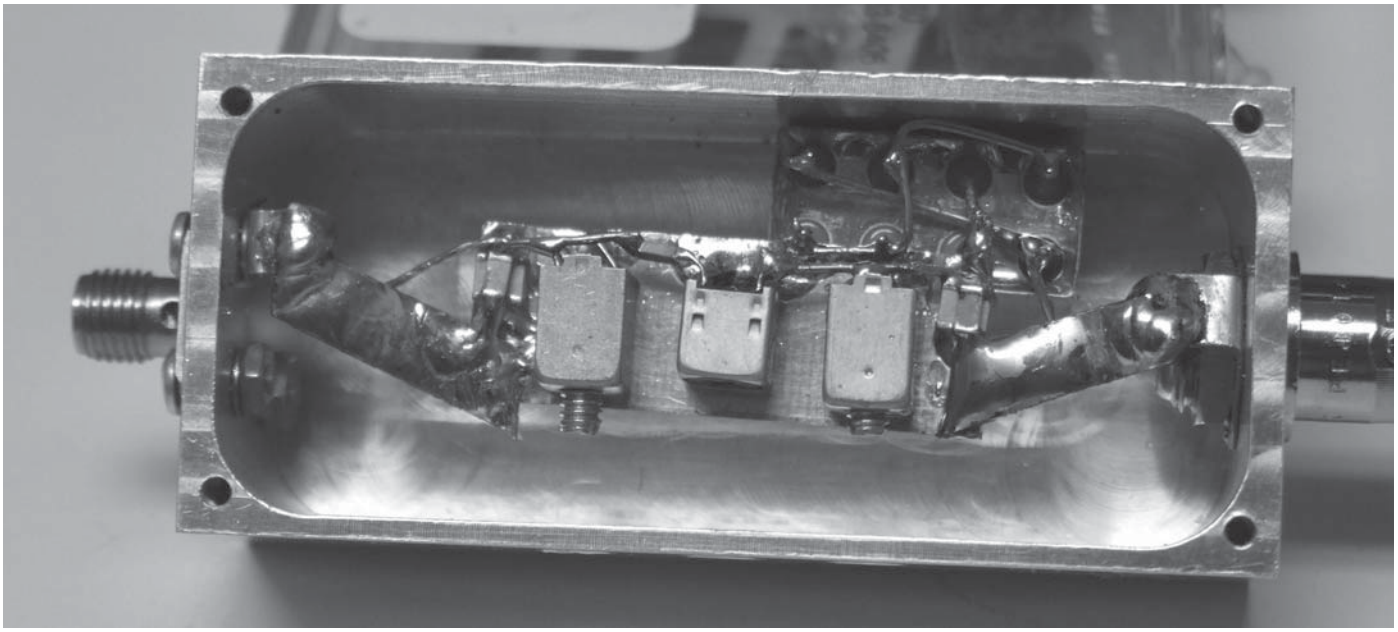


Photo B — Here you can see the internal construction of the frequency doubler and band-pass filter.

The GPS is left on all the time, consuming about 11 W. Fortunately the ICOM 7800 transceiver uses the same 10.000 MHz reference clock as the GPS provides. So, by turning off the internal oven-controlled oscillator in the ICOM 7800, switching to "external mode" and supplying a properly sized signal through a "special" attenuator to the external reference BNC connector, the ICOM 7800 will always have the proper reference clock signal as soon as it is powered up. The ICOM draws 200 W or more when operating. I use a GPS power supply to power all components. I use a "bullet" active antenna connected to the Thunderbolt GPS through a 70 Ω high quality coax.

The "special" attenuator between the GPS and the ICOM 7800 has an attenuation of 20:1 and its 80 Ω input and output impedance doesn't introduce any problem because the 50 Ω cable is very short.

An RS232 cable is used with a computer loaded with the Thunderbolt *TBOLTMON* software to see — and if necessary correct — the parameters of the GPS receiver and digital control circuits. After setting and saving the values, the computer is disconnected and the liquid crystal display is connected to the RS232 output of the Thunderbolt GPS. The display shows the key functions of the GPS system.

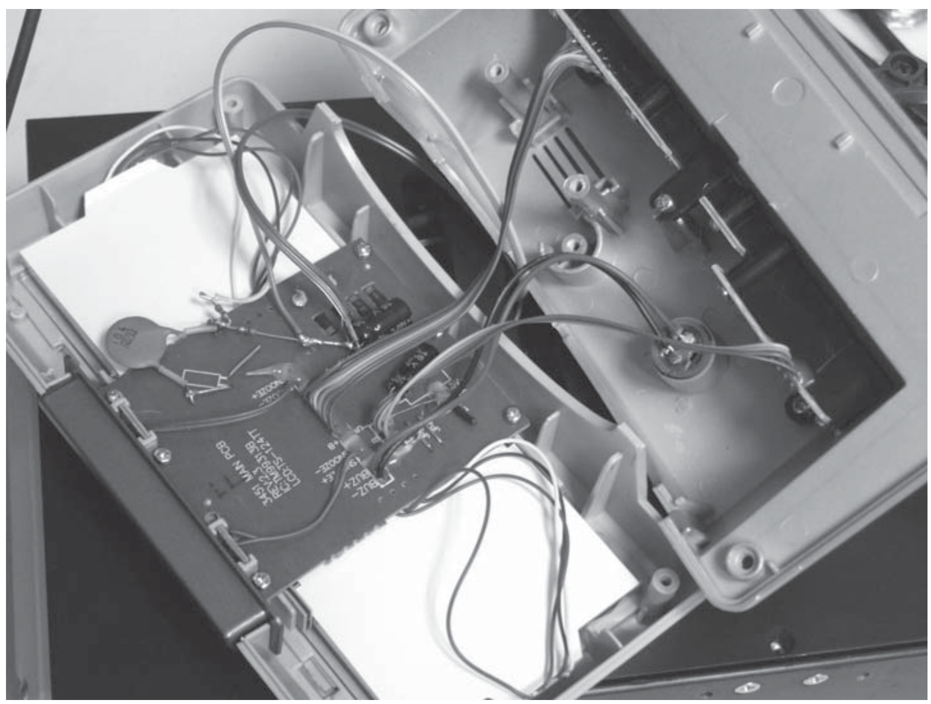


Photo C — This photo shows the wiring modifications for the Elgin Clock.

The 10.000 MHz GPS output also drives the 32.768 kHz PLL circuit, which has a 1 kΩ input impedance at 10.000 MHz. The PLL drives the Elgin 3451E alarm clock that was chosen to display the time as UTC. The Elgin 32.768 kHz reference oscillator was disconnected, and the PLL output replaces the Elgin reference oscillator. The Elgin clock is also left on all the time and consumes about 600 mW. The Elgin display is powered from the GPS power supply through an LM1086 three-pin regulator circuit. Note that the Elgin display shows the date, day of week, hours, minutes and seconds in large characters. See the lead photo.

Figure 2 is a schematic of the PLL circuit that converts the 10.000 MHz GPS signal to the 32.678 kHz signal used by the Elgin clock. The 10.000 MHz GPS signal is buffered by the circuits of Q1 and Q2. The 74HC390 chips divide the 10.000 MHz signal by $5^7 = 78125$ to give an output of 128 Hz. The 74HC393 chip divides the MC14046 output 32.768 kHz by 256 to also give an output of 128 Hz. The 74HC74 chip further divides the two 128 Hz waveforms to provide 64 Hz square waves to the MC14046 PLL. The MC14046 sends a signal to the Elgin clock.

The clock processor needs an input of

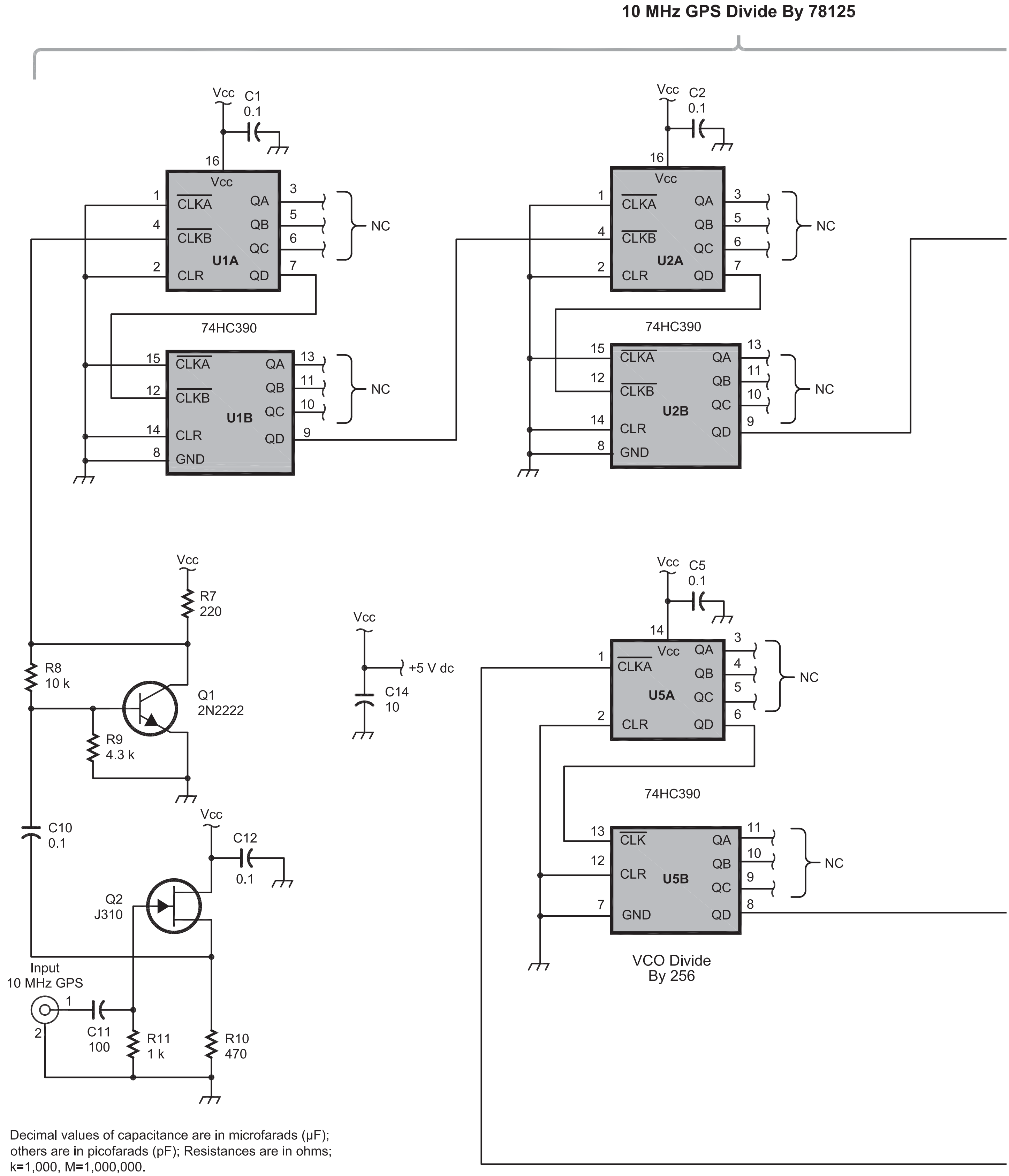
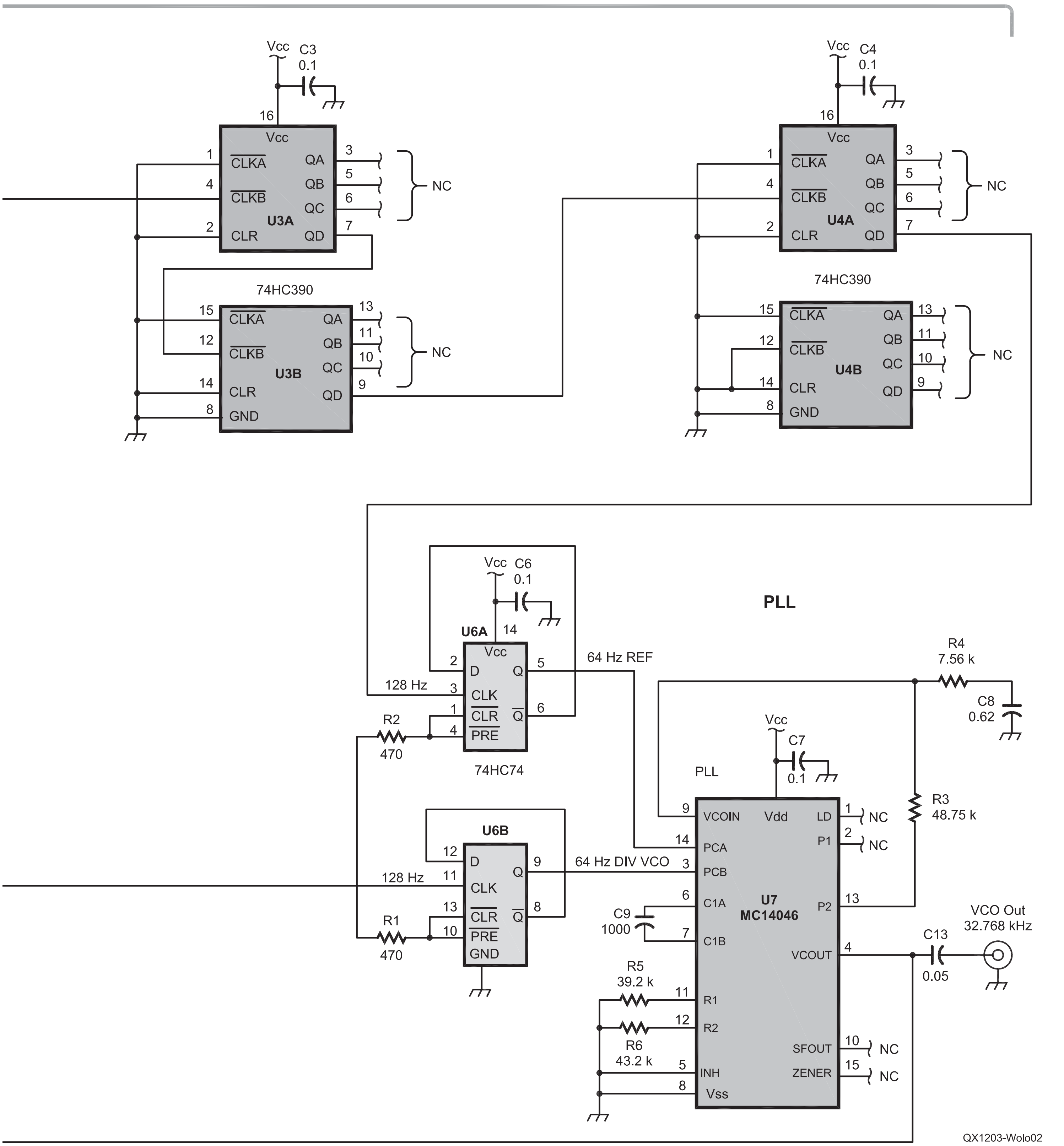


Figure 2 — This schematic diagram shows the PLL circuit that converts the 10.000 MHz GPS signal to the 32.678 kHz signal used by the Elgin clock.



Vcc
C3
0.1
16
CLKA
CLKB
CLR
QA
QB
QC
QD
U3A
NC
74HC390
U3B
GND
C4
U4A
U4B
PLL
U6A
C6
64 Hz REF
128 Hz
D
CLK
CLR
PRE
Q
R2
470
74HC74
R4
7.56 k
C8
0.62
C7
0.1
R3
48.75 k
VCOIN
Vdd
LD
P1
PCA
PCB
C1A
C1B
U7
MC14046
P2
VCOUT
U6B
64 Hz DIV VCO
R1
470
C9
1000
VCO Out
32.768 kHz
C13
0.05
R5
39.2 k
R6
43.2 k
R1
R2
INH
Vss
SFOUT
ZENER
QX1203-Wolo02

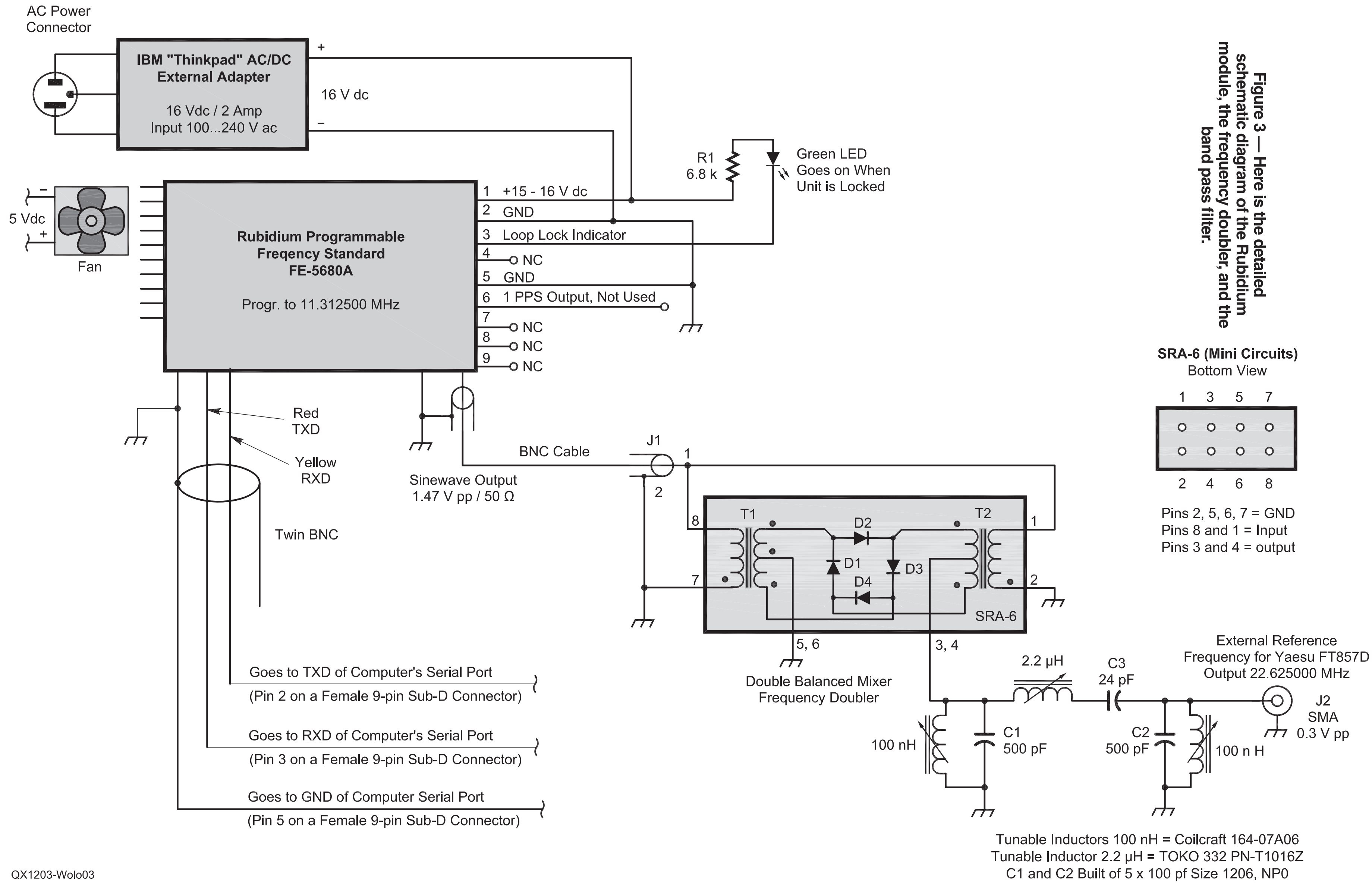


Figure 3 — Here is the detailed schematic diagram of the Rubidium module, the frequency doubler, and the band pass filter.

about 2 Vp-p. A higher level will disturb the processor. The PLL output goes through a voltage divider of 150/300 Ω and 0.1 μF to the removed crystal output. The signal on this pad was a distorted sine-wave, and this is what we get through the filter.

Figure 3 is a detailed schematic of the Rubidium module, frequency doubler, and band-pass filter.

I chose the FE-5680A Rubidium programmable frequency standard to provide the reference frequency for the Yaesu FT857D transceiver. The FE-5680A needed a rework as described by Matthias Bopp, DD1US, to connect the RF output and serial port 3. The DD1US article provides the additional step-by-step instructions for modifying the Rubidium module to provide the RF output coax and RS232 wiring. Go to **www.dd1us.de**, and click on the "Ham Downloads" button, and then click on "A rubidium frequency standard as precise reference source."

The Rubidium frequency standard also needs to be left powered up all the time to provide a stable output frequency. It consumes 11 W.

The Rubidium standard is programmable from 1 Hz to 20 MHz, but the FT857D has a reference oscillator of 22.625 MHz. So in order to allow the Rubidium clock to drive the FT857D, a frequency doubler and band-pass filter were placed between the Rubidium clock and the FT857D. The GPS may be used to check the accuracy of the Rubidium clock at any time.

I use an IBM ThinkPad ac/dc external adapter to supply power to the Rubidium clock circuits. A Resistor and LED provide a "locked" indicator. A Mini-Circuits SRA-6 frequency doubler is connected to the Rubidium clock output. The frequency doubler output goes to a custom band-pass filter tuned to 22.625 MHz. The output of the band-pass filter goes to the Yaesu 857D transceiver.

Getting the Parts and Software

It is easiest to acquire the Thunderbolt components by buying the "Thunderbolt Precision GPS 10 MHz, 1 pps Standard Easy Kit". The kit includes the *TBOLTMON* software. Search for "Thunderbolt Precision GPS 10 MHz 1 pps Standard LCD Monitor" and you can find a kit that also includes the LCD monitor for the Thunderbolt. The kit includes the Thunderbolt, Thunderbolt power supply, bullet antenna, antenna cable, RS232 cable, and LCD monitor. [I found several on-line sources for the starter kits with prices of just over $200. Some sellers include the LCD monitor for around $30. Not everyone includes a power supply. — Ed.]

The Rubidium FE-5680A can be ordered from various sellers on e-Bay. The price is around $40. The Elgin 3451E Clock can be ordered online from many vendors. Just do a search for Elgin 3451E. An alternate clock is the Advance 3451. The 16 V ac/dc adapter can be found by searching for "ac/dc adapters 16 V dc".

The ICs and passive components and connectors can be obtained from Mouser Electronics. The Mini-Circuits SRA-6 may have to be replaced with an SBL-1 (about $8) Mini-Circuits is the vendor I used.

The PLL circuit was built on a professional double-sided circuit board. See Figure 4. I ordered 10 pieces and paid $140 total. I still have 6 pieces, which I can sell below cost, at $10.00, to any interested ham, first come, first served.

Although some of the above parts were

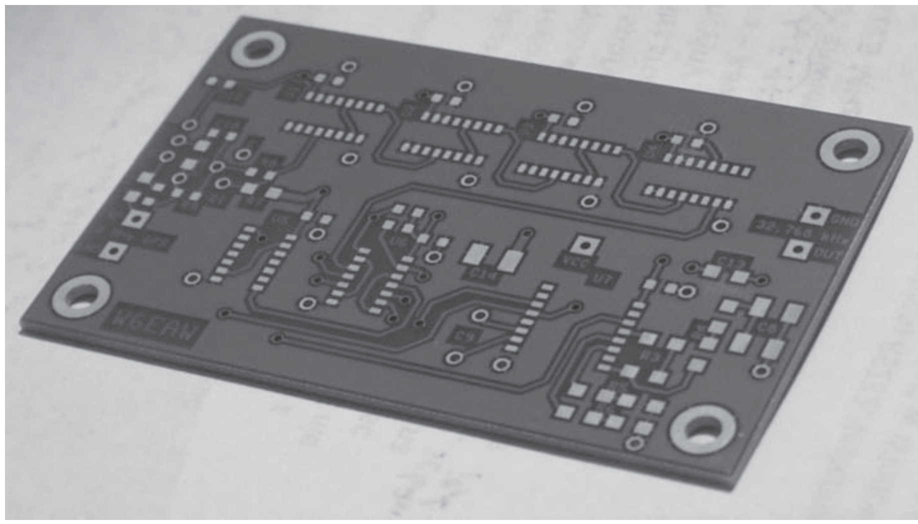


Figure 4 — This photo shows the professional two sided, plated-through-hole circuit board that was designed and fabricated for the PLL.

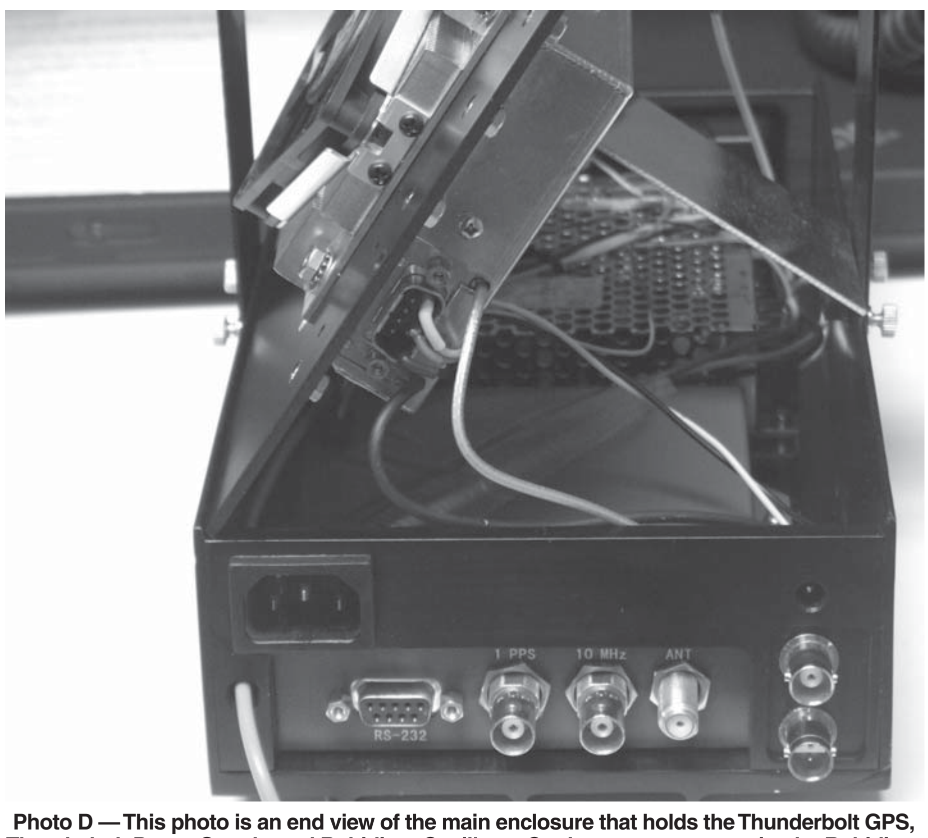


Photo D — This photo is an end view of the main enclosure that holds the Thunderbolt GPS, Thunderbolt Power Supply, and Rubidium Oscillator. On the top cover opposite the Rubidium Oscillator is the heat sink and fan.

previously used, I have not had any problems, and appreciate the much lower prices of the used equipment.

Building the System

There really isn't much to tell about putting all the pieces together. You may have different equipment or different ways of packaging it all. A series of photos show my pieces from various angles.

Within the FT-857D, the original oscillator was easily removed from the pins. A thin piece of coax is connected to the original oscillator pins. The other end of the thin coax has a male SMA connector, and goes outside of the FT-857D enclosure, where it is connected to the female SMA connector on the aluminum box holding the frequency doubler and band-pass filter. (The frequency doubler and band-pass filter circuits are shown as part of Figure 1.) These details will differ if you use a different transceiver than the FT-857D. Photos A and B show the frequency doubler and band-pass filter box.

To facilitate de-soldering the crystal and soldering the wires from the PLL, we need to unscrew the Elgin circuit board from the LCD. See Photo C. There is no risk of losing contact with the LCD because the contacting "rubber" block is broad and well guided during re-assembly.

The PLL output signal comes in with the wires shown at the top left of Photo C, and goes to the RC network on the circuit board. The capacitor output of the RC network goes to the pad of the removed crystal that goes to the microprocessor.

I found this pad on the unmodified working alarm clock with the help of an oscilloscope and probe before removing the 32.687 kHz crystal. The pad going to the microprocessor has a distorted signal, while the other crystal pad has a pure sine wave.

I didn't have much confidence in the miniature power line transformer that came with the Elgin clock, so I removed it and the power line cable. The LM1086 regulator and components are located on the top of the GPS power supply. The 6.8 V and ground connections come in to the clock on a pair of wires just below the PLL wires on Photo C.

Photos D, E, F and G show how the various pieces connect in the main chassis that I used. That chassis holds the Thunderbolt GPS and its power supply and the Rubidium module. There is a heat sink and fan bolted to the top cover.

Photo H shows the external connections to the main enclosure. The 10 MHz signal from the Thunderbolt goes to the ICOM 7800 transceiver, through a combination of BNC and SMA adapters to form a type T adapter mounted to the Thunderbolt output. I used a short length of coax to reach the "special attenuator" shown on Figure 1 which produces the required reference signal level for the IC-7800. The attenuator connects to the back of the transceiver. The coaxial line off the top of the T goes through the SMA connector and back inside the main enclosure to the PLL circuit board (not shown)

On the top right of the enclosure shown in Photo H is the connector for the 16 V dc power input. Below that is the Rubidium 11.3125 MHz output coax. Below that is the twin BNC connector for the serial input for programming the Rubidium reference source.

On the left is the RS232 connector, shown with the cable connecting to the front display. This will be removed and connected to a computer when programming the GPS.

Photo I is a close-up view of the front panel, with the GPS status display.

The lead photo shows the whole working system. At the top center is the Elgin clock set to UTC. Below that is the main enclo-

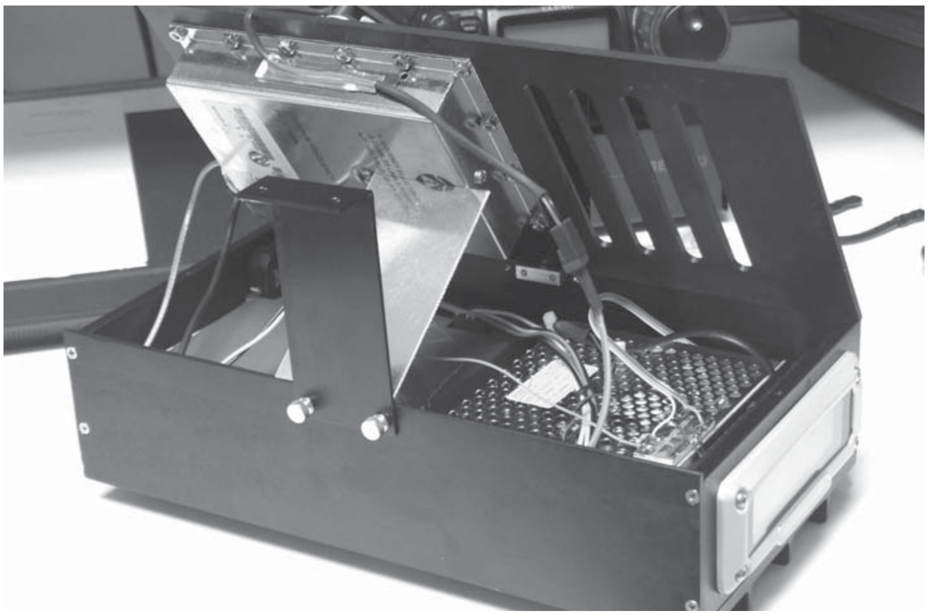


Photo E — Here is an open view of the main enclosure. The GPS power supply is on the bottom right, and the GPS display is on the outside right end. The Rubidium module is under the top cover.

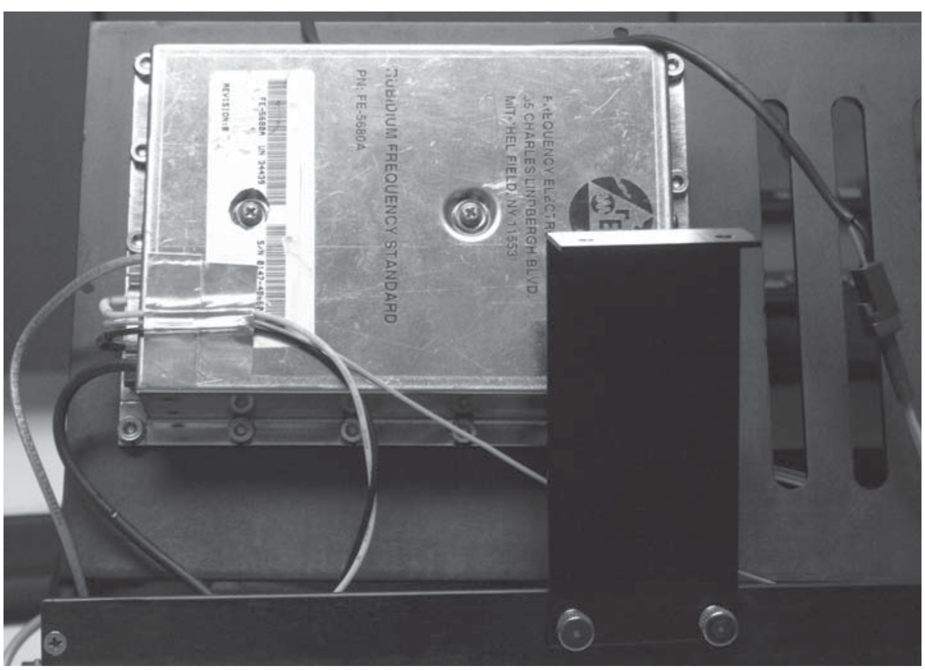


Photo F — Here is a close-up view of the Rubidium module mounted to the chassis lid. You can see the external wiring to the Rubidium module. These wires are from the modification described in the article by DD1US.

sure with the GPS status display. All of the "messy" details are located within the main enclosure, and the Elgin clock. The 16 V dc power adapter is plugged into a power strip on the floor.

The "bullet" GPS antenna is not shown, but needs to be mounted outside in an area where it has a wide view of the sky so that it can receive constant GPS satellite signals. The best place is on the roof. The GPS manual has details on this.

Configuring the GPS

First the RS232 connector to the Thunderbolt GPS is removed from the back of the main enclosure, and an RS232 cable is connected from a PC to that jack. Load the PC with the program *TBOLTMON* supplied with the GPS kit.

1) Power-up the unit, enter the *TBOLTMON* program, then select serial port (usually COM1).

2) Enter the next page, which shows all of the parameters of the GPS system, alarms, satellite signal levels, date and GPS time.

3) Enter "Setup" on the top left corner of the page, where you can configure the parameters of the GPS receiver, like elevation mask, signal level mask and so on. This is done to improve the accuracy of disciplining by relying on the stronger signals from the closer satellites.

4) There are more advanced set-up capabilities explained in "Thunderbolt GPS Disciplined Clock Manual," version 3.0. You can find this pdf file with a Google search. The manual also describes details and pictures about the mounting of the "bullet" antenna.

5) Don't forget to save the settings. Without "save," the settings will return to factory default .

6) Disconnect the RS232 cable to the PC, and re-connect the RS232 cable to the front display.

Programming the Rubidium FE-5680A Module

Here we have to thank Matthias Bopp, DD1US, for instructions in programming the FE5680A. To access these instructions, go to his website, **www.dd1us.de** and look under "Ham Downloads". There you will find an article titled: "A rubidium frequency standard as a precise reference source for the ham radio station," which has his instructions. Programming the FE5680A is easy.

First, I needed to calibrate the FE5680A using the 10.000 MHz GPS standard. To do this, I needed to set the Rubidium clock to 10.000 MHz. After connecting the serial port to the computer, power up the unit and let the temperature stabilize for 2 hours. Then:

1) Enter *Hyper Terminal*. In *Windows*

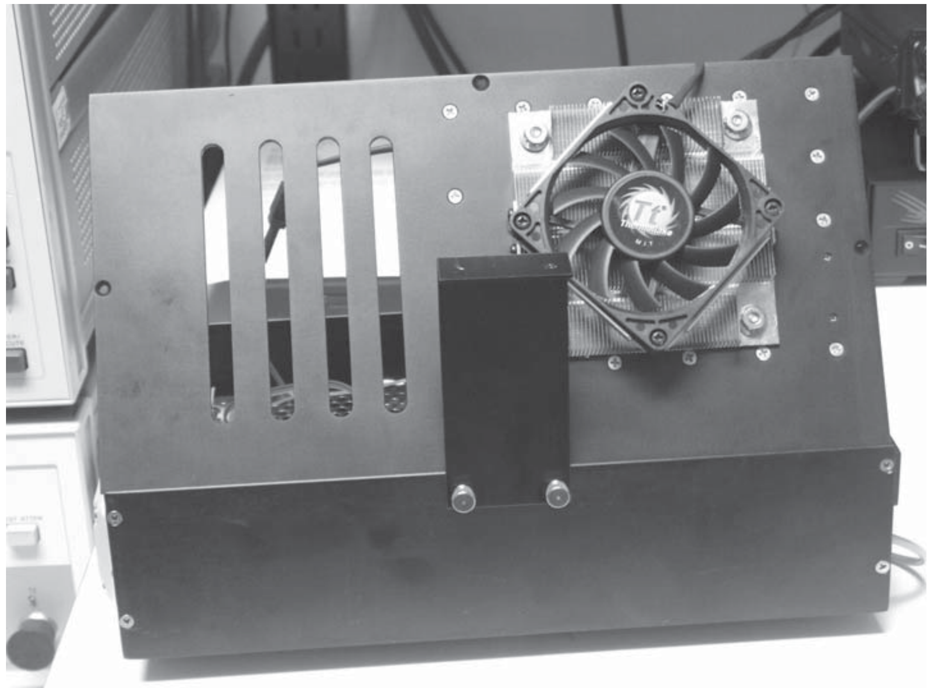


Photo G — This photo shows how the heat sink and fan are mounted on the lid of the main enclosure.

Photo H — This is the back panel of the main enclosure. Notice the T adapter on the 10 MHz output from the Thunderbolt GPS unit. The coax on the top of the T goes back inside the enclosure, to the PLL circuit board. The straight end of the T accepts the coax that carries the 10 MHz reference signal to the "special attenuator" (see Figure 1) and then to the ICOM IC-7800 transceiver.

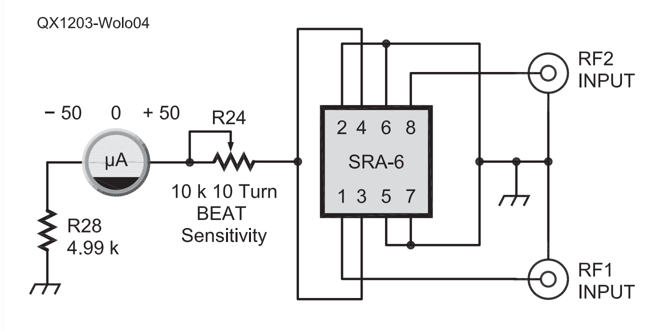


Figure 5 — This frequency/phase meter schematic diagram was described by Bob Miller, KE6F in the Sep/Oct 2009 issue of *QEX*. I made one modification, using the 50 µA center needle meter.

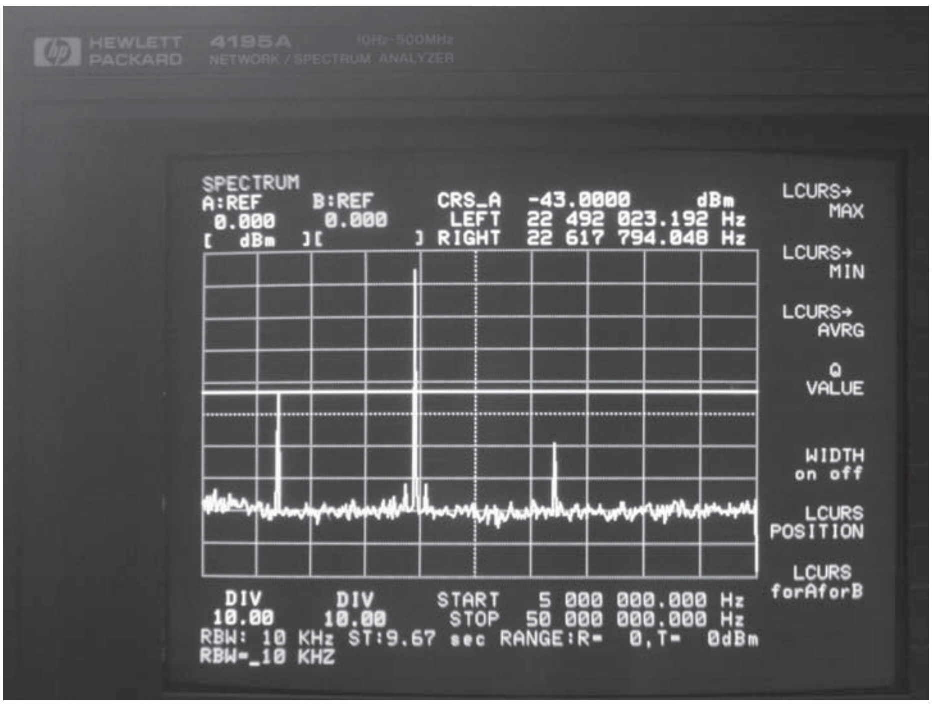


Figure 6 — This spectrum analyzer display shows the spectrum of the 22.625 MHz signal from the frequency doubler circuit. This signal is used as the synthesizer reference signal for my Yaesu FT-587D transceiver.

XP go to All Programs, Accessories, Communications , *Hyper Terminal*.

2) Open a COM port (usually COM1). You need to talk to the port that is connected to the Rubidium module.

3) Set 9600 baud, 8 bits, 1 stop bit, no parity control.

4) Set to output CR and LF characters when ENTER is pressed.

4) Now follow the instructions from Matthias' article:

4.1) When you press S and Enter, you should receive a message like "R = 50255057.012932 Hz F = 2ABB504000000000" (Your numbers may be slightly different.)

4.2) The command structure for setting the frequency is F = abcdefgh (ENTER). The combination "abcdefgh" is a 4 byte hexadecimal word, such as 31AB56DF (base 16).

4.3) The output frequency obtained by programming is $F_{out} = N / 2^{32} \times F_{ref}$ where: N is your 4 byte hex word, F_{ref} is a reference frequency of approximately 50.255 MHz (the "R" frequency)

4.4) The reference frequency is given by the R = xxxxxxxx response to the "S" command you entered in step 4.1 above. You can also measure the reference frequency yourself, if you have a well calibrated counter. The Reference frequency is available at a test point at the top of the DDS board.

4.5) There is an "E" command that stores the frequency settings in a non volatile memory in the unit. It works fine and avoids the need to reprogram the unit after each power up.

5) In my case, I entered F = 32F0AD99.304BFD3A9 and received the intended output frequency, 10.0 MHz. I saved it by entering the "E" command.

6) In checking the frequency versus the GPS, I found an error of about 0.06 Hz. I used my frequency/phase detector meter to make that measurement. See Figure 5.

The frequency/phase detector shown in Figure 5 was described in the Sep/Oct 2009 issue of *QEX* by Bob Miller, KE6F, in his article "Atomic Frequency Reference for Your Shack."

I decided to improve the accuracy of seeing the calibration process. Instead of using a regular 100 µA meter, I used a 50 µA meter that has "zero" at the center. This modification has a big advantage over my previous method, when comparing two signals that are very close in frequency.

When the frequencies are nearly equal, we can see the phase changes on the meter, and when we tune two frequencies so the meter needle remains at the center "zero" position, we can be assured that both signals are absolutely identical. It is a very simple and inexpensive method!

7) This difference (found in step 6) can be reduced by adjusting the potentiometer, which is accessible via a small hole for a screwdriver on the side of the top cover of the Rubidium FE-5869A module. This small hole is visible in Photo E between the heatsink fan power cable and one of the mounting screws of the FE-5680A to the top cover of the main enclosure. This potentiometer controls the level of a weak magnetic field at the rubidium lamp. This field has a limited effect on resonance frequency of the "atomic" module and allows one to adjust the output frequency of the rubidium standard slightly.

8) Finally, when I was satisfied that the 10 MHz of the Rubidium module was close enough to zero error, and my µA meter needle on the frequency/phase detector was slowly moving around the center, I re-programmed the Rubidium module to 11.3125 MHz by:

8.1) Setting F = 39A04462.183C5DDB. After frequency doubling, I had the necessary 22.625000 MHz, which is the reference signal for the FT-857D transceiver.

8.2) Then I entered the "E" command to save to program.

Results

Figure 6 shows the spectrum of the 22.625 MHz signal at the output of the filter. The level is –5 dBm, as required by the FT-587D. The 11.3125 MHz component is down to –43 dBm, and the level of the third harmonic is down to –58 dBm. The FT-857D has further internal filtering This result is no worse than the spectrum of original reference signal.

The stability and accuracy of the Rubidium standard allows it to be used as a reference frequency source for very narrow band work at very high frequencies. That was the basic objective of Matthias, DD1US, who deals with amateur satellite communications and other things.

The Rubidium oscillator, due to its atomic reference, has kept the correct frequency for more than 8 months so far. Also, after more than 8 months, I have not seen a slip of the Elgin clock as compared to the GPS clock.

A Few Notes

PLL circuits can be difficult to understand. Design equations are given, but that doesn't make them easier to understand. The basic problem is that the phase comparator acts like a classical integrator when presented with two signals that are nearly the same frequency and phase. So the secret to proper operation is to ensure that the applied frequencies are nearly the same, and that the loop filter and closed loop gain are stable when the loop is closed. Otherwise the loop is erratic or unstable. To accomplish this confusing task, I took the easy approach: I used the stable loop solution provided in the data sheet, and just scaled the loop components to my 64 Hz operating frequency.

When powered up, the VCO is not near the correct frequency, and it takes a second or so for the loop to lock. But once locked, the loop is stable, and counts according to the applied GPS input. After that the PLL is left on all the time.

Setting the Clock

GPS time and UTC run with the same clock rate, but have different times. They both use similar atomic clocks, but UTC is "shifted" once in a while so that it tells Earth time. The GPS time is not shifted. The earth rotation is gradually slowing down, and now UTC is 15 seconds later than GPS time, due to a second being removed from the standard UTC clock every time it is necessary.

The Elgin clock has a button that when held down stops the clock, and when released lets the clock run. So to set the Elgin clock to UTC, preset the Elgin to say 10:00:00, and hold the button in until the GPS clock shown on the Thunderbolt says 10:00:15. Then release the button. After this we expect the 15 second difference to remain fixed until it is reset again, or the GPS skips, or the PLL-Elgin system slips.

I hope this article will arouse interest and may result in even better designs. I will be happy to answer any questions about the project.

I thank Ted Rees for helping me write this article.

Photo I — Here is a front view of the main enclosure, with the LCD Monitor for the GPS unit.

Eugeniusz Adam Woloszczuk, W6EAW, former SP7/SP8BJI and KG6TED, is a retired engineer. Eugene began his Amateur Radio "life" as a 10 year old boy in a very unfriendly communist regime in Poland. It was very difficult to obtain any components, and many of the resistors and capacitors he used in his projects were home made. Tubes from old radios (from before WWII) were a real treasure. Everything had to be done in secret in those days. At the age of 16 he built his first (illegal) amplitude modulated transmitter on 40 meters using 6L6 tubes. He found some friends in his city who were also experimenting with Amateur Radio. Eventually the regime softened and began to issue Amateur Radio licenses.

After high school, he passed the CW and technical exams, and obtained his first license — SP8BJI. He began working in the electronics industry in Warsaw, and continued to design and build Amateur Radio equipment. He published many articles about SSB exciters, crystal filters and analog and digital synthesizers in the Polish magazine, RADIOAMATOR.

Later, Eugene moved to SP7, was married and became an electronic specialist working in the ceramic industry, building complete automation systems for high power hydraulic presses.

In 1982, as an active Solidarity organizer, he was forced to leave the country. From a refugee camp in Austria he was sponsored to immigrate to the US, and moved to Sunnyvale, CA. In the US, he worked as an engineering associate for Verbatim Corporation, in the magneto-optical group. He also worked for LASERDRIVE Ltd and MAXOPTIX Corp as an RF engineer, until moving on to INTERSIL Corp as a Staff Engineer, where he built a range of unique test equipment used with optical storage systems, optical sensors, projectors and other systems.

Eugene holds two US patents, with a third pending. Now retired, he enjoys plenty of time to continue his hobby "for life." He has many new projects to be finished, and has a well equipped home RF lab. Eugene earned his General class license in 2003 and his Amateur Extra in 2008.

References

Yaesu FT857D: **http://w4ti.net/yaesu/ft857/ft857_Technical_Supplement.pdf**

ICOM 7800: **www.scribd.com/doc/37295683/Icom-IC-7800-Service-Manual**

Thunderbolt GPS: **www.dpie.com/gps/thunderbolt_e.html**

Thunderbolt Manual: **ftp://ftp.trimble.com/pub/sct/embedded/bin/Manuals/Old%20Manuals/Thunderbolt.pdf**

Rubidium FE-5680: **www.vk3um.com/Reference%20Data/FE5680A_Opt2_TechManRevised%202.pdf**

Elgin 3451E: **www.4alarmclocks.com/el34elalcl.html**

GPS Easy Kit: **http://cgi.ebay.com/Thunderbolt-PRECISION-GPS-10Mhz-1PPS-Standard-Easy-Kit-/270515147719**

LM1086 spec: **www.national.com/ds/LM/LM1086.pdf**

MC14046: **www.datasheetcatalog.org/datasheet/motorola/MC14046BCP.pdf**

74HC390: **http://focus.ti.com/lit/ds/symlink/cd74hc390.pdf**

74HC393: **www.nxp.com/documents/data_sheet/74HC_HCT393.pdf**

74HC74: **www.fairchildsemi.com/ds/74/74VHC74.pdf**

J310: **www.onsemi.com/pub_link/Collateral/J309-D.PDF**

QEX

Bob Simmons, WB6EYV

1268 Veronica Springs Rd, Santa Barbara, CA 93105; **pelican2@silicon.com**

A Simple Internet VoIP Board

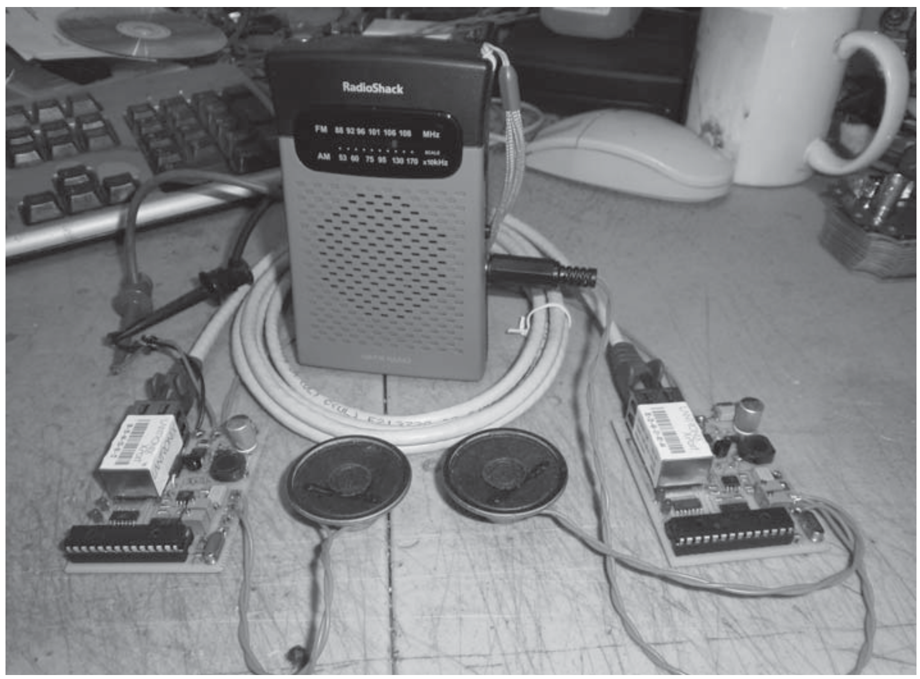
A pair of these boards will provide a full duplex Internet audio link

This article describes a full duplex Internet audio board (a "VoIP" board) using a PIC® microcomputer and a Lantronix XPORT Ethernet module. This board sends and receives (approximately) 16 user datagram protocol (UDP) packets per second, with 512 payload bytes per packet. Each payload byte expresses an 8 bit audio sample, taken 8000 times per second.

The message format of 8 bits per sample and 8000 samples per second was as a matter of convenience and simplicity, and is *not* compatible with any common VoIP standard. Indeed, "VoIP" is actually a "buzz word" and does not really identify any "formal" internet protocol. Several audio formats are commonly used on the internet, but none seem to be predominant.

Two boards will provide a basic (two way) point-to-point Internet audio system, like an intercom or telephone. For bench tests, two boards can communicate directly with each other using an Ethernet crossover cable. The lead photo shows my test setup. (There is more about this later.) No provisions are made for broadcasts to multiple listeners. The IP addresses and ports of both locations must be identified and programmed into both XPORT modules, at the time of installation. The intention here is to describe a minimal system that can provide a foundation for more sophisticated internet-linked devices.

Details are provided here for construction, but achieving an Internet link will also require knowledge of local area network (LAN) principles, (configuring routers, switchers and so on) to provide a clear path to the Internet for each module. In general, a static IP address for the gateway computer will also be required, at both ends of the link. These topics are beyond the scope of this article and every LAN situation is different, so if you are not familiar with networking, consider enlisting the help of a friend with LAN experience.



The author's test-bench setup for testing a pair of the VoIP boards.

The Lantronix XPORT Module

The Lantronix XPORT module looks like a large RJ45 jack, but it actually contains its own microcomputer, which provides all the "intelligence" required for the Internet link. The module is quite flexible, and can be configured using a utility "Device Installer" program from Lantronix.

Interface with the PIC microcomputer is done with a full duplex serial port, similar to an RS232 port. The Lantronix port uses 3.3 V logic signal levels and positive logic polarity, but genuine RS232 signals are bipolar (3 to 15 V) and use negative logic polarity.

The VoIP Board

The PIC microcomputer is a socket-mounted 28 pin DIP device that can be re-programmed with a PicStart Plus programmer. The board has an LM386 (receive) speaker amplifier, and an op amp microphone amplifier for transmit. Trimpots are provided to adjust audio levels for both transmit and receive. See Figure 1. The schematic diagram for the VoIP board is shown in Figure 2.

The XPORT module requires 3.3 V dc for power, provided here by a switching regulator for good efficiency and reduced heating. The PIC uses a separate 5 V dc regulator. The supply load ranges from 1.6 W (12.6 V dc, idle/silent) to nearly 5 W with full speaker volume. Digital inputs to the XPORT module are tolerant of 5 V PIC output signals, but a 74CX07 chip is employed to translate 3.3 V XPORT output signals to 5 V levels for the benefit of the PIC inputs.

The circuit board is a double-sided board. Figure 3 is the top of the board and Figure 4 is the bottom pattern. I used the ExpressPCB CAD tool to lay out the circuit board. I created "two up" artwork to create a pair of boards, but only one half of the artwork is shown here. Figure 5 is the parts placement overlay.

The PIC Software

The PIC assembly code is fairly simple, yielding about 250 program bytes. TMR0 is configured to generate interrupts 8000 times per second, crystal controlled. The PIC UART communicates to and from the XPORT module at 115.2 K baud. Microphone audio is digitized by the 10-bit ADC, (the bottom 2 bits are ignored) and the PIC pulse width modulation (PWM) module, operating at 78.25 kHz, is used to generate PWM speaker audio. The ADC input is dc biased at mid supply (2.50 V) to allow for both positive and negative excursions of the microphone signal. As a result, audio "silence" will be expressed with a byte value of 0×80 (which is mid supply).

Interrupts are generated by TMR0 and the UART receiver. The program is 100% "interrupt driven," and there is no "main program loop."

Audio packets (UDP messages) are sent at an average rate of 15.625 packets/second, with 512 "payload" bytes of (8 bit) audio in each packet. Individual transmit audio bytes are sent to the Lantronix module immediately after they are measured with the ADC. The XPORT module has its own TX data buffer that is configured (in this design) to hold 512 bytes. Once that buffer is full, the XPORT module "triggers" and sends a VOIP packet.

For receiving, a 2 K byte RAM buffer in the PIC chip (2048 bytes) is used to store and hold the arriving VoIP bytes before speaker playback. Separate READ and WRITE memory pointers (FSR0 and FSR1) allow READ and WRITE operations to be performed in different parts of the RAM buffer, simultaneously. The average speed of both READ and WRITE operations is 8000 bytes/second. (crystal controlled)

The speed of READ operations (for speaker playback) is constant, and equal to 8000 bytes per second, but WRITE operations occur at 11,520 bytes per second, whenever UDP data is arriving from the XPORT module. There is some "idle time" between arriving UDP packets, which brings the average WRITE speed down to 8000 bytes per second.

The READ and WRITE pointers (FSR0 and FSR1) are initialized with 1 K of address separation, (50% of buffer capacity) but this separation increases and decreases constantly because of the varying WRITE speed. Because the average READ and WRITE speeds are equal, (and the buffer is large enough to accommodate the variations of WRITE speed) the two address pointers (FSR0 and FSR1) normally will never "overrun" each other. (If they did, it would yield garbled audio.)

The RAM buffer therefore provides a way to compensate for these READ/WRITE speed differences and variations, but it also compensates for variations of UDP packet travel times — successive UDP packets can travel at different speeds through the Internet, yielding a variation of arrival time as great as 30 milliseconds, measured from one packet to the next. The RAM buffer has enough capacity to hold 256 milliseconds of audio, and therefore it can accommodate these variations, as well.

In reality, the average READ/WRITE data rates are controlled by two separate crystals, (one at each end of the link) and minor differences between these two crystals will gradually accumulate (in the memory pointer values) and eventually cause an "overrun" between the READ and WRITE areas of the buffer RAM.

To deal with this, a few lines of PIC code constantly check the address difference between the two pointers, and if the separation distance ever gets "too close for comfort", (equal to 32 ms or less) the WRITE pointer address is re-initialized to a value with 1 K of address separation from the READ pointer address. This will cause a slight "glitch" in the playback audio (less than 100 ms) perhaps once every 5 to 10 minutes.

As a result of the normal operation of the RAM buffer, speaker audio is delayed by a typical value of 128 ms, but the delay can actually range from 32 to 224 ms. Delays smaller than 32 ms or greater than 224 ms will trigger an automatic "adjustment" of the WRITE pointer address.

Once a buffer byte is read and sent to the PWM for speaker playback, the value of the buffer byte is set to 0 × 80, to "erase" the byte. (0 × 80 = "no audio")

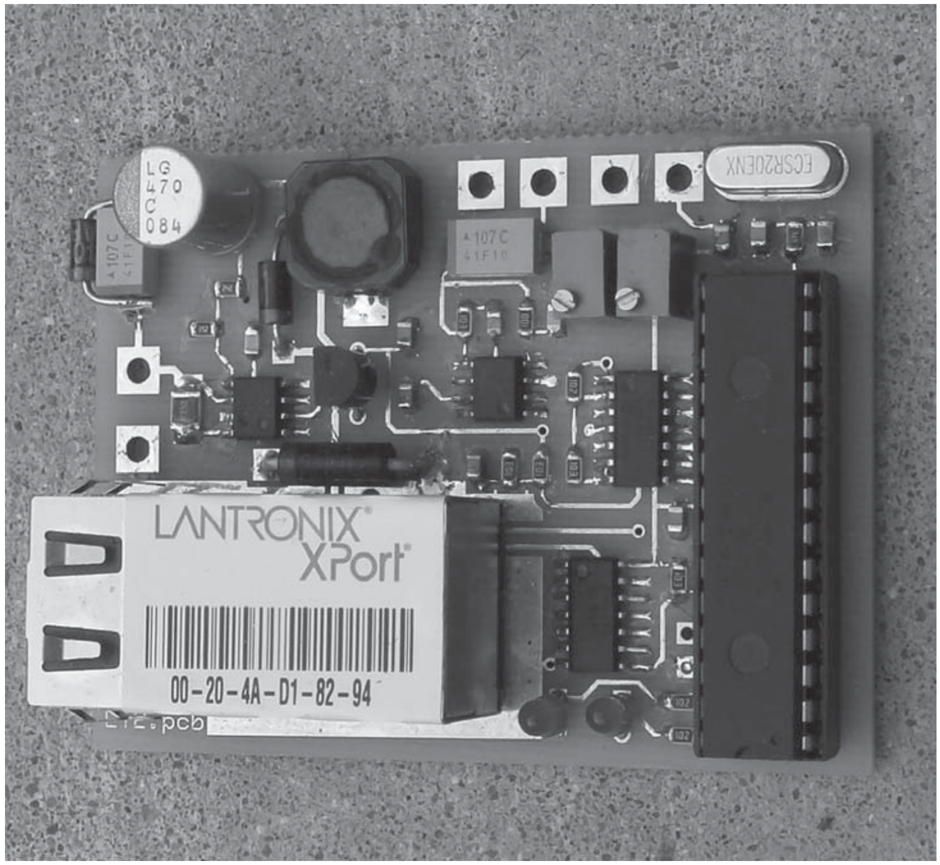


Figure 1 — This photo shows one of the completed VoIP boards. The Lantronix XPORT module is clearly visible.

The XPORT Module: More Information

General information about the XPORT module is available on the Lantronix website: **www.lantronix.com/device-networking/embedded-device-servers/xport.html**. In particular, the Quick Start Guide (**www.lantronix.com/pdf/XPort_QS.pdf**) and Users Guide (**www.lantronix.com/pdf/XPort_UG.pdf**) are worth viewing.

The (utility) ***Device Installer*** program is

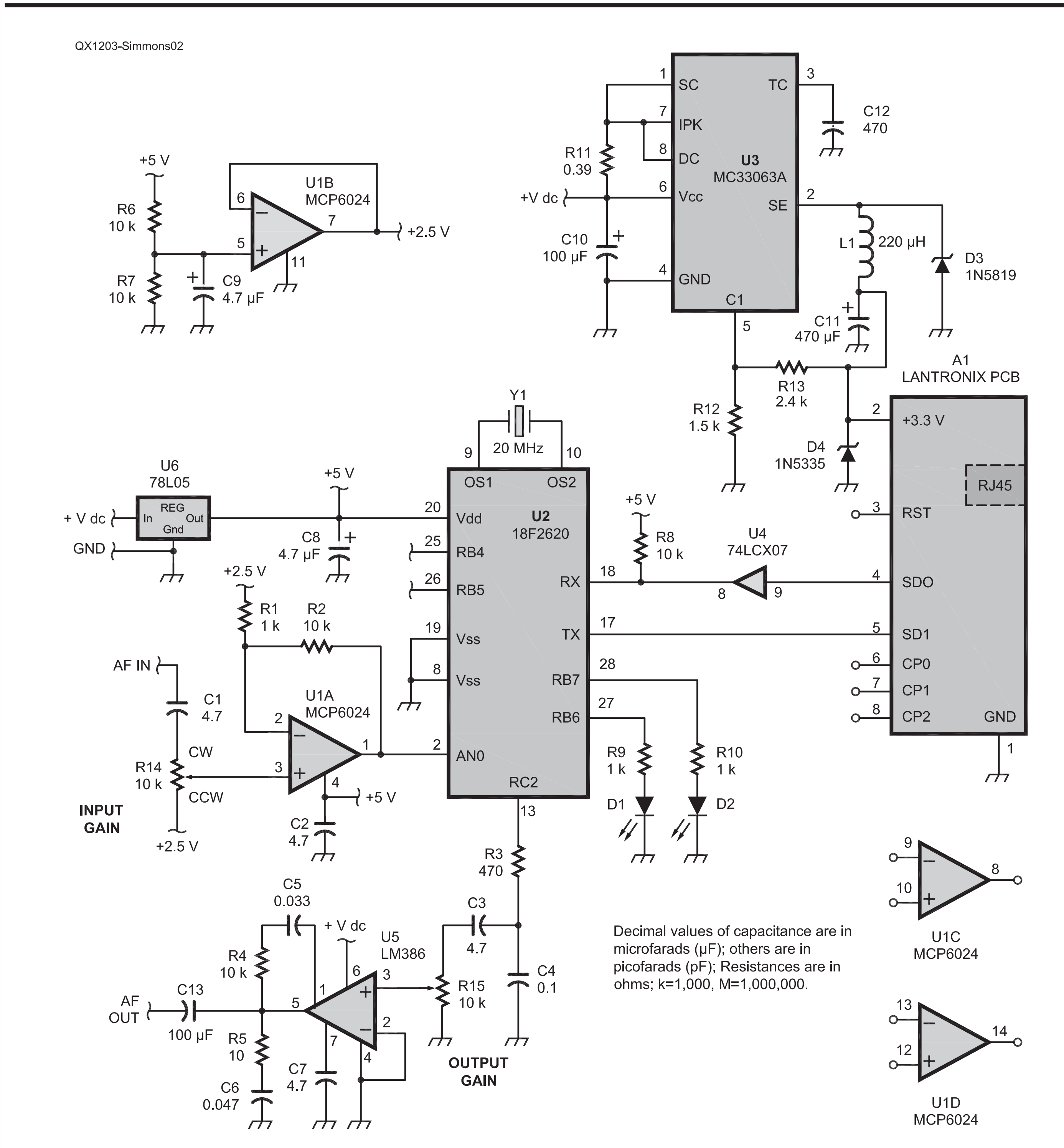


Figure 2 — Here is the schematic diagram of the VoIP module.

available here: **http://ltxfaq.custhelp.com/app/answers/detail/a_id/644**. Instructions about installing and running the ***Device Installer*** program are in the Quick Start Guide.

The Device Installer Program

The ***Device Installer*** program is used for several Lantronix products, not just the XPORT module. Some features in the ***Device Installer*** program, therefore, will not be applicable to the XPORT module (just ignore them).

Furthermore, the ***Device Installer*** program is a bit "buggy" in a few respects — if a parameter is changed in the XPORT module, you must click the APPLY SETTINGS button and wait several seconds for the new settings to be applied, before switching to a different display page. Checking to verify the new parameter settings have really been installed in the XPORT module will usually reveal that NO CHANGE has occurred, but this is a false display. The browser used in the

NOTE: All part numbers are DigiKey unless noted otherwise. Quantities are for a single VoIP module. Quantities must be doubled to make 2 boards for a complete link.

Component	Description	Part No.	Unit Price
	Circuit board	EXPRESSPCB	$10.89
A1	Lantronix XPORT	515-XP1001001-04R (Mouser)	$49.00
C1, C2, C3, C7, C8, C9	4.7 µF 0805	PCC2321CT-ND	$0.25
C4	0.1 µF 0805	478-1395-1-ND	$0.05
C5	33 nF 0805	478-1389-1-ND	$0.09
C6	47 nF 0805	478-1391-1-ND	$0.11
C10, C13	100 µF	478-1723-1-ND	$1.30
C11	470 µF	493-3114-ND	$1.73
C12	470 pF 0805	478-1324-1-ND	$0.09
D1, D2	LED T-1	67-1064-ND	$0.10
D3	1N5819	1N5819FSCT-ND	$0.68
D4	1N5335	1N5335BGOS-ND	$0.44
L1	220 µH	495-1801-1-ND	$1.82
R1, R9, R10	1.0 kΩ 0805	P1.0KACT-ND	$0.04
R2, R4, R6, R7, R8	10 kΩ 0805	P10KACT-ND	$0.04
R3	470 Ω 0805	P470ACT-ND	$0.04
R5	10 Ω 0805	P10ACT-ND	$0.04
R11	0.39 Ω 1210	RHM.39SCT-ND	$0.32
R12	1.5 kΩ 0805	P1.5KACT-ND	$0.04
R13	2.4 kΩ 0805	P2.4KACT-ND	$0.04
R14, R15	10 kΩ Trim Pot	490-2970-ND	$1.84
U1	MCP6024	MCP6024-I/SL-ND	$2.08
U2	18F2620	PIC18F2620-I/SP-ND	$7.68
U3	MC33063A	296-17763-1-ND	$0.68
U4	74LCX07	74LCX07M-ND	$0.51
U5	LM386	LM386M-1-ND	$0.94
U6	78L05	296-1365-1-ND	$0.40
Y1	20 MHz	XC1723-ND	$0.55

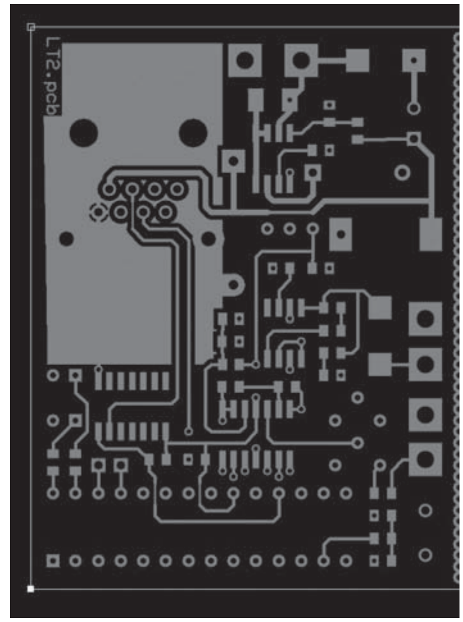


Figure 3 — This is a top view of the circuit board. The Lantronix XPORT module will be positioned over the large ground screen block at the top left corner.

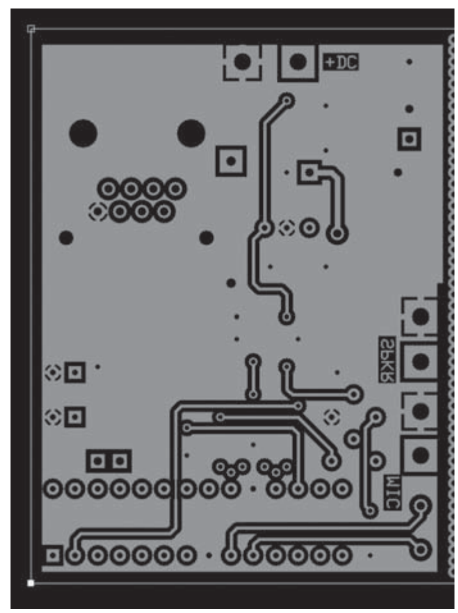


Figure 4 — This is an X-ray view of the bottom of the circuit board. Note the location for the Lantronix XPORT module in the upper right corner.

Device Installer program does not "refresh" properly, (even though a REFRESH button is provided) and typically the parameters truly have been changed. The only way to verify that the new parameters are properly installed in the XPORT module is to completely shut down (close) the ***Device Installer*** program and restart it. This flushes the browser's cache memory.

When the ***Device Installer*** program starts up, it conducts a search to find XPORT modules, but if multiple network interfaces are available, it might look in the wrong direction. It is prudent to disable or disconnect all network connections except the one to the XPORT module, while using the ***Device Installer*** program.

Configuring the XPORT Module

After installing the ***Device Installer*** program on your computer, and with the foregoing warnings about the program's "peculiarities" in mind, follow the instructions in the Quick Start Guide to begin the configuration process for the Lantronix XPORT module.

Some additional changes must be made after you have finished all the steps described in the Quick Start Guide. The information provided below will allow two VoIP modules to connect directly to each other with a cross-over cable, for bench tests.

Start up the ***Device Installer*** program, establish contact with the XPORT module, open up the web interface for the module and make the following changes:

NETWORK: Use static IP address

IP Address: 192.168.1.65 for board number 1

IP Address: 192.168.1.70 for board number 2

SERVER: MTU size = 512 (both boards)

CH 1 SERIAL SETTINGS: Baud rate = 115200 (both boards)

CH 1 CONNECTION: Protocol = UDP (both boards)

Datagram Type = 01(both boards)

Local Port = 10001 (both boards)

Remote Port = 10001 (both boards)

Remote Host = 192.168.1.70 for board number 1

Remote Host = 192.168.1.65 for board number 2

Shut down the ***Device Installer*** program and restart it. Examine the parameters to verify they have all truly been entered properly and saved in the XPORT module. Do this for both VoIP boards.

Bench Testing the VoIP Boards

At this point, hook up the two VoIP boards as shown in Figure 6, to test their operation. The two speakers shown in the diagram will allow you to verify the operation. The lead photo shows the test setup on my workbench. If everything is working properly, you will hear speaker audio in both speakers. Speaker audio in the receiving board will be delayed several dozen milliseconds, yielding a distinct "echo" quality to the sound. You can adjust the audio levels with the trimpots to equalize the speaker levels, which will maximize the "echo" effect.

Setting Up an Internet Link

This topic is beyond the expertise of the author, and the scope of this article. Some people reading this article will have a very good idea what must be done, but others will not — for the latter folks, enlist the help of someone well-versed in the black art of setting up computer LANs.

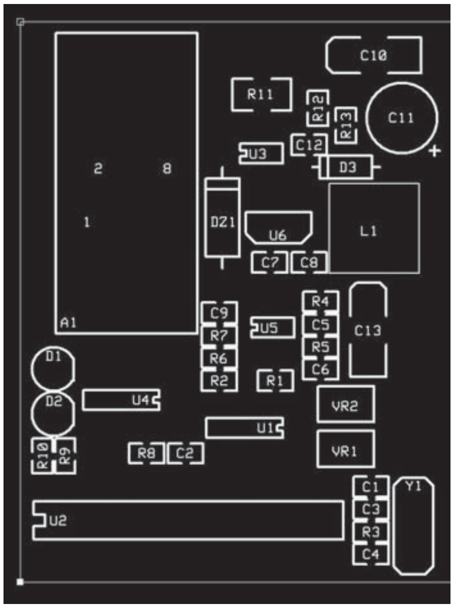


Figure 5 — The parts placement screen pattern shows the various circuit components.

Please note "machine level" IP addresses (IPv4) presently consist of 4 individual bytes. (6 byte addresses for IPv6 are coming in a few years) These bytes are written as individual numbers, in decimal notation with leading zero suppression, delimited by decimal points. The range of possible IP addresses runs from "0.0.0.0" to "255.255.255.255". This yields almost 4.3 billion possible addresses.

IP port numbers (at the machine level for IPv4) consist of a 2 byte unsigned integer, expressed in decimal notation as a single number. Therefore the possible range of port number runs from "0" to 65535".

Each IP address has 65536 possible port numbers associated with it. By analogy, an IP address is like a telephone number, and the port number is like an "extension number" available at that phone number.

Basically, a clear pathway directly between the Lantronix module and the Internet must be created in the LAN. This will require selecting an IP port number for the Lantronix module, one that is "free" and not being used by any other computers in the LAN. The port number should lie between 49152 and 65535, and definitely avoid port numbers below 1024, which are pre-defined in the Internet community as "well known" ports.

Once a suitable port number is identified, the gateway computer (the switcher or router that connects the LAN to the Internet) can be programmed to pass all traffic using that port directly to/from the Lantronix module. This is called "port forwarding."

The Internet address ("worldwide web" address) for the LAN gateway computer cannot be independently selected, its value is "dictated" by the ISP (Internet Service

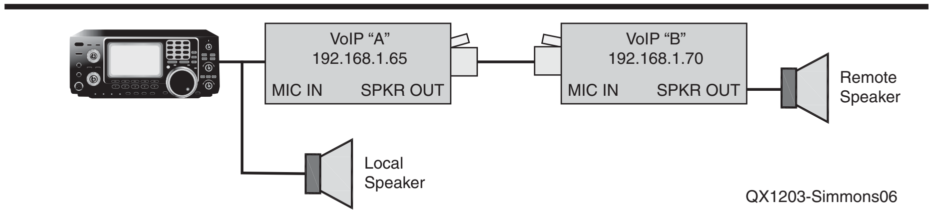


Figure 6 — This block diagram shows the setup used to test a pair of the VoIP circuit boards.

Provider), which provides the external Internet connection to the LAN. This address must be identified, because it will be used to program the Lantronix module at the other end of the link. (The port number will also be required.)

For reliable operation, the IP address should be a "static" address (one that never changes) and static addresses can be provided (rented) from ISP companies for a monthly fee. Most end-user IP addresses are "dynamic" (temporary), and can change at any time, with no warning. Generally, these addresses only change whenever a user (the LAN gateway computer) "signs off" from an Internet connection, but if this never happens, a dynamic address might remain constant and never change for weeks or months. In contrast, a static IP address is guaranteed to never change.

All these actions will involve some "configuration" work with the switcher or router that provides the Internet connection for the LAN (the "gateway" computer). If the switcher/router also has a firewall program, it might also be necessary to instruct the firewall to "make an exception" for any Internet traffic to or from the specific port assigned to the Lantronix module. ("Don't interfere" with traffic to or from the Lantronix module.)

The CAD Files and Parts List

The schematic and PCB CAD files were created with ExpressPCB, which provides free CAD tools to do this kind of work. The ExpressPCB CAD tool has a feature for directly ordering the PC boards via the Internet, with a credit card. (This option is in the LAYOUT menu.)

Use the ExpressPCB "MiniBoard" service when ordering the boards. This is their most economical service, but orders are limited to 3 identical boards measuring 2.5 × 3.8 inches, which will yield enough modules for 3 complete links (6 VoIP modules). Each ExpressPCB board holds 2 VoIP modules, with a line of closely-spaced VIAs separating them. The VIAs serve as "perforations" that allow the two modules to be manually "broken apart" into separate boards.

You can get the CAD tools to view the files and order the circuit boards at the ExpressPCB website: **www.expresspcb.com/**

The CAD files for these circuit boards (schematic and PCB) are available for download from the ARRL *QEX* files website.[1]

The PC board itself should be ordered using the ExpressPCB "MiniBoard" service, which will yield enough boards for 3 complete Internet links.

Credits

This module was developed and tested in collaboration with John Piri, WD6CSV, and GTMR Corporation. They have both kindly granted permission to report the results in this article, in exchange for honorable mention. You can visit the GTMR company website at: **www.gtmrinc.com/**.

The author maintains a website with information about some of his other projects, mostly related to radio direction finding and hidden transmitter hunting. Visit this website at: **www.picodopp.com**

[1]The ExpressPCB files as well as the Assembly language program code are available for download from the ARRL QEX files website. Go to **www.arrl.org/qexfiles** and look for the file **3x12_Simmons.zip**.

Bob Simmons, WB6EYV, was first licensed as a novice in 1964 at age 13, and remained licensed (more or less) constantly ever since. He also earned a commercial FCC license in 1967. He served Naval Reserve duty as a radar technician (ETR2) with about 6 months of total sea time. He spent several years of civilian work in nautical and marine electronics in Los Angeles harbor, as well as doing some land mobile radio work, followed by 5 years in flight line avionics, working on business jets. He moved to Santa Barbara, CA in 1992 and worked on vacuum deposition systems for 5 years, and held assorted odd engineering jobs at other times. Presently, Bob is self employed and runs a website making and selling radio direction finding equipment and modules, with a majority of his "new" work spent creating embedded software / hardware and developing technologies to enable Internet-linked remote DF stations. His primary interest is developing and applying new technologies to old problems, and pushing the DF "art" forward.

QEX

Roger Traylor, ND7PA

6800 NW Quailwood Dr, Albany, OR 97321; **traylor@ece.orst.edu**

An Open Source Keyer with Programmable Control Outputs

Not sure how an iambic keyer works? Think state machines, and code your own.

Richard Chapman's (KC4IFB) *QEX* article, "Build a Low-Cost Iambic Keyer Using Open-Source Hardware" describes the software and hardware design of an iambic keyer.[1] This was the first software design of a keyer I'd ever seen that was easily comprehended. His design was that of a simple keyer, uncluttered by dozens of features. Richard also implemented his project in a high-level language and documented it well in his writing.

Inspired by his work, I determined to improve the design and provide documentation so that others could easily understand, copy, and then further enhance it. This article documents my design approach and the features I added. The result is an open-source keyer that can be easily modified by non-professional programmers to meet their

[1]Notes appear on page 31.

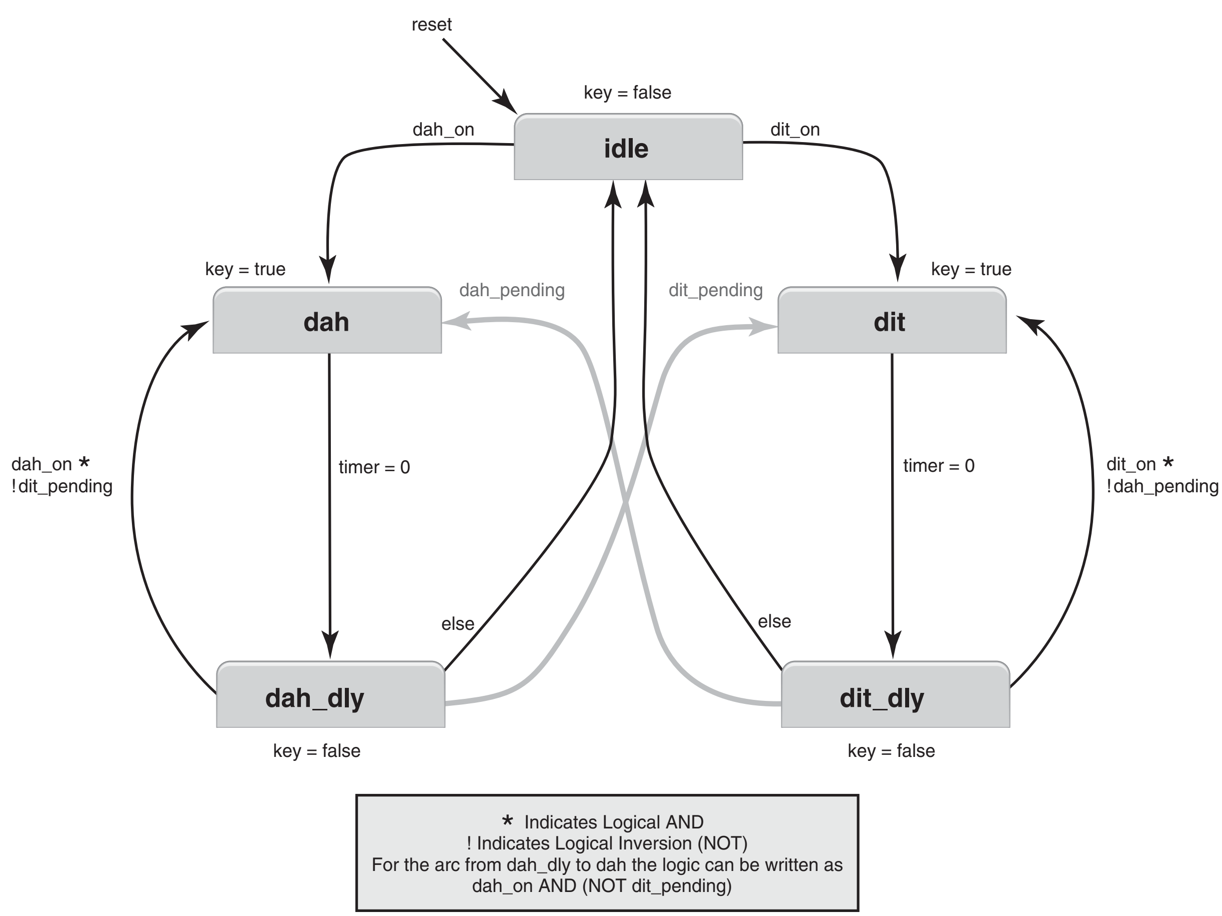


Figure 1 — This is the main keyer state machine diagram.

specific needs. In addition, my hope is that this article can become an entry point for those who want to learn about programming microcontrollers using a high-level language.

My improvements were:

• A state machine approach to improve design clarity.

• Interrupt driven code execution.

• Better insight and programmability to the iambic sampling period.

• User programmable iambic sampling times.

• User programmable timing of multiple control outputs.

• Sidetone output.

• Save and restore the previous keyer speed.

Things I changed were:

• Use of standard *C* language instead of Arduino *Sketch.*

• Use of easily available GNU AVR toolchain.

• Use of an encoder instead of a potentiometer for speed adjustment.

Language and Programming Environment

The original article leverages the very popular Arduino **(www.arduino.cc)** *Sketch* language. While convenient for expressing simple hobby projects, I use the standard *C* language in my day job teaching microcontroller system design at the local university. In my class we use the free GNU toolchain for Atmel's AVR microcontroller with standard *C* language. As such, I wanted to implement the keyer in this language and development environment.

Fortunately, *Sketch* is very similar to *C*. I translated Richard's first keyer design to *C* and had working code in about 30 minutes. If you can program in *Sketch* you can easily make the jump to *C*.

The GNU toolchain is a set of tools for developing *C* programs on Atmel's AVR family. It includes the compiler, assembler, linker and other tools needed. The tool set runs on *Windows*, *Linux* and *Macintosh* computers.[2] It is heavily used and supported by a very active group of developers. The code generated is of high quality.

In my opinion, for most applications, there is little need for assembly language. Essentially shackled to one microcontroller family, it's far more difficult to maintain and clearly document. Also, if I want to switch microcontrollers, I simply compile to a different target. For me, life is too short for assembly language.

Keyer State Diagrams

By his own admission, Richard's "Version Two" keyer implementation was a bit complicated. While reading his code, I realized that the code structure was very similar to a hardware state machine. I thought about what it might look like to implement iambic behavior within a state machine paradigm.

The result was a main state machine with five states plus two, two-state state machines for dot and dash iambic memory. The result is a simple and easy to understand implementation of iambic keying. Before looking at any code, let's look at the state diagrams for the keyer. If you can comprehend the state diagrams, following the code becomes straightforward.

Referring to Figure 1, the main state machine variable ***keyer_state*** can take on one of the five state values: ***idle***, ***dit***, ***dah***, ***dit_dly***, or ***dah_dly***. After power-up reset, ***keyer_state*** becomes ***idle***. Transitions between states as well as changes to software counters always occur at a timer interrupt. In my design, a timer/counter is programmed to interrupt the processor every 128 µs. This sets the timing resolution of the keyer to 128 µs/20 ms of a dot at 60 WPM.

Starting from ***idle,*** if the dot paddle is pressed, at the next interrupt the new state becomes ***dit***. Likewise, if the dash paddle is pressed, at the next interrupt the new state becomes ***dah***. Once in either ***dit*** or ***dah*** states, the software counter ***timer*** decrements until the period of the dot or dash element is complete. The length of the dash element is three times the delay of the dot element by convention. Software counter ***timer*** decrements each time an interrupt occurs. The following transition from ***dit*** or ***dah*** is always to the corresponding ***dit_dly*** or ***dah_dly*** state.

While in either ***dit*** or ***dah***, an internal variable ***key*** is asserted true. This is an internal signal indicating an element is being formed. If no control outputs were implemented, this signal could be applied to a microcontroller pin to key a transmitter. Here, I use it as an internal signal only.

The states ***dit_dly*** and ***dah_dly*** provide a one dot period time delay between the dot and dash elements respectively. Variable ***key*** is deasserted during ***dit_dly*** or ***dah_dly***. These states also maintain the history of what element was sent last. Transitions from ***dit_dly*** or ***dah_dly*** depend on what happened while the respective element was being sent. For example, if the letter "I" was sent, the sequence of states would be ***idle***, ***dit***, ***dit_dly***, ***dit***, ***dit_dly***, ***idle***. After the last ***dit_dly*** state, since no further dots remain to be sent and no dash has been detected during the sending of the "I," the ***else*** arc back to ***idle*** is taken. If, however, the dash paddle was hit sometime during the creation of the "I," then the ***dah_pending*** arc would be taken to ***dah***.

The code for the main state machine is shown in Figure 2. For each state in the

```
//keyer main state machine
 timer--;       //decrement clock
 switch(keyer_state){ //go the present state
  case (IDLE) :
   key = FALSE;   //key not asserted
   if   (dit_on()){timer = timeout;  keyer_state = DIT;}
   else if(dah_on()){timer = timeout*3; keyer_state = DAH;}
   break;
  case (DIT) :
   key = TRUE;
   if(!timer){timer = timeout; keyer_state = DIT_DLY;}
   break;
  case (DAH) :
   key = TRUE;
   if(!timer){timer = timeout; keyer_state = DAH_DLY;}
   break;
  case (DIT_DLY) :
   key = FALSE;
   if(!timer){
    if(dah_pending == TRUE) {timer=timeout*3; keyer_state = DAH;}
    else if(dit_on())    {timer = timeout;keyer_state = DIT; }
    else   {keyer_state = IDLE;}
   }
   break;
  case (DAH_DLY) :
   key = FALSE;
   if(!timer){
    if(dit_pending == TRUE) {timer=timeout;  keyer_state = DIT;}
    else if(dah_on())    {timer=timeout*3; keyer_state = DAH; }
    else  {keyer_state = IDLE;}
   }
   break;
 }//switch keyer state
```

Figure 2 — Here is the *C* code for the keyer main state machine.

main state machine there is a ***case*** selection. Each arc from a state has a corresponding ***if/else*** conditional statement that determines if the arc will be taken. In each state, ***key*** may be asserted or not. This style of coding is very similar to those found in the hardware description languages *Verilog* and *VHDL*.

Figure 3 shows the state machines that create the iambic memory. Essentially, they preserve the information that a dot paddle was pressed while a dash was being sent or a dash paddle was pressed while a dot was being sent. These state machines provide this information to the main state machine for making the transition from the ***dit_dly*** or ***dah_dly*** states.

For example, if while a dot is being sent, the dash paddle is pressed, the ***dah_pending*** state variable assumes the ***TRUE*** state. Thus, when the ***dit_dly*** state has timed out, the transition is to the ***dah*** state. Once in ***dah***, the ***dah_pending*** state variable transitions back to the ***FALSE*** state, since the pending event is underway. If a dash or dot element completes without another pending element, the transition is back to the ***idle***.

Detection of Iambic Keying Condition

Detection of a dot or dash during formation of the opposite element is a bit more subtle than described above. After some reading, I believe that different keyers use varying schemes to sample the state of the "other" paddle. I suspect that the sampling method is "secret sauce" that differentiates them. I experimented with exactly where my code was sampling the other paddle and came up with a setting that behaves like many other keyers I have used, and is natural to me. I will show how I set mine up, but the timing shown is adaptable to other schemes.

Figure 4 shows the main keyer states and the value of ***key*** when a dot followed by a dash and another dot is sent. The sampling window interval for a dash while a dot is being sent is shown first. The sampling window includes the latter half of ***dit*** and the first half of ***dit_dly***. If the dash paddle is depressed within the window, the ***dah_pending*** state variable will transition to ***TRUE***. Likewise, a dot may be detected during the window of time including last third of a dash period or the first half of the ***dah_dly*** time that follows.

The code for the iambic state machines is listed in Figure 5. The code structure is very similar to that of the main state machine.

Making the Control Outputs

Just after I had tinkered with Richard's keyer design, I was in the process of designing a 40 m direct conversion (DC) transceiver. I included a popular keyer chip in the design and found it only had one muting output, which did not provide the functionality necessary for audio muting and transmit relay changeover sequencing. Pleased with the new keyer design, I thought that sequencing the receiver muting and receive/transmit relay outputs could be easily done with the state machine paradigm within the new keyer design.

My first keyer design used the keyer state machine variable ***key*** to actually key the transmitter. This now had to become an inter-

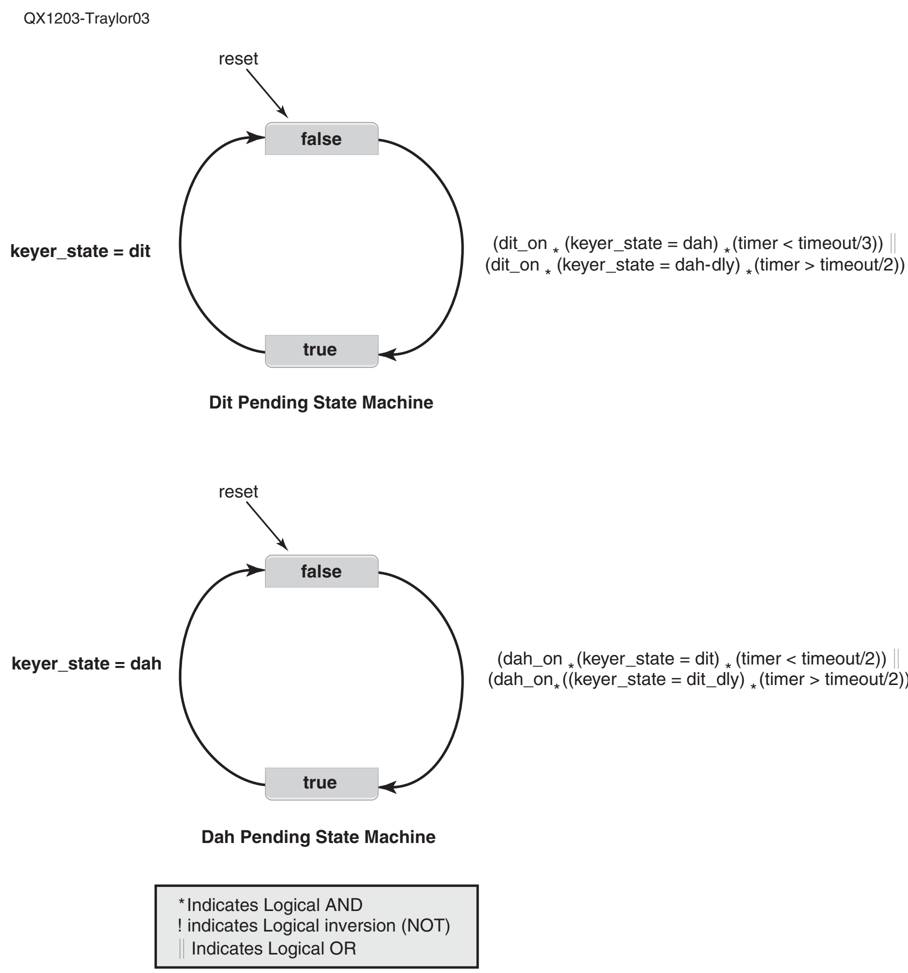


Figure 3 — These are the dot and dash iambic memory state machine diagrams.

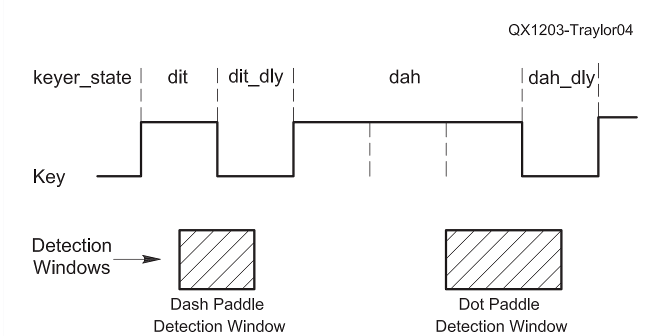


Figure 4 — This diagram shows the dot and dash paddle sampling times for iambic operation.

```
//***************** dit pending state machine  ********************
 switch(dit_pending){ //see if a dot is pending
  case (FALSE) :
   if( (dit_on() && (keyer_state == DAH)  && (timer < timeout / 3))  ||
     (dit_on() && (keyer_state == DAH_DLY)&& (timer > half_timeout)))
    { dit_pending = TRUE;}
   break;
  case (TRUE) :
   if(keyer_state == DIT){dit_pending = FALSE;}
   break;
 }//switch dit_pending

//***************** dah pending state machine  ********************
 switch(dah_pending){ //see if a dah is pending
  case (FALSE) :
   if( (dah_on() && (keyer_state == DIT)  && (timer < half_timeout)) ||
     (dah_on() && (keyer_state == DIT_DLY)&& (timer > half_timeout)))
     {dah_pending = TRUE;}
   break;
  case (TRUE) :
   if(keyer_state == DAH){dah_pending = FALSE;}
   break;
 }//switch dah_pending
```

Figure 5 — Here is the *C* code for the iambic memory state machines.

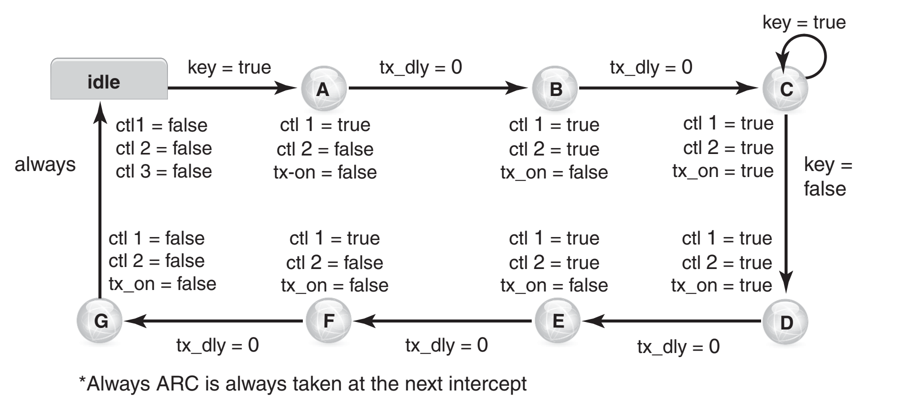


Figure 6 — This is the output sequencer state machine diagram.

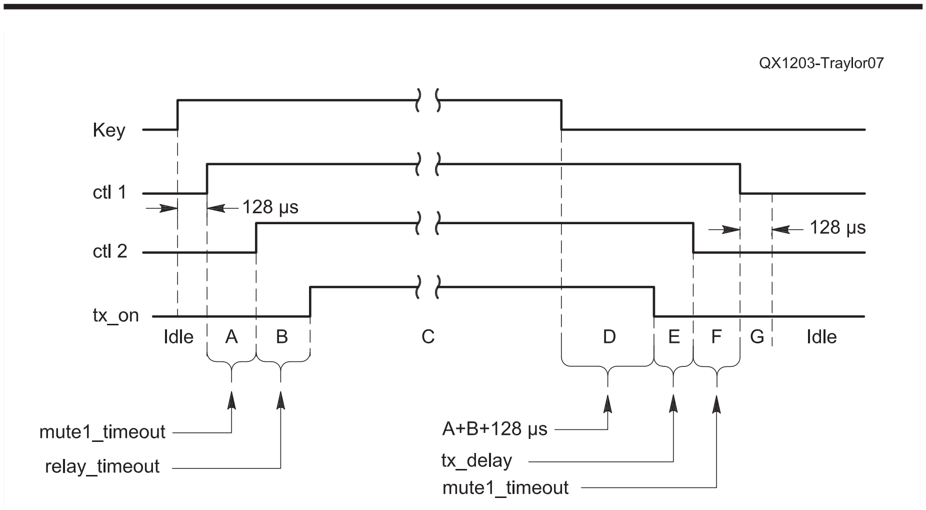


Figure 7 — This chart shows the output sequencer timing.

nal signal. With the inclusion of the control outputs, the variable ***key*** is used by the output sequencer state machine to kick-off the sequence of muting events and also to actually key the transmitter via the signal ***tx_on***.

Figure 6 shows the state diagram of the output sequencer state machine. Its design follows the prior examples in form and style. The outputs controlled by this state machine are the variables ***ctl1***, ***ctl2*** and ***tx_on***. The control of the sidetone is also under control of this state machine. I use ***ctl1*** as an audio mute, ***ctl2*** as the transmit/receive relay control and ***tx_on*** to actually key the transmitter. The delays between states are controlled by another software timer ***tx_dly*** that is decremented at each interrupt as before.

This state machine uses several user programmable constants as *C* ***#define*** statements to control the delay between the control signals. These are defined in Table 1.

Figure 7 shows a timing diagram of how the outputs sequence, and their relative timing. In Figure 7, the control outputs are shown as well as the states that correspond to the state diagram. Note in state D that the delay is computed from the sum of the delays in states A and B plus one interrupt period. This lengthens the element to compensate for the delays inserted by states A and B. Thus, the first element and all following elements have exactly correct periods.

If the variable ***key*** becomes ***TRUE***, the ***ctl1*** mute output asserts within one interrupt period, (128 µs). Following this by a delay of ***mute1_timeout***, the ***ctl2*** output is asserted. This is used for the transmit/receive changeover relay. Since the relay needs a minimum changeover time, a delay of ***relay_timeout*** must pass before asserting ***tx_on***.

This brings the state machine to state C, which is where it stays until ***key*** deasserts. When that happens, a delay is applied to the element period of A+B+128 µs. This keeps the first element from being shortened by the switching delays. Then ***tx_on*** is deasserted. To allow for the decay in transmitter output caused by key shaping, a delay of ***tx_decay*** must pass before the relay is switched back to receive. A period of ***mute1_timeout*** later, the audio is unmuted by ***ctl1***. The final transition to the ***idle*** state completes the cycle.

This scheme provides full break-in keying at up to the speed of the relay and audio muting system. It is also user programmable for different needs. If desired, additional control outputs may be implemented by simply adding more states and setting the timer to count down as needed.

Sidetone

To provide a sidetone, a timer/counter is used to generate a square wave tone of user defined frequency. It is activated by enabling and disabling the clock source to the timer counter in sync with the output key assertion. Since a square wave tone is not pleasing to the ears, a 1 kHz low pass filter was added to the output, which results in a nice tone. The sidetone frequency may be changed by altering the compare value to the counter timer used to create the side tone. The complete keyer code listing is available for download from the ARRL *QEX* files website.[3]

Speed Adjustment and Memory

To preserve the limited ADC inputs to the microcontroller for other uses, a mechanical encoder is used to adjust the speed. At each interrupt, the state of the encoder outputs are sensed and debounced to detect clockwise, counterclockwise or no rotation. This con-

```
//check encoder for movement and adjust speed accordingly
 switch(encoder_chk()){          //see if user changed the speed setting
  case (0) : if(wds_per_min > 5){wds_per_min--;  }//check, possibly decrement
        else        {wds_per_min = 5; } //bound if necessary
        ee_wait_cnt=32760;    //countdown to save new setting to eeprom
        break; //decrease speed
  case (1) : if(wds_per_min < 60){wds_per_min++;  }//check, possibly increment
        else        {wds_per_min = 60;} //bound if necessary
        ee_wait_cnt=32760;    //countdown to save new setting to eeprom
    break;          //increase speed
  case (-1): break;    //no change
  } //switch
//see if we need to save the present speed setting
 if(ee_wait_cnt>0){ee_wait_cnt--;} //keep counting down till its time to write
 else if(ee_wait_cnt==0){      //upon reaching zero, we save the setting
  eeprom_write_byte(&eeprom_wds_per_min, wds_per_min); //save the setting
  ee_wait_cnt--;          //decrement to -1 to prevent more writes
 }
```

Table 1
User Configurable Delays for Control Outputs

Constant	*Function*
mute1_timeout	Sets delay between assertion of ***ctl1*** and ***ctl2***. As used, this sets the delay between muting receiver audio and switching the transmit/receive relay. After the deassertion of ***tx_on***, mute1_timeout also sets the delay from the deassertion of ***ctl2*** to the deassertion of ***ctl1***.
relay_timeout	Sets delay between the assertion of the transmit/receive relay and assertion of ***tx_on***. As used it provides time for the relay to change prior to energizing the transmitter.
tx_decay	Sets delay between deassertion of ***tx_on*** to switching the relay back to receive. It allows for the transmitter RF envelope to decay (typically 5ms) before the relay switches.

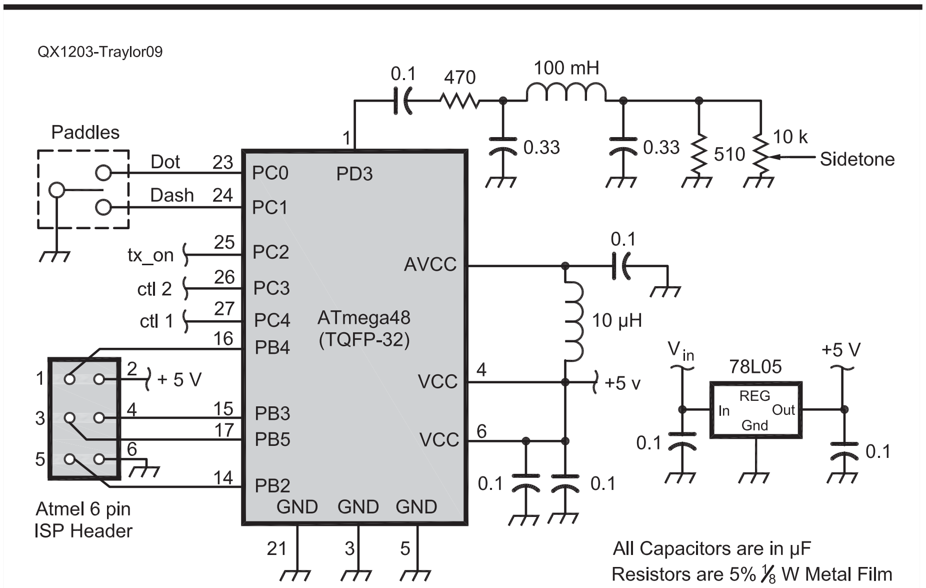


Figure 9 — This schematic diagram shows the keyer circuit.

trol directly sets the keyer speed from 5 to 60 WPM. The variable ***timer***, which controls the element length is computed from the variable ***wds_per_min***.

When the speed is changed, and is not adjusted again for about 5 s, the current speed setting is saved to EEPROM. This is accomplished by setting a counter to a large value and decrementing it until zero is reached. At zero, the setting is saved. After power up, the setting is restored from EEPROM in the function ***setup()***. The code for these functions is given in Figure 8.

Hardware Design

My hardware was similar to Richard's. I used a prototyping board with a surface mount ATmega48 processor ***(www.atmel.com/dyn/resources/prod_documents/doc2545.pdf)***. Unlike Richard's keyer, I use the internal 8 MHz oscillator instead of an external crystal to provide the clock to the microcontroller. Pull-up resistors internal to the microcontroller are used for the paddle and encoder inputs. An external keying transistor is also used as in the former article. The schematic for my keyer is shown in Figure 9.

My keyer is embedded within a homebrew direct conversion transceiver. It controls my version of Rick Campbell's R2 receiver from the Jan 1993 issue of *QST*, with an RF LNA similar to the R2-Pro design presented by Rick Campbell and Bill Kelsey at the 2000 Four Days in May.[4, 5] The R2 receiver muting is the conventional JFET series muting element. Pulling the gate terminal of the FET to ground mutes the audio portion of the receiver. Both of these two mute functions use the ***ctl1*** output from the keyer. The transmit offset at the VFO is controlled by ***ctl1***.

The ***ctl2*** output controls the transmit/receive relay. The ***tx_on*** output is applied to deactivate the receive incremental tuning and to activate the PA driver. The sidetone output is applied to the audio output stage of the R2.

Programming

The microcontroller can be programmed by one of the many Atmel programmers available. One of my favorites is a USB-based programmer called the **USBTinyISP** available from **Adafruit Industries (www.adafruit.com)**. Compiling and downloading of the code is done with a Makefile. Avrdude is the software tool used to do the programming. The Makefile controlling compilation and programming is included in the file set that is available. The six-pin header on the prototyping board is used with the programmer to load the programs into the chip.

Results

The keyer works nicely. The method of detection of the dot/dash overlap condition gives me an iambic keyer that I never had to adapt to. It seems very natural to use. The sequencing of the muting function works as expected. By using a reed relay, I can hear signals between dots at 40+ WPM. The timing of all the outputs is programmable and easy to change. In fact, my initial guess for relay timing was reconnecting the receiver just a few microseconds too soon leading to a big thump in the speaker once the key was released. A quick edit of code resolved the issue immediately. I can also tailor this design to the requirements of other projects easily.

Even though the keyer is located very close to a 5 W PA (see Figure 11), no interaction was noted between the transmitter and microcontroller. Another possible issue, especially with DC receivers, is that noise from the microcontroller would be picked up by the large audio gain in the R2. No such noise was ever noticed in casual operation, however.

Output Waveforms

Figure 12 shows the actual output waveforms from the keyer. The dot paddle was hit for a very short time relative to the actual element length. The top trace is the input coming from the paddle. Note the slow rising edge from the large pull-up resistors inside the microcontroller. The second trace is the output ***ctl1***. It is used to mute the audio on the R2 board as well as deactivating the RF LNA. The third trace is the ***ctl2*** output, which controls the transmit/receive relay. Finally, the bottom trace is the transmitter RF envelope.

What's Left?

There is an enormous amount of work that can still be done with this project. The current code only consumes 1380 bytes out of 4096 available bytes. If that is not enough, the ATmega88 chip is a pin-for-pin compatible replacement with 8 Kbytes of memory. I have intentionally left out any further bells and whistles, as my current needs have been met. Some possible new features include:

- Sleep between code elements to save power.
- Make programmable features adjustable via external controls.
- Current speed indication via code or display.
- Addition of other features such as a frequency counter.

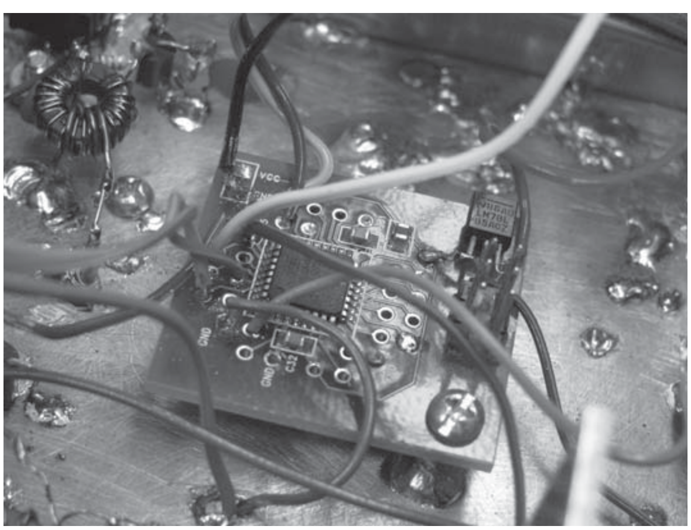


Figure 10 — This photo shows the keyer, built on a proto-board.

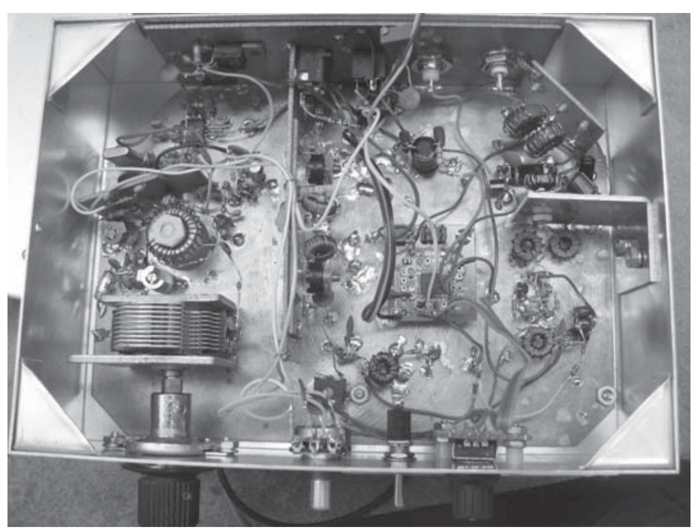


Figure 11 — Here, the keyer board (just to the right of center) is installed inside a direct conversion rig.

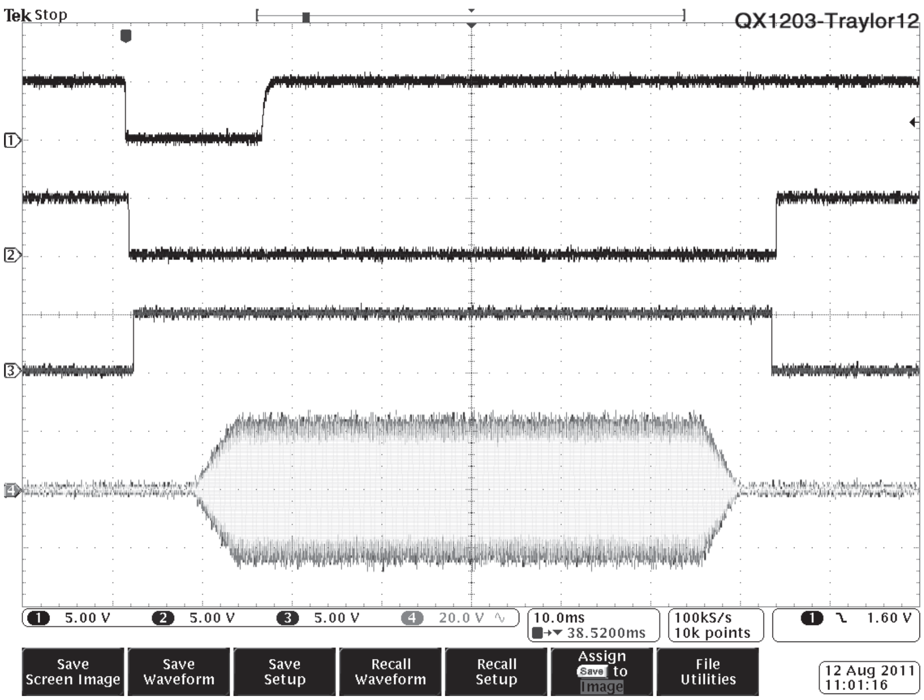


Figure 12 — This oscilloscope screen capture shows the keyer output waveforms. The top trace represents the paddle input to the microcontroller. The second trace is the audio muting signal (ctl1) and the third trace is the transmit/receive relay control (ctl2). The bottom trace is the transmitter RF envelope.

Summary

The code is available for anyone to improve and add to under the conditions of the GNU General Public License. The code may be downloaded from the ARRL *QEX* files website (see Note 3).

Notes

[1]Richard Chapman, "Build a Low-Cost Iambic Keyer Using Open-Source Hardware," *QEX*, Sep/Oct 2009, pp 3-7.

[2]For *Windows* computers, see: **winavr.sourceforge.net**, For Macintosh computers, see: **www.obdev.at/products/crosspack/index-de.html**, For computers running *Linux*, see: **www.cs.hut.fi/Studies/T-106.530/2006/installation.html**.

[3]The program code listing is available for download from the ARRL *QEX* files website. Go to **www.arrl.org/qexfiles** and look for the file **3x12_Traylor.zip**.

[4]Rick Campbell, "High-Performance, Single-Signal Direct-Conversion Receivers," *QST*, January 1993, pp. 32-40.

[5]Rick Campbell and Bill Kelsey, "A Next-Generation R2 Single-Signal Direct Conversion Receiver," presentation given at Four Days In May, Dayton, Ohio, May 2000.

Roger Traylor, ND7PA, lives in Albany, Oregon. He has Bachelors and Masters Degrees in Electrical and Computer Engineering. He worked for GTE's Strategic Systems Division for 4 years as a design engineer and for Intel Corporation for 9 years working on various FPGA and ASIC projects for parallel supercomputers. Presently, he is a senior instructor in the School of Electrical and Computer Engineering at Oregon State University, teaching introductory ECE courses as well as electromagnetics, VLSI *design and microcontroller system design. He holds four US patents in the area of synchronous and asynchronous logic design.*

Roger was first licensed as WN4QIL in 1969 and later as WB4TPW in 1970. He currently holds an Amateur Extra Class license. Roger's ham radio interests usually revolve around QRP operation, homebrewing, and CW. His first rig was a Ten-Tec PM2B, which is probably to blame for his continuing interest in building direct conversion receivers.

Roger is married and has three children. He enjoys backpacking, fishing, hunting, gardening or any other activity that gets him outdoors. In his spare time he continues to enjoy tinkering with electronics.

QEX

Rudy Severns, N6LF

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

A Closer Look at Vertical Antennas With Elevated Ground Systems

N6LF shares his results from more vertical antenna experiments.

[This article is being published in two parts. — Ed.]

Among amateurs, there has been a long running discussion regarding the effectiveness of a vertical antenna with an elevated ground system compared to one using a large number of radials either buried or lying on the ground surface. NEC modeling has indicated that an antenna with four elevated λ/4 radials would be as efficient as one with 60 or more λ/4 ground based radials. Over the years there have been a number of attempts to confirm or refute the NEC prediction experimentally, with mixed results. These conflicting results prompted me to conduct a series of experiments directly comparing verticals with the two types of ground systems. The results of my experiments were reported in a series of *QEX*[1-7] and *QST*[8] articles (Adobe *Acrobat* .pdf files of these articles are posted at **www.antennasbyn6lf.com**). From these experiments I concluded that at least under *ideal* conditions four elevated λ/4 radials could be equivalent to a large number of radials on the ground.

Confirmation of the NEC predictions was very satisfying but that work *must not be taken uncritically*! My articles on that work failed to emphasize how prone to asymmetric radial currents and degraded performance the 4-radial elevated system is. You cannot just throw up any four radials and get the expected results! I'm by no means the first to point out that the performance of a vertical with only a few radials is sensitive to even modest asymmetries in the radial fan.[9, 10, 11] It is also sensitive to the presence of nearby conductors or even variations in the soil under the fan.[12] These can cause significant changes in the resonant frequency, the feed point impedance, the radiation pattern and the radiation efficiency. While these problems have been pointed out before, as far as I can tell no detailed follow-up has been published. Besides the practical problem of construction asymmetries, at many locations it's simply not possible to build an ideal elevated system even if you wanted to. There may not be enough space or there may be obstacles preventing the placement of radials in some areas or other limitations. I think it's very possible that some of the conflicting results from earlier experiments may well have been due to pattern distortion and increased ground loss that the simple 4-wire elevated system is susceptible to.

As the sensitivity of the 4-radial system and its consequences sank into my consciousness I began to strongly recommend that people use at least 10 to 12 or more radials in elevated systems. Although I have heard anecdotal accounts of significant improvements in antenna performance when the radial numbers were increased to 12 or more, I have not seen any detailed justification for that. What follows is my justification for my current advice.

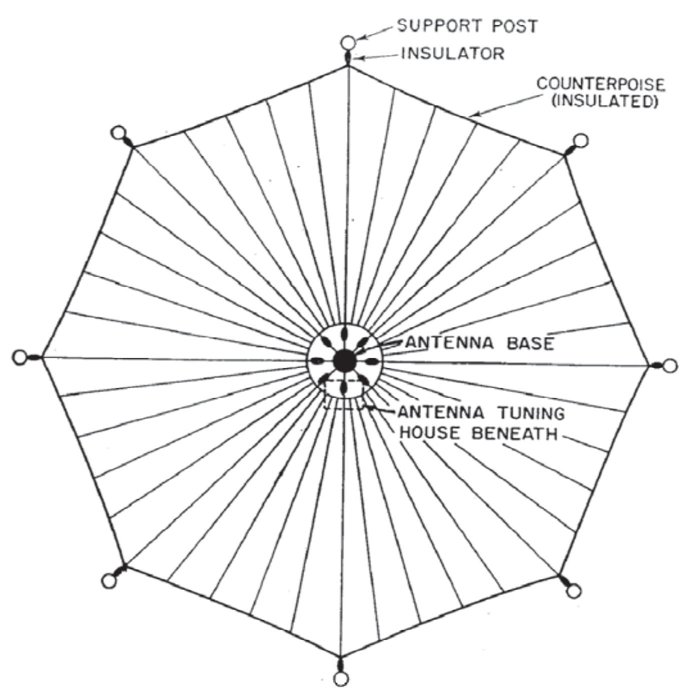


Figure 1 — A typical counterpoise ground system. Figure adapted from from Laport.[14]

[1]Notes appear on page 41

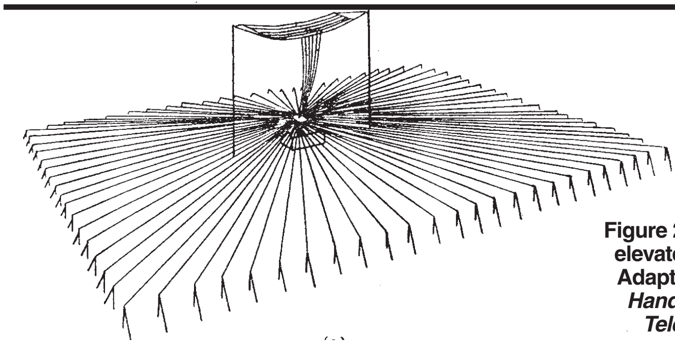


Figure 2 — A very large LF elevated ground system. Adapted from *Admiralty Handbook of Wireless Telegraphy*, 1932.[34]

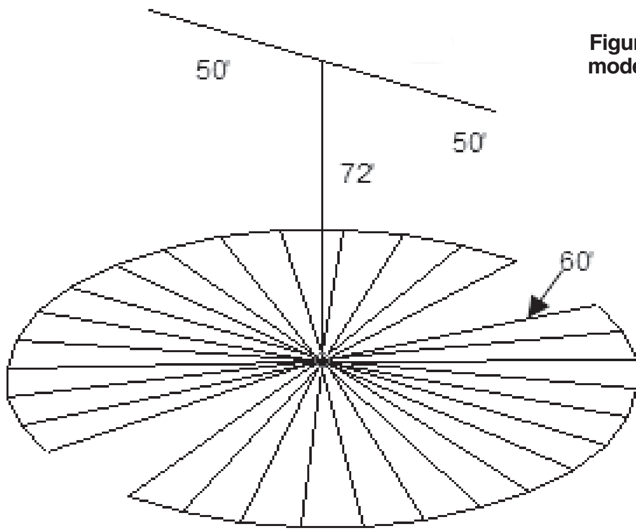


Figure 3 — *EZNEC* model of the 1BCG antenna.

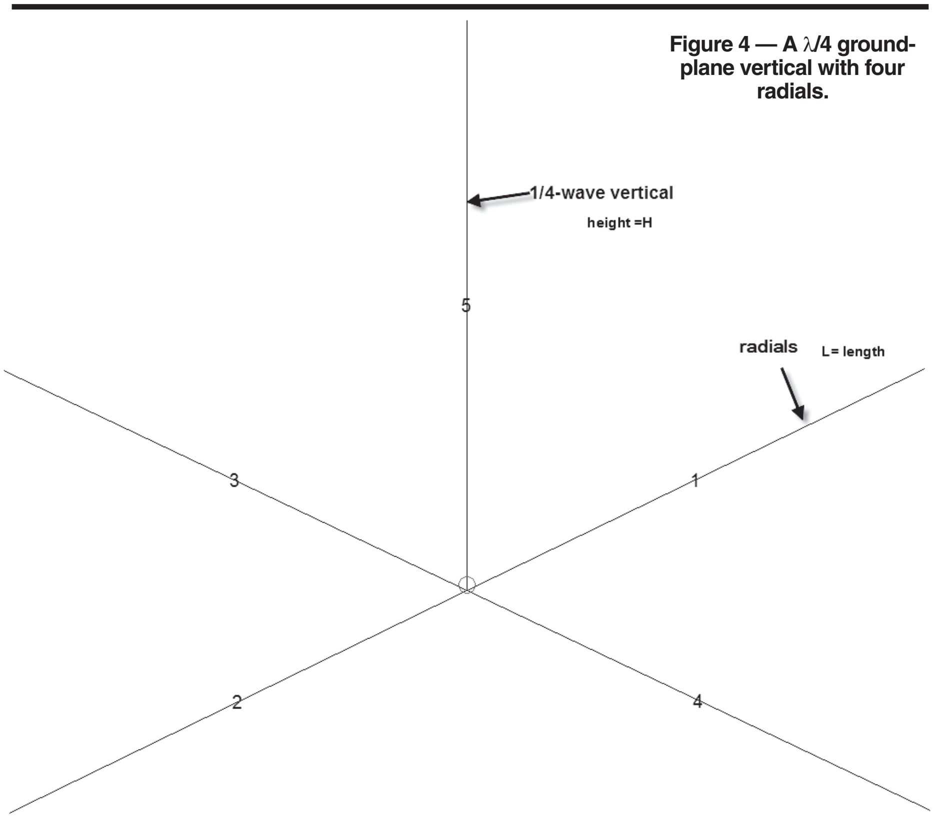


Figure 4 — A $\lambda/4$ ground-plane vertical with four radials.

My original intention for this article was to illustrate the problems introduced by radial fan asymmetries and to discuss some possible remedies. In the process, however, I came to realize that before going into the effects and cures for asymmetries it was necessary to first understand the behavior of ideal systems. Ideal systems can show us when and why they are sensitive and point the way towards possible cures or at least ways minimize problems. The discussion of ideal antennas (over real ground however!) also illustrates a number of subtleties in the design and possibly useful variations that differ somewhat from current conventions.

For these reasons, after some historical examples of elevated wire ground systems, I'll spend a lot of time analyzing ideal systems and then move on to the original purpose of this article: asymmetric radial currents and how to avoid them. At the end of this article I summarize my advice for verticals using elevated ground systems. While much of what follows is derived from NEC modeling, I have incorporated as much experimental data as I could find and compared it to the NEC predictions to see if NEC corresponds to reality.

Prior Work on Elevated Ground Systems

There is a lot of prior information on elevated ground systems: Moxon,[10, 11] Shanney,[13] Laport,[14] Doty, Frey and Mills,[12] Weber,[9] Burke and Miller,[15, 16] Christman,[18 to 33] Belrose[39, 42] and many others. There is also my own work, some published but most not.

Some History

In the early days of radio, operating wavelengths were in the hundreds or thousands of meters. Ground systems with $\lambda_0/4$ radials were rarely practical but very early it was recognized that an elevated system called a "counterpoise" or "capacitive ground," with dimensions significantly smaller than $\lambda_0/4$, could be quite efficient. Note, λ_0 is the free space wavelength at the frequency of interest. Figure 1 shows a typical example of a counterpoise.

Here is an interesting quotation from *Radio Antenna Engineering* by Edmund Laport[14] regarding counterpoises:

"*From the earliest days of radio the merits of the counterpoise as a low-loss ground system have been recognized because of the way in that the current densities in the ground are more or less uniformly distributed over the area of the counterpoise. It is inconvenient structurally to use very extensive counterpoise systems, and this is the principle reason that has limited their application. The size of the counterpoise depends upon the frequency. It should have sufficient capacitance to have* *a relatively low reactance at the working frequency so as to minimize the counterpoise potentials with respect to ground. The potential existing on the counterpoise may be a physical hazard that may also be objectionable.*"

Laport was referring to counterpoises that were smaller than $\lambda_0/4$ in radius. In situations where $\lambda_0/4$ elevated radials are not possible amateurs may be able to use counterpoises instead. Unfortunately, beyond the brief remarks made here, I have to defer further discussion of counterpoises to a subsequent article.

Rectangular counterpoises, some with a coarse rectangular mesh, were also common. A rather grand radial-wire counterpoise is illustrated in Figure 2.

Amateurs also used counterpoises. Figure 3 is a sketch of the antenna used for the initial transatlantic tests by amateurs (1BCG) in 1921-22.[35, 36] The operating frequency for the tests was about 1.3 MHz (230 m). At 1.3 MHz, $\lambda_0/4$ = 189 feet, so the 60 foot radius of the counterpoise corresponds to ≈ 0.08 λ_0.

Note that in all these examples, a large number of radials are used. The use of only a few radials, initially with VHF antennas elevated well above ground, seems to have started with the work of Ponte[37] and Brown.[38]

Behavior With Ideal Radial Fans

In this section we'll look at verticals with a length (H) ≈ $\lambda_0/4$ (λ_0 is the free space wavelength) and symmetric elevated radial systems where the height above ground (J) and the number (N) and length (L) of the radials is varied. We'll also look at the effect of soils with different characteristics from poor to very good. Even though we will be looking at verticals with H ≈ $\lambda_0/4$, keep in mind that elevated ground systems can also be used with verticals of other lengths, with or without loading, inverted Ls, and other antenna types. Elevated radials can also be used with multi-band antennas.

NEC Modeling

Figure 4 shows a typical model of a vertical with a radial system. Except as noted, the following discussion will focus on operation on 3.5 to 3.8 or 7.0 to 7.3 MHz as the operating band and 3.65 or 7.2 MHz as a spot frequency near mid-band. The conductors (both the vertical and the radials) are lossless no. 12 wire. Most of the modeling was done over real grounds. The modeling used *EZNEC* Pro4 v.5.0.45, using the *NEC4D* engine. The use of *NEC4D* over real soils gives the correct interaction between ground and the antenna. Excellent free programs based on *NEC2* are available, but these do not properly model the ground-antenna interaction, so that results obtained from them must be used with some caution.[41] For HF verticals close to ground this is an important limitation.

The Effect of Element Dimensions on Performance

The simplest idea of a ground-plane antenna is that you take a quarter-wave vertical and add four quarter-wave radials at the base. It is well known that the elements of a dipole will be a few percent shorter than λ_0 so it is usually assumed that in a ground-plane antenna the vertical and the radial lengths will also be a few percent less than λ_0. Typically

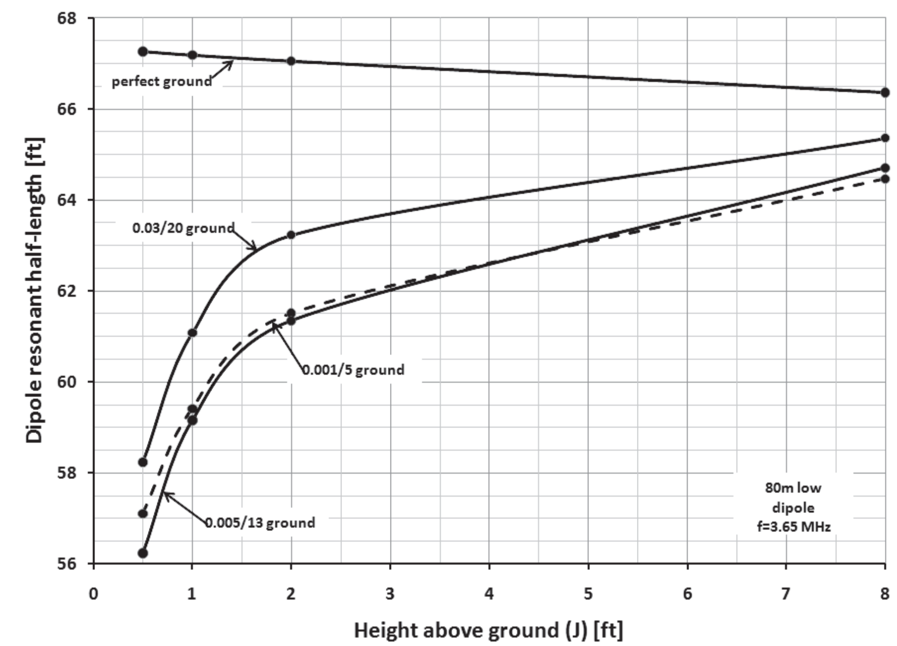


Figure 5 — Dipole half-length for resonance for different values of J and different soils.

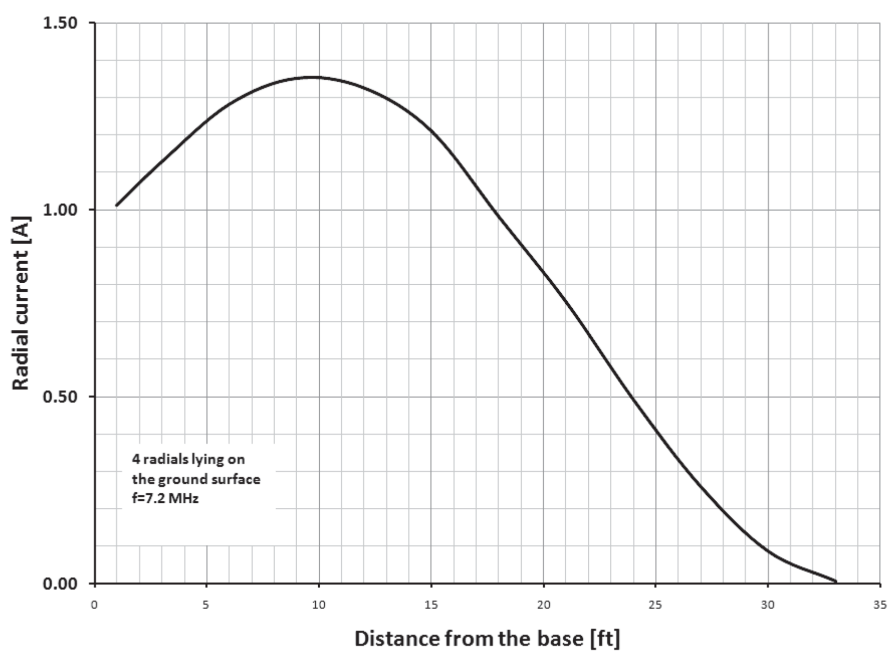


Figure 6 — Measured current on a 33 foot radial at 7.2 MHz. This antenna uses four radials lying on the ground surface.

it is assumed that the vertical and the radials will be *individually* resonant at the operating frequency. Unfortunately it's not that simple, because the vertical is coupled to the radials and both interact strongly with ground because, at least at lower HF (<20 m), the base of the vertical and radial fan will usually be only a fraction of λ_0 above ground. What you have in reality is a coupled multi-tuned system with complicated interactions. It turns out that there are a wide range of pairs of values for H and L that result in resonance, or $X_{in} = 0$ at the feed point (where $Z_{in} = R_{in} + j\,X_{in}$ and Z_{in} is the feed point impedance). Some of these combinations where neither the vertical nor the radials are individually resonant may be useful.

Antenna Resonance and Element Dimensions

The free space wavelength (λ_0) at a given frequency in MHz (f_{MHz}) is given as:

$$\lambda_0 = \frac{299.792}{f_{MHz}}[m] = \frac{983.570}{f_{MHz}}[feet] \quad \text{[Eq 1]}$$

At 3.65 MHz, $\lambda_0/4$ = 67.368 feet. If we model a resonant $\lambda/4$ vertical over perfect ground using no. 12 wire, we find that at 3.65 MHz, $\lambda/4$ = H = 65.663 feet, which is about 3.5% shorter than $\lambda_0/4$.

To take into account the effect of ground on radial resonance for a given value of J and soil characteristic, it has been suggested that we can erect a low dipole at the desired radial height (J) and trim its length to resonance. An example of this is given in Figure 5.

For J = 8 feet, depending on the soil, L varies from 64.5 feet to 66.4 feet. As we reduce J we find that L gets smaller. The shift in resonance for radials close to ground has also been demonstrated experimentally. (See Note 2.) Figure 6 shows the measured radial current at 7.2 MHz on 33 foot radials (sum of four radials). Clearly this radial is $\lambda/4$ resonant at a lower frequency than 7.2 MHz! As Figures 5 and 6 show, the effect gets much larger for small values of J.

What do we mean by "resonant" values for H and L "independently"? It's not just that the reactances cancel at the feed point. When I say "the resonant length for H or L" I'm talking about the case where the current distribution on the vertical and the radials independently corresponds to resonance: in other words, the current just reaches a maximum at either the base of the vertical or at the inner ends of the radials. If either H or L is made longer than resonance, the current maximum will move out onto the radials or up the vertical. Figure 7 shows the current distribution on a vertical and the radials for three combinations of H and L, each of which yield $X_{in} = 0$ at the feed point.

To better understand what's happening we can expand Figure 7 around the 1 A feed point (indicated by the arrow) as shown in Figure 8.

For H = 64 feet and L = 80.85 feet, the current on the vertical has not peaked so the vertical is too short for resonance. The radial current peak is well out on the radials, however, so clearly the radials are too long for resonance. The reactance of the vertical and the radials cancels at the feed point so the antenna is "resonant" but not the vertical and radials individually. Similarly, for H = 69 feet and L =58.8 feet, the current in the vertical peaks and begins to fall (moving from the top to the bottom of the vertical) before the feed point is reached. Again, we have a resonant antenna but the vertical and the radials are not

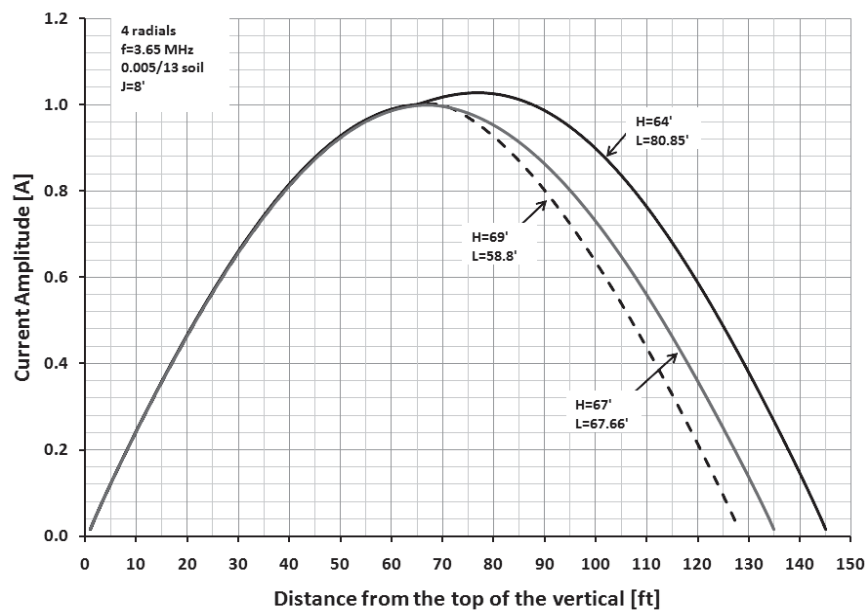


Figure 7 — Current distribution on the vertical and the radials. The current starts at the top of the vertical, runs to the base and then out along the radials. The radial current is the sum of the currents in the four radials. The currents are for 1 A_{rms} at the feed point.

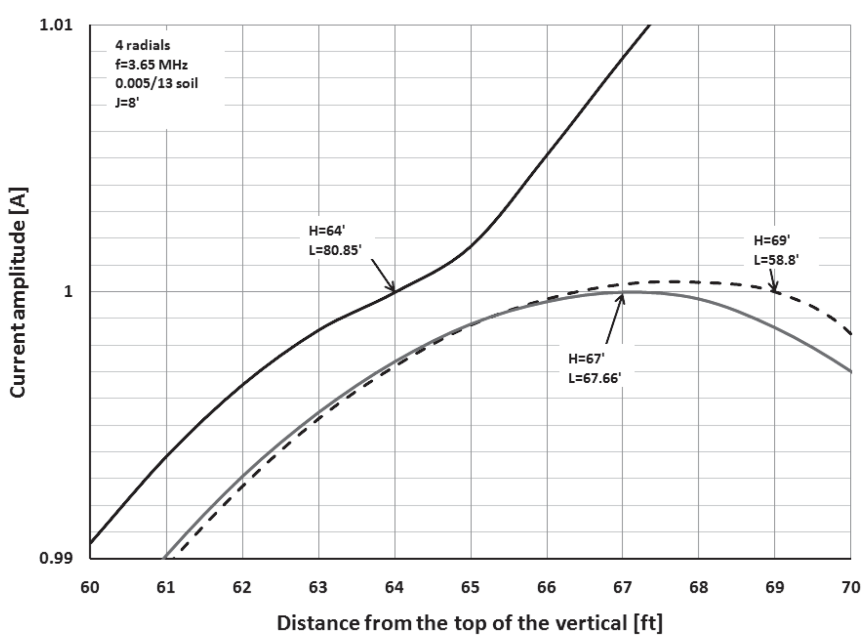


Figure 8 — Current distribution on the vertical and the radials expanded around the feed point. The arrows point to the junctions between the vertical and the radials.

individually resonant. If we set H = 67 feet and L = 67.66 feet, however, both the vertical and the radials are $\lambda/4$ resonant individually.

The "resonant length" (by the definition given above!) of the vertical is 67 feet and the "resonant" length for the radials is 67.7 feet, both of these lengths are substantially different than the value we got earlier for $\lambda/4$ resonance for a vertical over an infinite perfect ground-plane (65.7 feet). The "resonant" radial length of 67.7 feet is quite different from the dipole 8 feet above average ground (64.7 feet). H and L are actually closest to λ_0 (67.4 feet). What we have just seen is only one particular example. If we change J and/or the soil characteristics and/or the number of radials, these lengths will change!

Setting up the antenna so that both the vertical and the radials are individually resonant turns out to not be so simple and we might ask, "Is it really necessary to have both the vertical and the radials resonant individually?" It turns out that there are other considerations besides the current distribution with regard to the choice of L for a given H. It is possible to use values of L where $X_{in} \neq 0$ and compensate for that with a tuning impedance at the feed point for example, or perhaps use some top-loading. In addition, in some situations it may not be possible to have radials long enough to make $X_{in} = 0$ while keeping the radial fan symmetric. Further, Weber has suggested that radials with L <$\lambda/4$ or >$\lambda/4$ are a possible cure for radial current division inequality. (See Note 9.) So we have reasons to investigate the effect of variations in vertical height and radial length on antenna behavior.

For each value of H, number of radials (N), height above ground (J), ground characteristic (σ = conductivity and ε_r = permittivity) and choice of operating frequency, there will be some radial length (L_r) that makes the antenna resonant. That's a lot of variables! So we will look at only a few examples to get a general idea of what happens.

Figure 9 gives an example of the variation in the value for L (L_r) that results in resonance at the feed point ($X_{in} = 0$) as a function of N and several values of H, with fixed values of f, J and soil.

Notice how widely L_r varies with N for most values of H although there is one value for H (66.71 feet) that seems to have only a small variation in L_r as N is changed. Note also how much shorter L_r becomes when H is increased by a few feet. This could be very useful in situations where space for the radial fan is limited. On the other hand note how quickly L_r grows when H is shortened. For N = 16 we see that when H = 64 feet, L_r = 106 feet but for H = 69 feet, L_r is only 39 feet! That's a difference in L_r of almost 3:1. If you cannot make H long enough, all is not lost! A bit of top loading has an effect much like increasing H.

Another way to explore the interaction between L and N is to set L equal to L_r for some value of N (say 16 radials) and while watching the resonant frequency (f_r), vary the number of radials as shown in Figure 10. Note that the most stable f_r is where H = L = 66.71 feet. That is relatively close to the values we got earlier for independently resonant vertical and radials. (Be careful, this is particular to this example; things will vary with different J, ground type, and other variables). Note also that for H a bit tall, f_r decreases as radials are added, but if H is a bit short f_r increases as radials are added. This kind of behavior can be confusing if you are trimming the radials to resonate at a particular frequency, especially if you add some radials. It is possible you could add some radials and then have to make all the original radials longer!

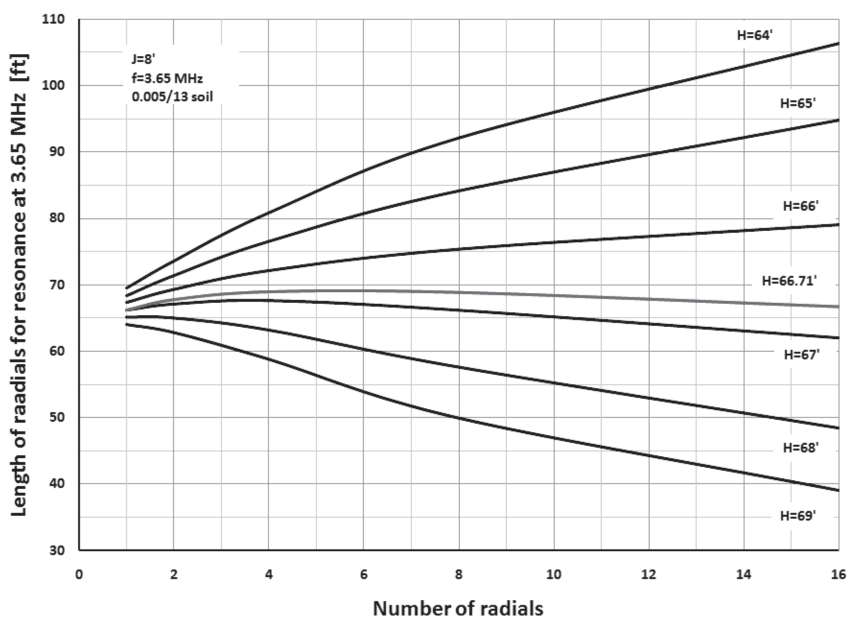


Figure 9 — Examples of the effect of radial number on the radial length for resonance at 3.650 MHz (L_r) for several different values of H.

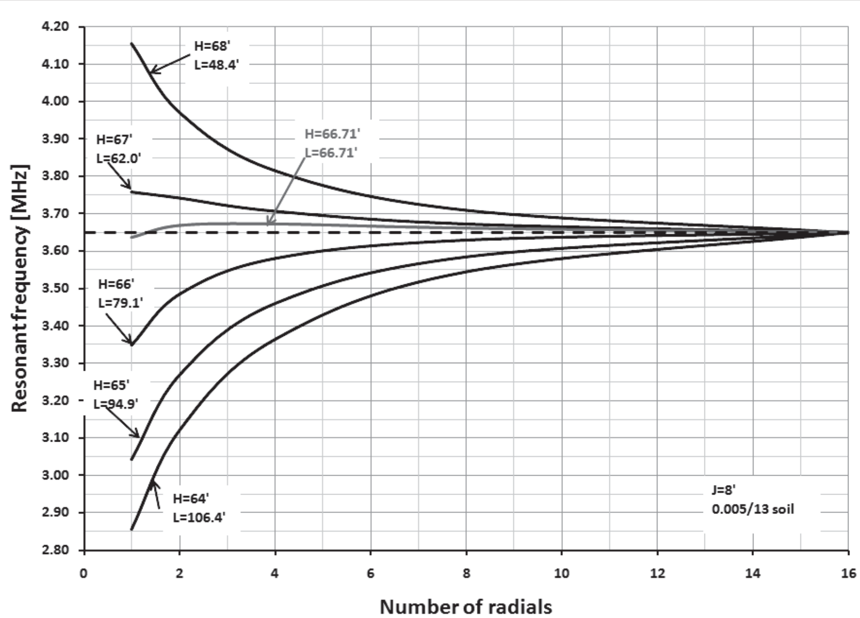


Figure 10 — Resonant frequency of the antenna as a function of radial number for several combinations of H and L that are resonant at 3.650 MHz with N = 16.

This raises the question, "Do real antennas actually behave this way?" During the ground system experiments, I saw exactly this kind of behavior. For the 160 m vertical, f_r went down as I added radials but for the 40 m verticals, f_r went up with radial number. Figure 11 shows graphs of experimental measurements, one for 160 m and the other for 40 m. Real antennas can behave as the modeling predicts.

At this point it's pretty clear that there is considerable interaction between the variables (H, L, J, and so on) but it's not obvious yet if there are optimum combinations (some better than others).

The effect of radial length on efficiency

It turns out that the values for both N and L can have a significant effect on the efficiency of the antenna. Burke and Miller published a very interesting paper in 1989 with the results of NEC modeling of both elevated and buried radial systems for a wide range of N, L, J and soil characteristics.[15] I read this paper many years ago but I have to admit that it did not dawn on me just how much important information was there. Recently the light dawned as I re-read the paper and some additional graphs that Jerry Burke kindly sent me, so I have been redoing some of their modeling. Some of the Burke-Miller graphs were plots of average gain (G_a) versus radial length with radial number as a parameter. G_a is a useful proxy for radiation efficiency in that it gives the proportion of the input power to the antenna that is actually radiated into space. G_a is the ratio of the radiated power (P_r) to the input power (P_{in}) in dB ($G_a = 10 \text{ Log } [P_r/P_{in}]$). All of the power dissipated in the earth, including the near-field losses and reflections in the far-field, are subtracted from the input power. What is actually done is to integrate the power flow across a hemisphere with a very large radius centered on the antenna. The total power flowing through the surface of the hemisphere is P_r. I should emphasize that this is the power radiated towards the ionosphere, power in the ground-wave is considered a loss. For Amateurs, where sky-wave propagation is the norm at HF, this makes sense.

The Burke-Miller graphs used a constant value for H. I will begin with similar graphs but for Amateurs it is more likely that as L is increased H will be decreased to maintain resonance at a given frequency, so I will also show that variation.

Figure 12 is an example of the effect of radial length and radial number on G_a of the antenna when H is kept constant (68 feet in this example).

Figure 12 has some interesting features:

1) Beginning with short values for L, G_a increases slowly up to a maximum. Below maximum, using radials somewhat shorter

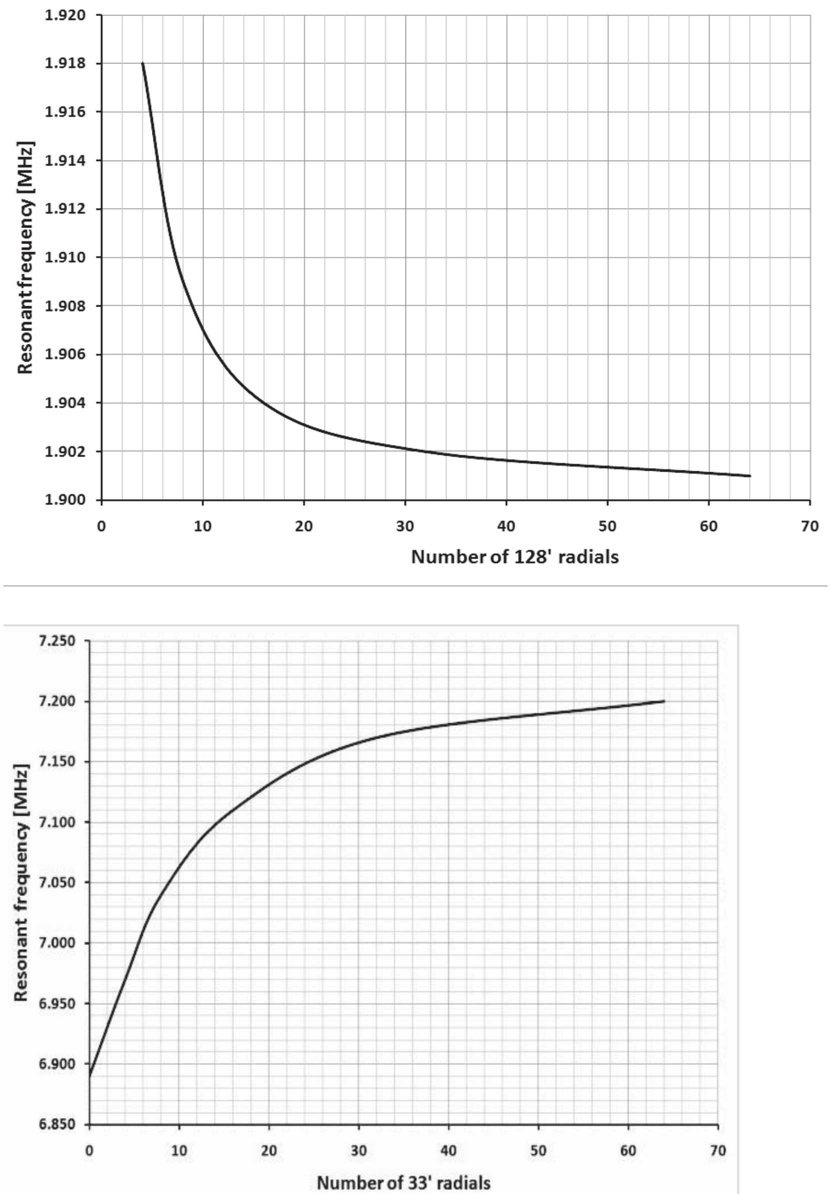


Figure 11 — Experimental measurements of the effect of radial number on resonant frequency.

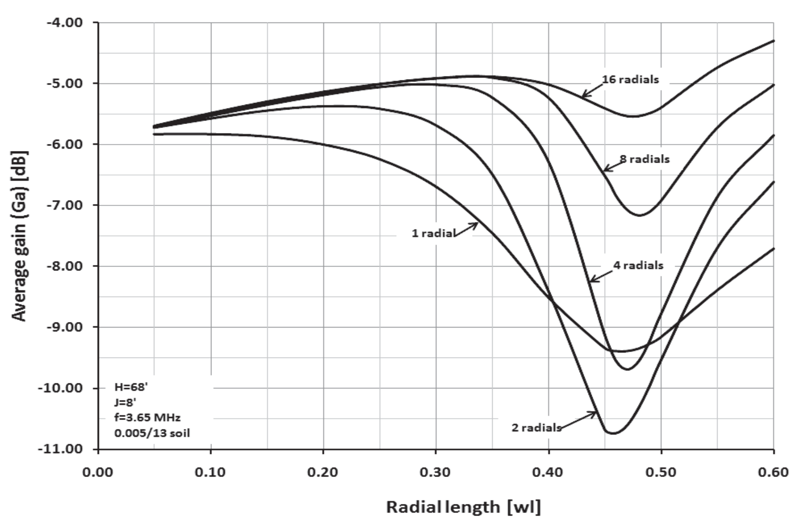


Figure 12 — Average gain as a function of radial length (in wavelengths, λ_0) and number of radials. H = 68 feet, J = 8 feet, f = 3.650 MHz and 0.005/13 soil.

than λ/4 does not seriously reduce the efficiency.

2) Above the maximum, however, there is a large dip! The bottom of the dip can be as much as –7 dB before G_a rises again for longer lengths.

3) Up to the length where G_a starts to fall, increasing N doesn't make much difference in G_a as long as you have four or more radials, but increasing N does push the dip towards longer radial lengths and reduces the depth of the dip.

Figure 12 is for the case where J = 8 feet. If we reduce J, the G_a graphs will change, as illustrated in Figure 13.

As the antenna is moved closer to ground, the efficiency starts to fall, the maximum is lower and the dip gets deeper and occurs at shorter values of L. In fact, if you push J down to 1 inch or less (the case for radials lying on the ground surface) the notch gets even deeper and begins to fall off at lengths well below $\lambda_0/4$. Note, however, that the effect is substantially reduced when larger numbers of radials are used.

One of the suggestions for improving current division between radials was to make them substantially longer than $\lambda_0/4$, in other words, L = 3 $\lambda_0/8$. (See Note 9.) As Figures 12 and 13 show, that's probably not a good idea unless you're using 16 or more radials, but with that many radials current division will already be much improved, as we'll see shortly. Before getting carried away with conclusions we have to ask, "Do real antennas actually behave this way and do we have any experimental verification?" As part of the ground system experiments reported in *QEX* and *QST* (see Notes 1 to 8), I measured the signal strength as N and L were varied with H constant. Figure 14 is a typical result.

I have to admit that during the experiments I did not make the connection between my measurements and the work of Burke and Miller (see Note 15) so I only extended the radial lengths out to slightly less than $\lambda_0/4$. But we can still see the predicted behavior:

1) For short L, the gain rises slowly to a point where it starts to fall.

2) When L is large the dip in gain is large.

3) Increasing N reduces the dip and moves it to larger values for L.

Besides the data shown in Figure 14, I ran spot checks on the gain with sixteen and thirty two 33 foot radials. These were also in agreement with the NEC predictions. I think it's pretty clear that NEC is telling us the truth and we need to pay attention! Radial length is an important consideration.

Figures 12 and 13 are for σ = 0.005 S/m and ε_r = 13, Figure 15 shows the effect of different soil characteristics on G_a for given H, J and N.

As we saw in Figure 6, close proximity

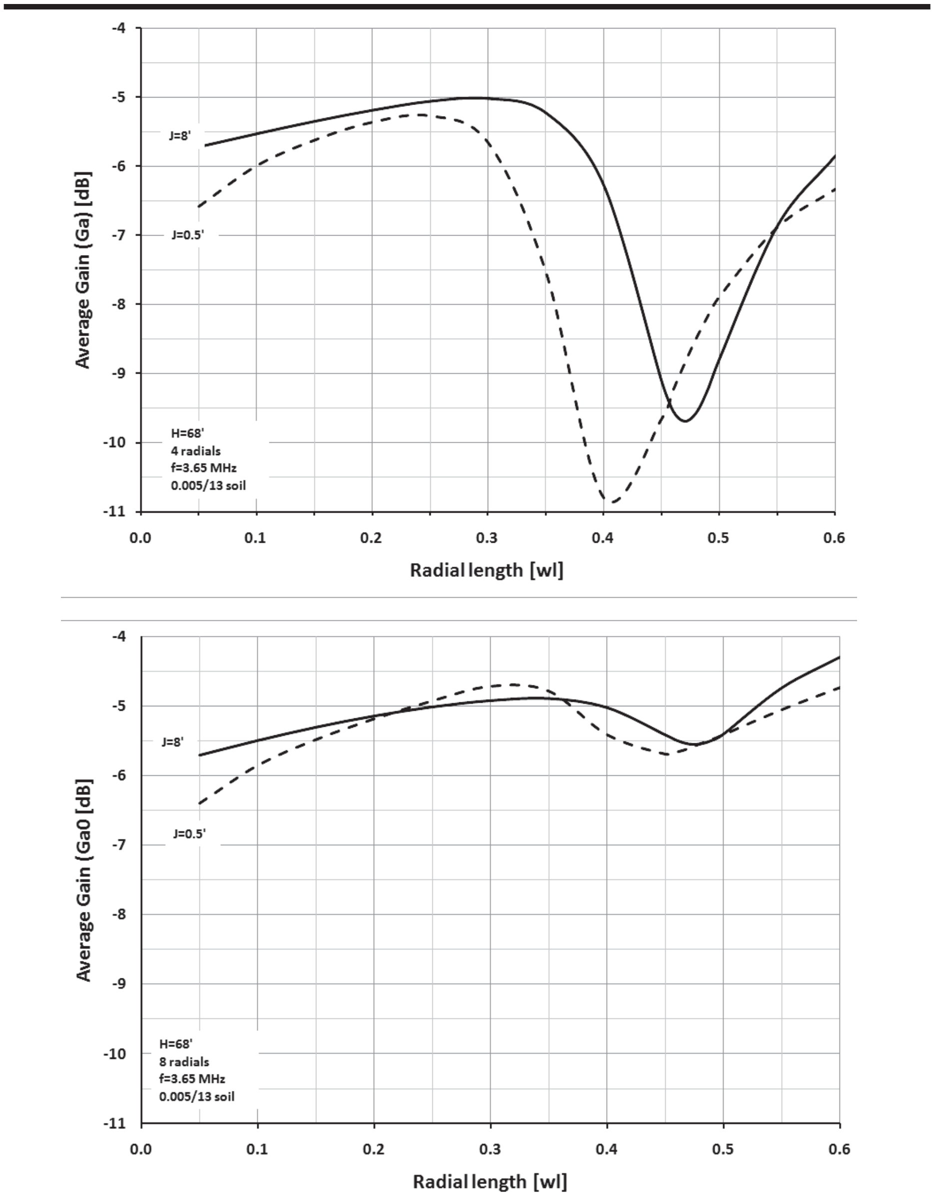


Figure 13 — Comparison of G_a for J = 8 feet and 0.5 feet. N = 4 and 8, and L is in λ_0 = wl.

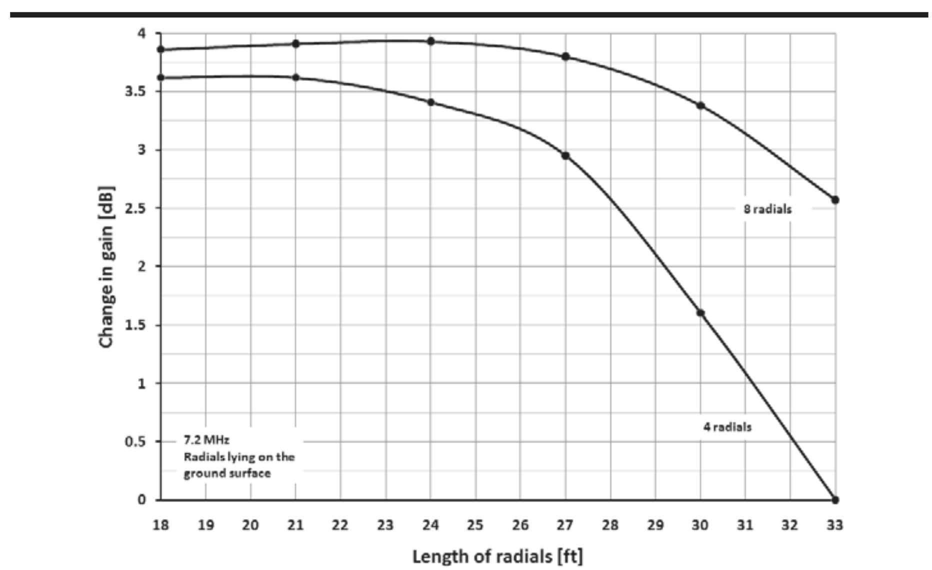


Figure 14 — Far-field change in signal strength as L and N are varied. Radials are lying on the ground surface. f = 7.2 MHz.

to ground has great effect on the radial resonant frequency. John Belrose, VE2CV, has modeled G_a for radials lying close to ground and the effect of different numbers of radials as shown in Figure 16.[42] Note that the data points in the graph were taken from Belrose's article and re-graphed.

The dashed line in Figure 16 represents the case where the lengths of the four radials are adjusted so that the radials are resonant. The predictions in Figure 16 agree with the experimental work shown in Figure 14 showing the effect of shortening the length of radials close to ground. Figure 16 also predicts that even a very small increase in height above ground for the radials will make a large difference in loss, especially if N is small. This large change in G_a with small elevations has been verified experimentally (see Note 3) as shown in Figure 17.

In some cases it may be necessary to use a vertical with H other than $\lambda/4$. Figure 18 shows G_a as a function of L for H = 100 feet ($\approx 3\,\lambda_0/8$), H = 68 feet ($\approx \lambda_0/4$) and H = 34 feet ($\approx \lambda_0/8$) with and without top-loading. Compared to H = 68 feet, the notch for H = 34 feet begins a lower value of L and is much deeper. Putting a short base loaded vertical over an elevated ground-plane may not be a good idea. (Note: this is something that needs to be explored further!) If we add two horizontal top-loading wires that restore the resonance of the 34 foot wire to that of the 68 foot wire, G_a is greatly improved. With the top-loaded vertical, the peak value for G_a is a few tenths of a dB lower than for the full height vertical but that may be acceptable because the vertical is only half as tall. That's something to think about for 160 m verticals. It is also interesting to note that the taller vertical ($H \approx 3\lambda/8$) while more tolerant of longer radials is somewhat less efficient (≈ -0.5 dB). The lesson to draw here is that using elevated ground systems with short verticals can be problematic but really tall verticals may not be all that great either. You have to model the specific situation carefully to make sure you understand what's going on.

The graphs in Figure 12 assume that H is constant. We could also have varied H so that $X_{in} = 0$ for every value of L. This may give us some insight into optimum combinations (with regard to G_a!) of H and L. Figure 19 shows what happens when we do this compared to the case where H was constant for N = 4 and 16. The curves for a fixed H (solid lines) and variable H (dashed lines) are very similar, except that for the four radial case, the dip sets in a bit earlier and is somewhat deeper. The maximum G_a point is about 0.28 λ_0 with four radials and about 0.35 λ_0 with sixteen radials, but in both cases the maximum is very broad. As long as you stay

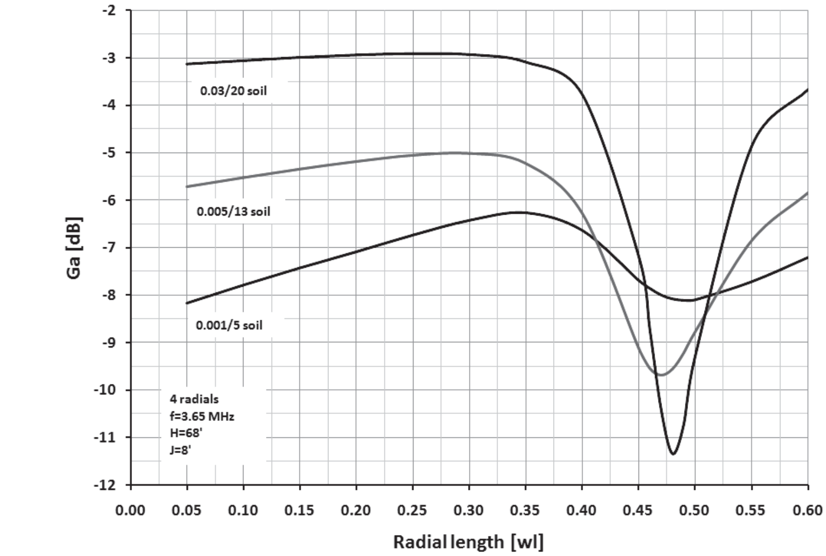


Figure 15 — Effect on G_a of different soils for H = 68 feet, J = 8 feet and N = 4.

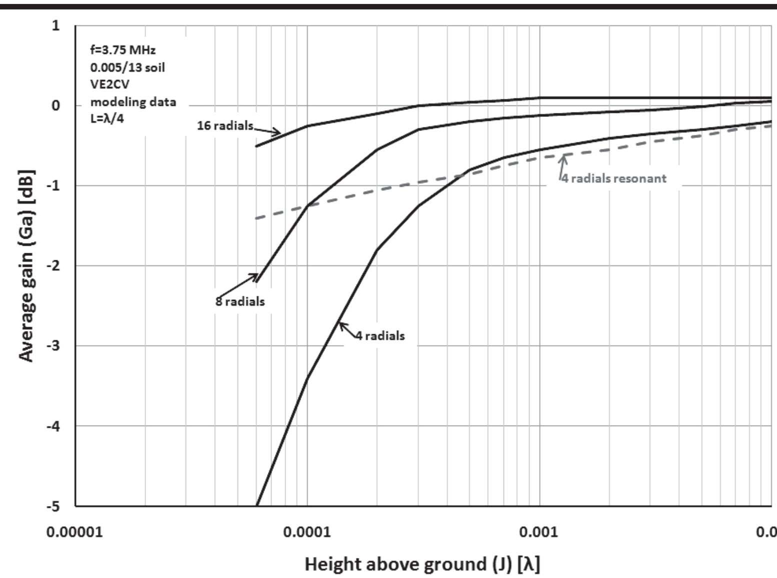


Figure 16 — Average gain when radials are placed close to ground.

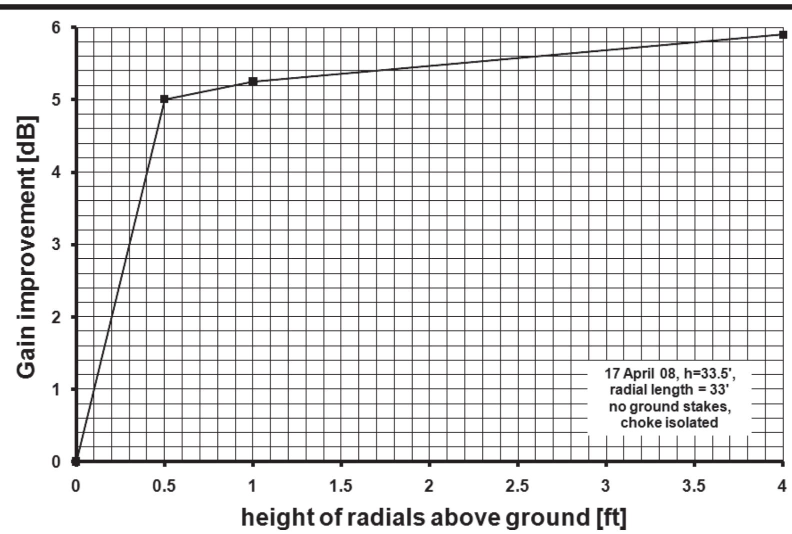


Figure 17 — Measured change in gain as four radials are elevated above ground.

below the point where G_a starts to fall, the value of L is not critical.

Figure 20 shows the values for H that result in resonance at 3.650 MHz for each radial length in Figure 19.

Again we see that the sensitivity to radial length is smaller when more radials are used. We can also look at the effect on R_{in} at resonance as we vary the H + L combination. An example is given in Figure 21.

When four radials are used there is also an important effect on the radiation pattern when the radials are too long.

Figure 22 compares the radiation patterns for two different combinations: L = 0.29 λ_0 and L = 0.46 λ_0. The first is close to the peak G_a value and the second is at the minimum of G_a. In the case of the long radials, not only is G_a much smaller but the peak of the radiation pattern has moved from about 22° to 45°! Clearly if you are using only a few radials, long radials are bad idea.

An Explanation for the Dips in G_a

Why do we see these large dips in G_a for some values of L? We can investigate this by looking at the current distributions on the radials and the associated E and H-field intensities close to ground under the radials. Figure 23 shows examples of the current distribution on the radials as a function of distance from the base (feed point) for several different radial lengths; 64, 70, 80, 100 and 121 feet. The graphs are for N = 4 except for the dashed line, where N = 16 and L = 121 feet.

For the same current at the feed point, with longer radials the currents are much higher as we go out from the base. We would expect these higher currents to increase both E and H-field intensities at ground level under the radials. Using the near-field plotting capability of NEC we can visualize the field intensities as shown in Figure 24.

Figure 24 shows the drastic increase in field intensities with longer radials. In this case I've chosen the longer radial length (121 feet) to correspond to the dip in G_a in Figure 12. Since the power dissipation in the soil will vary with the square of the field intensity, it's pretty clear why the efficiency takes such a large dip when the radials are too long. Figure 25 illustrates what happens to the fields under the radial fan when more radials are employed.

The earlier quotation from Laport stated that the use of more radials would make the fields under the radial fan more uniform. Figure 25 certainly supports that but we can go one step further to show how much the fields are smoothed with more numerous radials. Figure 26 makes that point.

Figure 26 is the E-field intensity just above ground level at points lying on a 90° arc with a radius of 40 feet (centered on the base) for two radial lengths (L = 64 feet and 121 feet) and N = 4 and 16. We can see that with only 4 radials, the E-field peaks sharply directly under the radials but with 16 radials the field is much more uniform.

In Part 2

In the second part of this series, we will examine radial systems for multiband verticals. We also take a look at the effect of various asymmetries in the radial fan.

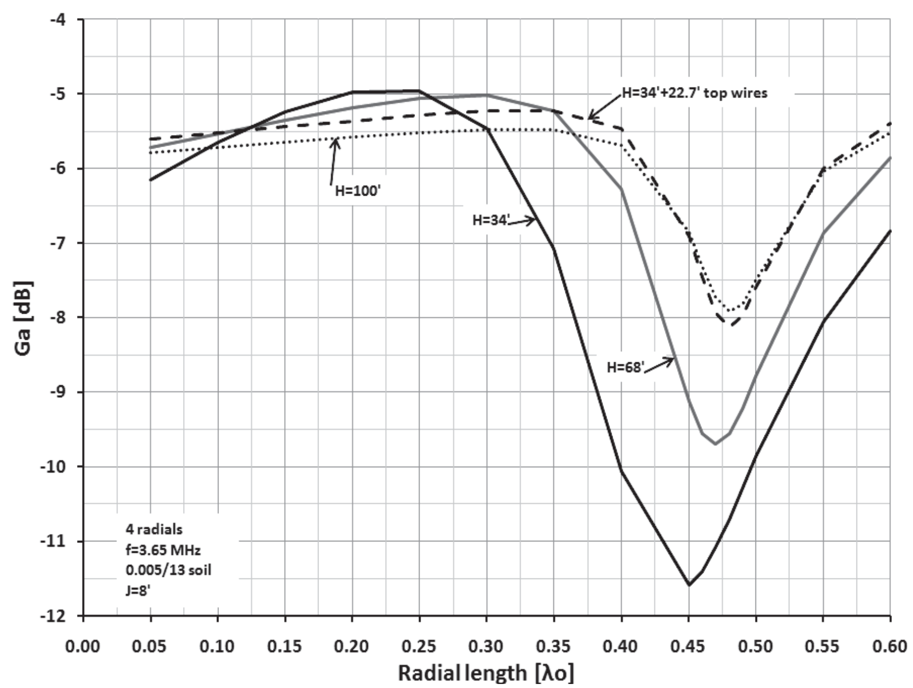


Figure 18 — Effect on G_a of short verticals. H = 100 feet, 68 feet, 34 feet and 34 feet with top-loading.

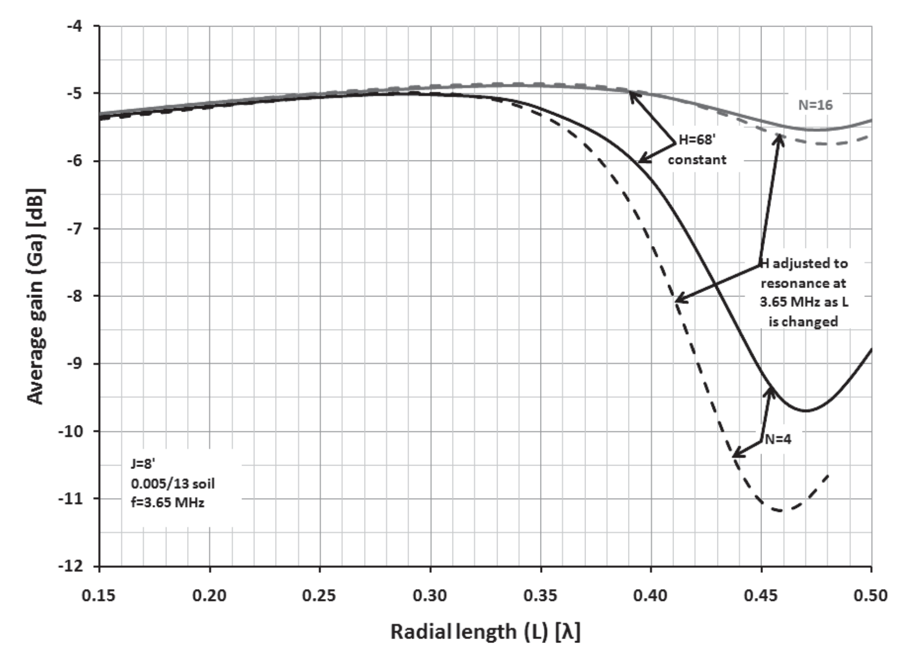


Figure 19 — Effect on G_a of radial length when H is varied to keep X_{in} = 0 at 3.650 MHz compared to the case where H is constant at 68 feet (from Figure 12).

Rudy Severns, N6LF, is a retired electrical engineer (UCLA '66). He holds an Amateur Extra class license and was first licensed in 1954 as WN7WAG. He is a life fellow of the IEEE and a live member of the ARRL. His current Amateur Radio interests are antennas, particularly HF vertical arrays and interactions between towers and arrays. He also enjoys 600 m operation as part of the group under the WD2XSH experimental license. Some of his publications about antennas are posted on his website at ***www.antennasbyn6lf.com****.*

Notes

[1]Severns, Rudy, N6LF, "Experimental Determination of Ground System Performance for HF Verticals, Part 1, Test Setup and Instrumentation," *QEX*, January/February 2009, pp 21-25.

[2]Severns, Rudy, N6LF, "Experimental Determination of Ground System Performance for HF Verticals, Part 2, Excessive Loss in Sparse Radial Screens," *QEX*, January/February 2009, pp 48-52.

[3]Severns, Rudy, N6LF, "Experimental Determination of Ground System Performance for HF Verticals, Part 3, Comparisons Between Ground Surface and Elevated Radials," *QEX*, March/April 2009, pp 29-32.

[4]Severns, Rudy, N6LF, "Experimental Determination of Ground System Performance for HF Verticals, Part 4, How Many Radials Does My Vertical Really Need?" *QEX*, May/June 2009, pp 38-42.

[5]Severns, Rudy, N6LF, "Experimental Determination of Ground System Performance for HF Verticals, Part 5, 160 m Vertical Ground System," *QEX*, July/August 2009, pp 15-17.

[6]Severns, Rudy, N6LF, "Experimental Determination of Ground System Performance for HF Verticals, Part 6, Ground Systems for Multi-band Verticals," *QEX*, January/February 2009, pp 19-24.

[7]Severns, Rudy, N6LF, "Experimental Determination of Ground System Performance for HF Verticals, Part 7, Ground Systems with Missing Sectors," *QEX*, January/February 2010, pp 18-19.

[8]Severns, Rudy, N6LF, "An Experimental Look at Ground Systems for HF Verticals," *QST*, Mar 2010, pp 30-33.

[9]Dick Weber, K5IU, "Optimum Elevated Radial Vertical Antennas," *Communication Quarterly*, Spring 1997, pp 9-27.

[10]Moxon, Les, G6XN, *HF antennas for All Locations*, 2nd edition, Radio Society of Great Britain, 1993.

[11]Moxon, Les, G6XN, "Ground Planes, Radial Systems and Asymmetric Dipoles," *ARRL Antenna Compendium*, Vol 3, 1992, pp. 19-27.

[12]Doty, Frey and Mills, "Efficient Ground Systems for Vertical Antennas," *QST*, Feb 1983, pg 20.

[13]Shanney, Bill, KJ6GR, "Understanding Elevated Vertical Antennas," *Communications Quarterly*, Spring 1995, pp 71-76.

[14]Laport, Edmund A., *Radio Antenna Engineering*, McGraw-Hill, 1952. Note:

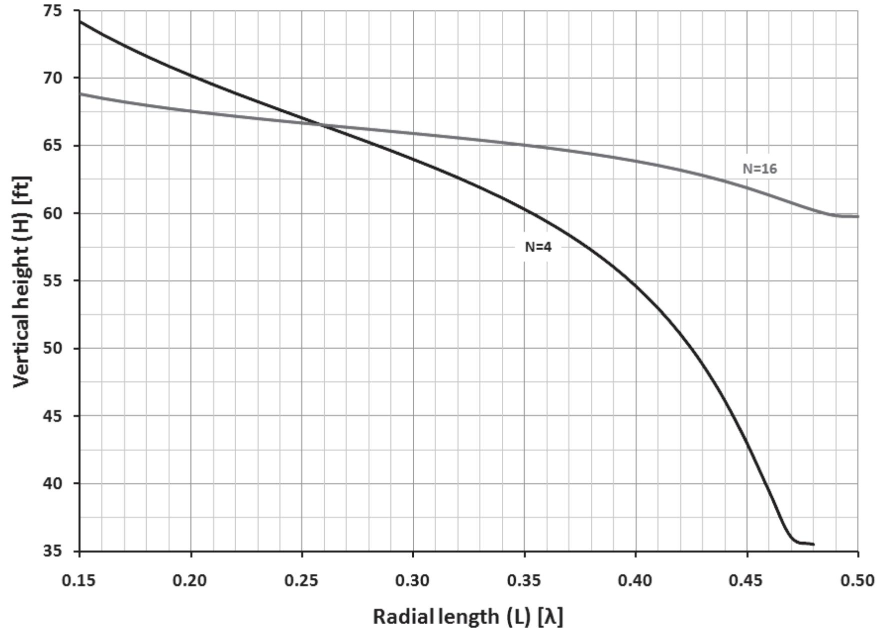


Figure 20 — Values for H that make X_{in} = 0 as L is varied.

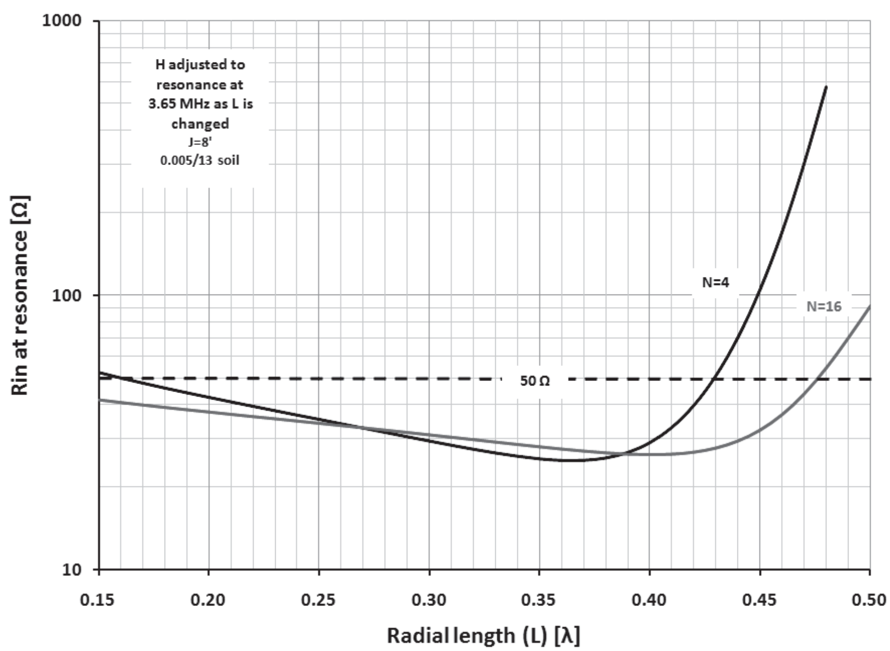


Figure 21 — R_{in} at resonance as a function of L.

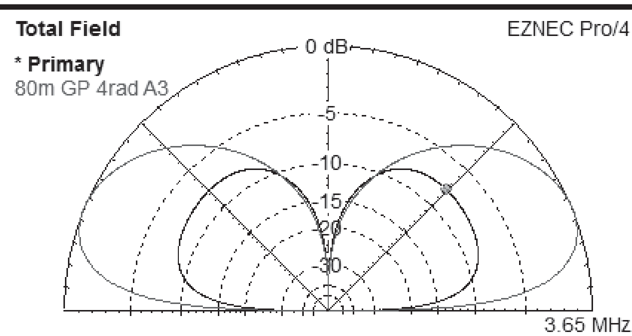


Figure 22 — Radiation pattern for H = 64.64 feet – L = 78.15 feet and H = 39.49 feet – L = 123.96 feet. N = 4 in both cases.

this book is available on-line in .pdf format at **http://snulbug.mtview.ca.us/books/RadioAntennaEngineering/**.

[15]Burke and Miller, "Numerical Modeling of Monopoles on Radial-wire Ground Screens", *IEEE Antennas and Propagation Society International Symposium Proceedings*, June 1989, pp 244-247.

[16]Burke, Weiner and Zamoscianyk, "Radiation Efficiency and Input Impedance of Monopole Elements With Radial-Wire Ground Planes in Proximity to Earth," *Electronics Letters*, 30 July 1992, pp 1550-51.

[18]Christman, Al, K3LC, "Ground Systems for Vertical Antennas," *Ham Radio*, August 1979, pp. 31-33.

[19]Christman, Al, K3LC, "AM Broadcast Antennas With Elevated Radial Ground Systems," *IEEE Transactions on Broadcasting*, March 1988, Vol 34, No. 1, pp. 75-77

[20]Christman, Al, K3LC, "Elevated Vertical-Antenna Systems", *QST*, August 1988, pp 35-42. Additional comments related to this article, *QST* Technical Correspondence, May 1989, pg.50.

[21]Christman and Radcliff, "Impedance Stability and Bandwidth Considerations for Elevated-Radial Antenna Systems," *IEEE Transactions on Broadcasting*, June 1989, Vol 35, No. 2, pp 167-171.

[22]Christman and Radcliff, "Elevated Vertical Monopole Antennas: Effects of Changes in Radiator Height and Radial Length," *IEEE Transactions on Broadcasting*, December 1990, Vol 36, No. 4, pp 262-269.

[23]Christman and Radcliff, "Using Elevated Radials With Ground Mounted Towers," *IEEE Transactions on Broadcasting*, September 1991, Vol 37, No. 3, pp 77-82.

[24]Christman, Zeineddin, Radcliff and Breakall, "Using Elevated Radials in conjunction with Deteriorated Buried-Radial Ground Systems," *IEEE Transactions on Broadcasting*, June 1993, Vol 39, No. 2, pp 249-254.

[25]Christman and Radcliff, "AM Broadcast Antennas with Elevated Radials: Varying the Number of Radials and Their Height Above Ground," *IEEE Transactions on Broadcasting*, 1996.

[26]Christman, Al, K3LC, "Elevated Vertical Antennas for the Low Bands: Varying the Height and Number of Radials," *ARRL Antenna Compendium*, Vol 5, 1996, pp 11-18.

[27]Christman, Al, K3LC, "Dual-Mode Elevated Verticals," *ARRL Antenna Compendium*, Vol 6, 1999, pp 30-33.

[28]Christman, Al, K3LC, "Elevated Vertical Antennas Over Sloping Ground," ARRL Antenna Compendium, Vol 6, 1999, pp 189-201.

[29]Christman, Al, K3LC, "'Gull-Wing' Vertical Antennas," *National Contest Journal (NCJ)*, Nov 2000, pg 14.

[30]Christman, Al, K3LC, "Using Elevated Radials With Grounded Towers," *IEEE Transactions on Broadcasting*, Vol 47, No. 3, September 2001, pp 292-296.

[31]Christman, Al, K3LC, "A Study of Elevated Radial Ground Systems for Vertical Antennas," Part 1, *National Contest Journal (NCJ)*, January/February 2005, pp 19-22. Part 2, *NCJ*, March/April 2005, pp 17-20.

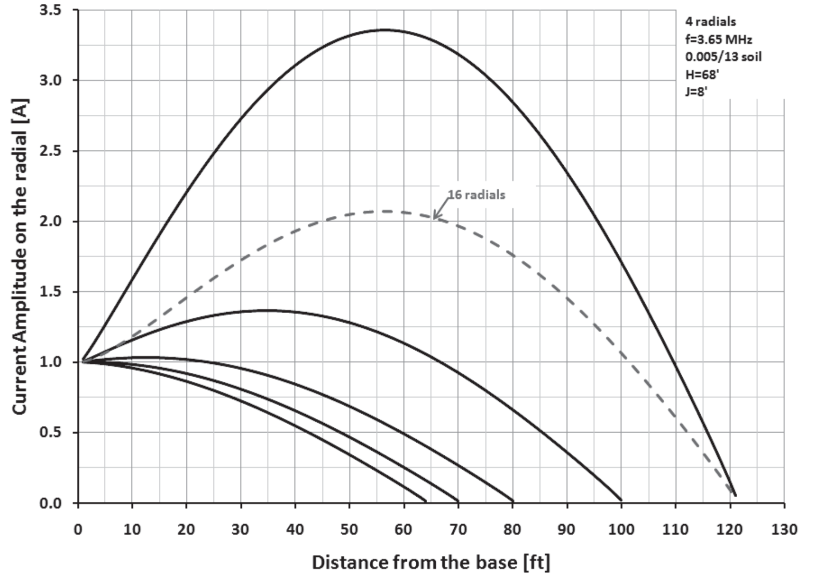


Figure 23 — Radial current distribution as a function of distance from the base. N = 4, H = 68 feet, f = 3.65 MHz, J = 8 feet and average soil.

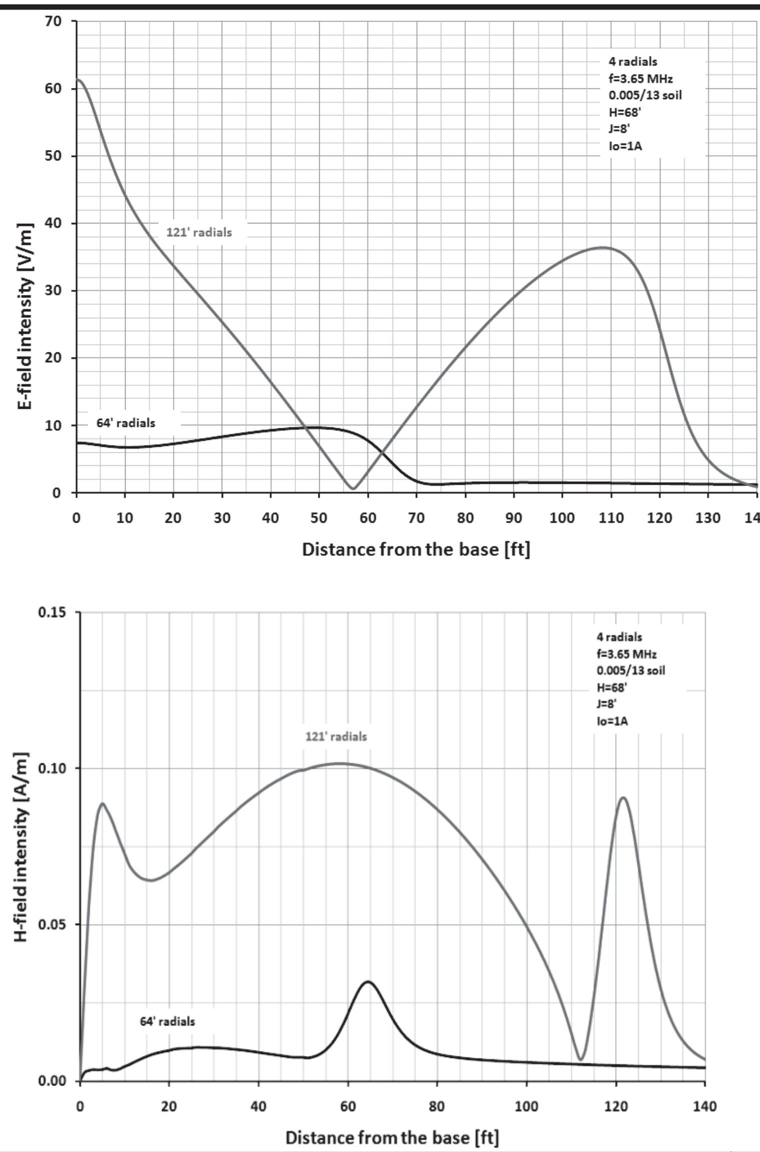


Figure 24 — E and H field intensities close to the ground surface directly below the radials with N = 4.

Part 3, *NCJ*, May/June 2005, pp 17-19.

[32]Christman, Al, K3LC, "Compact Four-Squares," *NCJ*, vol 32, No. 4, Jul/Aug 2004, pp 10-12.

[33]Christman, Al, K3LC, "Verticals by the Sea," Part 1, *NCJ*, July/August 2005, pp 9-12. Part 2, *NCJ*, September/October 2005, pp 4-6. Part 3, *NCJ*, November/December 2005, pp 17-20.

[34]*Admiralty Handbook of Wireless Telegraphy, His Majesty's Stationary Office*, 1932, pg 799, Figure 452.

[35]Burghard, George, "Station 1BCG," *QST*, February 1922, pp 29-33.

[36]Kelley and Hudson, "Hams Span the Atlantic on Shortwave!" *QST*, December 1996, pp 28-30.

[37]Ponte, Maurice, U.S. patent 2,026,652, "High Frequency Transmitter," 7 January 1936. French patent No. 764.473 (1933) and UK patent No. 414,296 (1934).

[38]Brown and Epstein, "An Ultra-High Frequency Antenna of Simple Construction," *Communications*, July 1940, pp 3-5.

[39]Belrose, John, VE2CV, "Vertical Monopoles with Elevated Radials: An Up-date," *10th International Conference on Antennas and Propagation*, April 1997, IEE conference publication #436, pp 1.190-1.195.

[40]Severns, Rudy, N6LF, letter to *QST* Technical Correspondence, awaiting publication, but available at **www.antennasbyn6lf.com**.

[41]See **www.4nec2.com**.

[42]Belrose, Jack, VE2CV, "Elevated Radial Wire Systems For Vertically Polarized Ground-Plane Type Antennas, Part 1-Monopoles," *Communications Quarterly*, Winter 1998, "Part 2-Phased arrays," *Communications Quarterly*, Spring 1998.

[43]Weber, Dick, K5IU, private communication, July 2011.

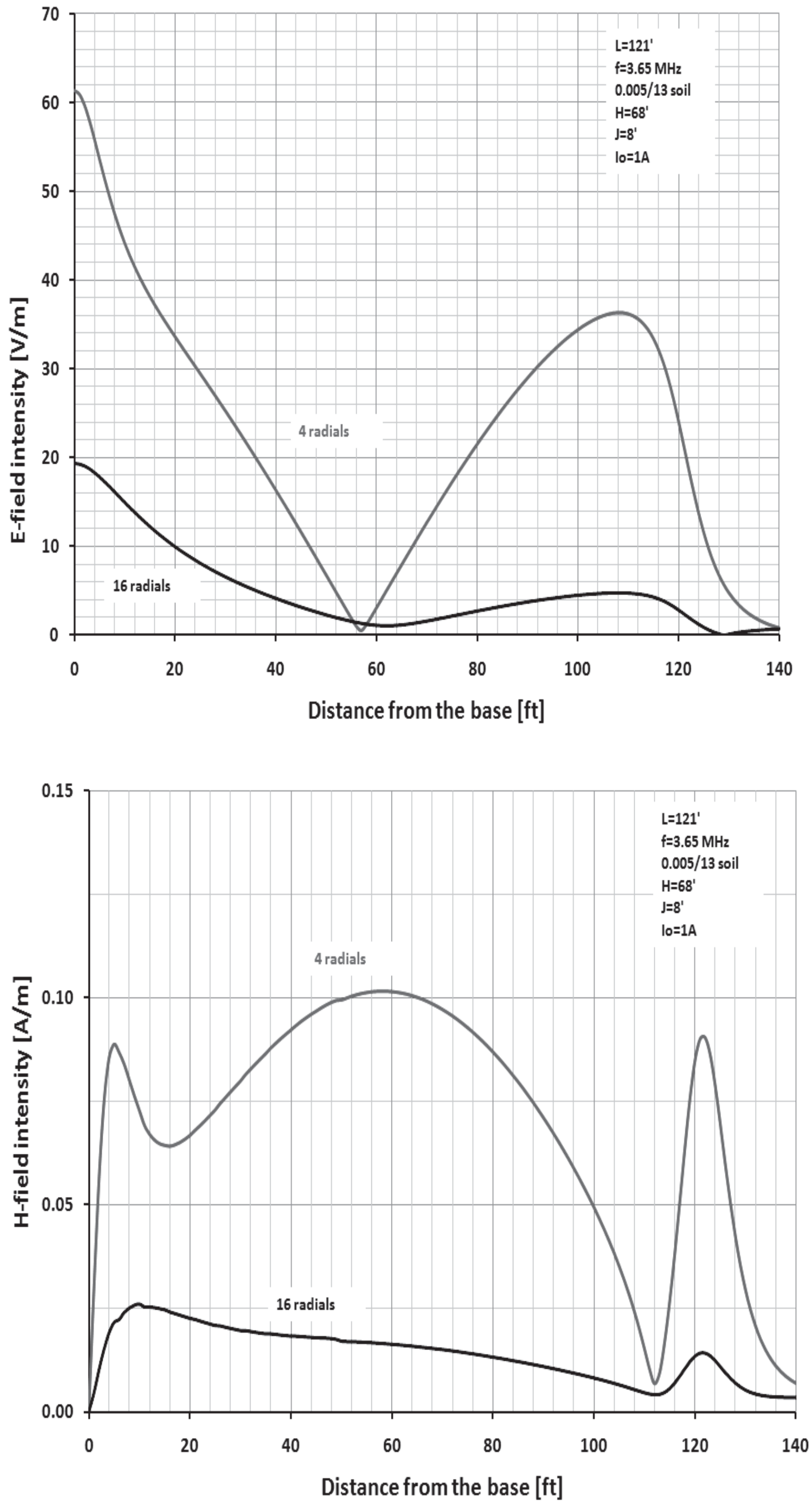


Figure 25 — E and H field intensities close to the ground surface directly below the radials. N = 4 and16.

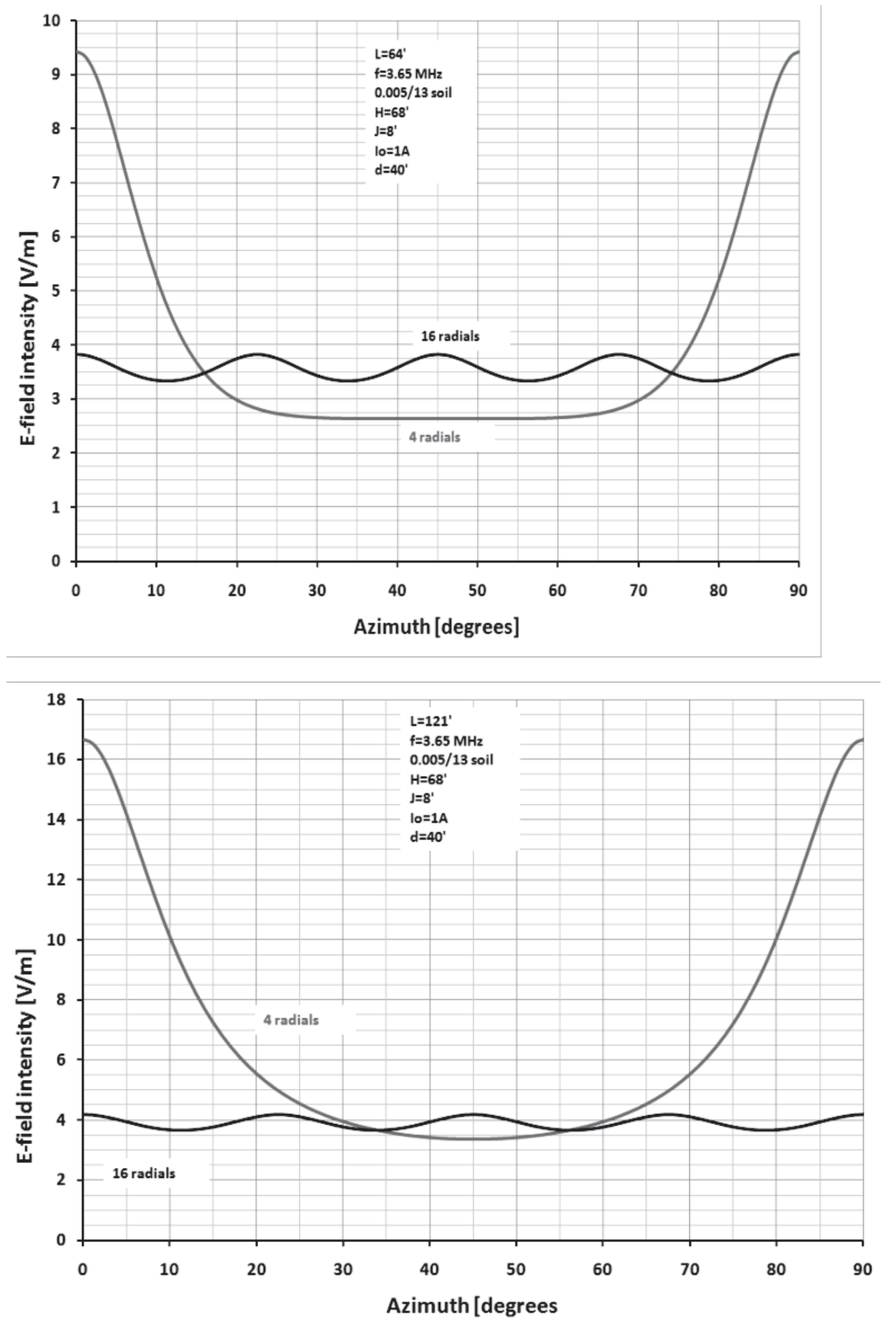


Figure 26 — E-field intensity just above ground on a 90° arc 40 feet from the base.

QEX

Ray Mack, W5IFS

17060 Conway Springs Ct, Austin, TX 78717; w5ifs@arrl.net

SDR: Simplified

We look at more work with the TI eZDSP board.

Software Breakthrough

If you looked at the *QEX* website just after the new year, you found that I had new software files available for download. The *DirectX* code allowed us to generate the DDS sine wave and a reasonable FM generator. There are still issues getting *Windows* to really do the kind of work we want it to do. I spent a lot more time going over the support documentation from TI and Spectrum Digital, and located code that exactly fits our needs. When bringing up new hardware, it is best to write a small test program to exercise the bare minimum of functions for each peripheral device. The Spectrum Digital site has a set of test programs that implement exactly the type of programs that interest us. The programs are actually copyrighted by TI and carry a right to distribute, so they are available in the zip file for this issue on the ARRL *QEX* files website.[1]

The most important program is the Codec test program. TI has implemented a test program very much like the one I produced in *DirectX*. It produces a sine wave tone in the headphones for five seconds and then loops the microphone input back out to the headphones. It is then a simple matter of programming to implement the DDS FM transmitter using the aic3204 test directory. I installed all of my TI software development tools in the C:\TI directory. I downloaded the Spectrum Digital test files and installed them in C:\TI\Spectrum_Digital\ezdsp5535_v1.

TI Development Software

I walked you through installing *Code Composer Studio* and the BIOS in the Nov/Dec 2011 installment, so we are ready to jump into building one of the Spectrum Digital test programs. These programs are "bare metal" programs that run without an operating system. The one of interest to us is the aic3204 directory.

For the most part, *Code Composer Studio* is intuitive and similar to other integrated development environments you might have used before like *Visual Studio* or *AVR Studio* from Atmel. Projects are the one area where the interface is different and not obvious. Most environments have a menu item called "Project" where all project related activities occur or at least an "Open Project" entry under the "File" menu. That is not the case with *Code Composer*. The Project menu entry is used only for building the active project and can be done more easily by just clicking an icon on a tool bar. The project information is empty when you first start *Code Composer*.

Add a project to *Code Composer* by selecting "File/Import…" and walking through the dialogs. Figure 1 shows the initial Import dialog box. Select "Existing CCS/CCE Eclipse Projects" then "next." Figure 2 shows the dialog box that allows you to select the projects you want. You can see from the dialog that all the test programs under the tests directory have been selected, which is the default condition. Any projects you leave selected will be added to the project environment in *Code Composer* when you select "Finish." This is a nice feature that I have not seen in other environments. One of the panes of *Code Composer* shows a listing of all the projects you have selected. Only one project is active at a time. Select a new active project by right clicking a project and selecting "Set as active project." You can show a tree structure of all the project parameters, or leave it collapsed.

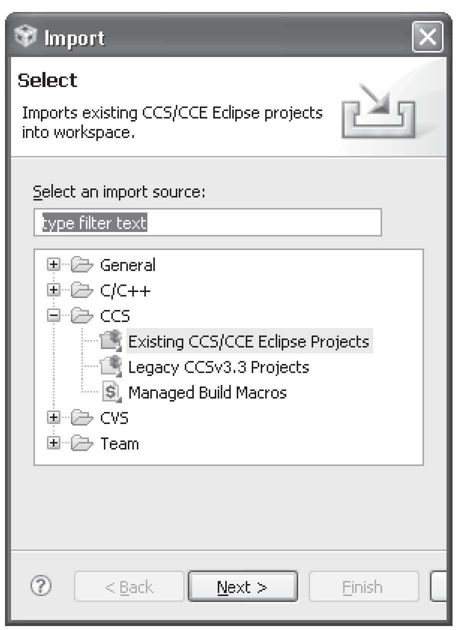


Figure 1 — The "Import…" dialog screen, which is the first step in adding a new project to the environment.

So, now we have added the aic3204 directory and made it our active project. The next step is to build the project and start debugging. The "Build Active Project" and "Rebuild Active Project" tool buttons are next to the print tool button. Either one will build the project. You are building a distribution project, so you should have no errors. Connect the earbud/microphone set to the phone jacks on the eZDSP Stick and plug the USB A connector into your computer. The board will power up and display the demo text on the display and light the status LEDs. Press the debug tool button (it looks like a green tick rather than an insect). If this is your first time to run the debugger, it will pop up a dialog to allow you to configure the debugger. You should select a shared location that suits your needs. You need to select the XDS100v2 debugger from the drop down menu. Check the box for the eZDSP5535 and save the configuration. This is the last time you will need to perform this step.

The debugger sets the execution point to the prolog of main(), so you can start single stepping from that point. This is another place where *Code Composer* is different from other environments. If the debug pane has not launched, you need to display it from the View menu. The tool buttons are part of the debug pane rather than a tool bar on the main *Code Composer* window. Most environments call the debug menu "debug"; *Code Composer* calls it "Target" instead. The tool tips for the tool buttons do not say which F key does the same task, so you have to open the Target menu if you do not remember. F5 is the "step into" key, F6 is the "step over" key, and F7 is the "step out of" function key.

The second line of main() is a call to printf(). Our Blackfin Stamp required a terminal emulator on the PC and hardware for an actual serial terminal. *Code Composer* uses the JTAG to implement a virtual console that shows up at the bottom of the *Code Composer* window. It is the same window that displays the progress of gmake as the project is built. This console can also be used for user input for your programs with scanf() for *C* or cin for *C++*. The console provides a handy way to include status output for your programs.

I have not found any documentation that describes how the debugger coexists with the BIOS code that is running in

[1]The software files described in this Column are available for download from the ARRL *QEX* files website. Go to **www.arrl.org/qexfiles** and look for the file **3x12_Mack_SDR.zip**.

parallel with the code we are debugging. The BIOS is obviously still running because the demo string still scrolls on the display. As long as it does not consume too much CPU, we can work as if we own the entire CPU.

Place the earbuds in your ears and press the Run button. You will hear the tone. I did not hear the microphone when I ran the code, but my revised code to continually loop back worked just fine. You will notice a strange function StopTest() at the end of all of the test programs. This is required by the debugger to force a software interrupt at the end of useful work in the program. Without that call, the system would continue with unpredictable results. You can leave out a call to StopTest() if your program is an infinite loop that will not exit from main().

Hardware Resources

We only need to concern ourselves with the TMS320C5535 and the AIC3204 codec for our experiments. I have reproduced the codec portion of the schematic in Figure 3. The codec has many resources that would make a nice sound card. It has three differential line input ports, which can sample as fast as 200 kHz. It has two single ended stereo outputs. One is a line level output, which can handle signals sampled as high as 200 kHz. The other

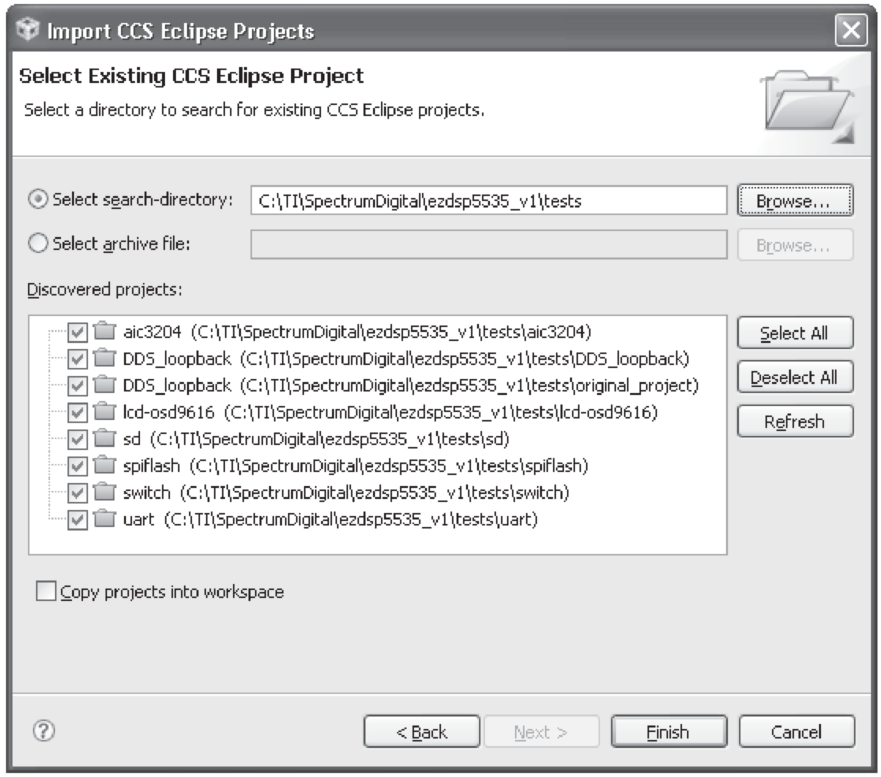


Figure 2 — The selection dialog that displays all of the potential projects in a directory. All projects that remain checked when pressing the "Finish" button will be added to the environment.

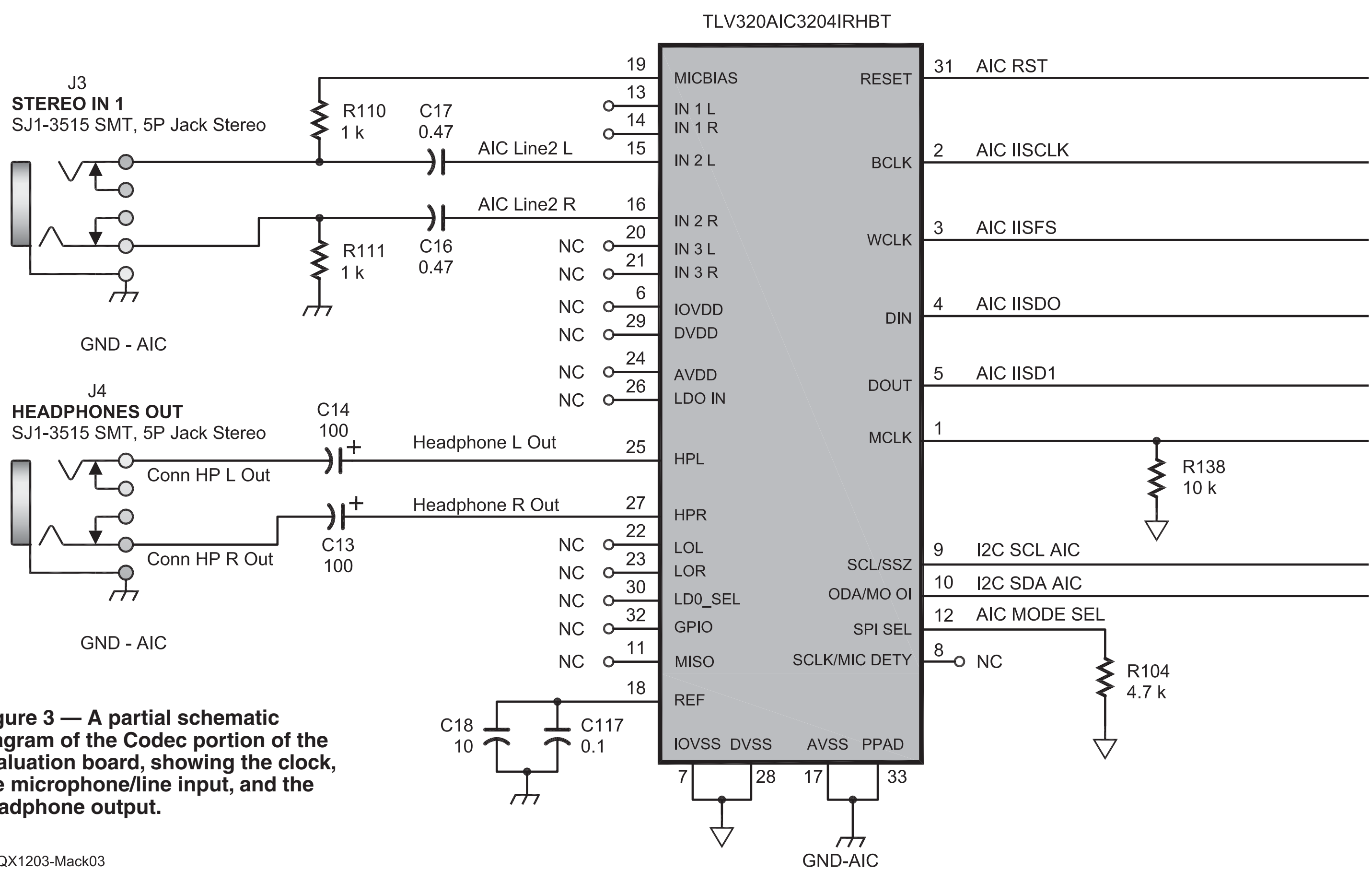


Figure 3 — A partial schematic diagram of the Codec portion of the evaluation board, showing the clock, the microphone/line input, and the headphone output.

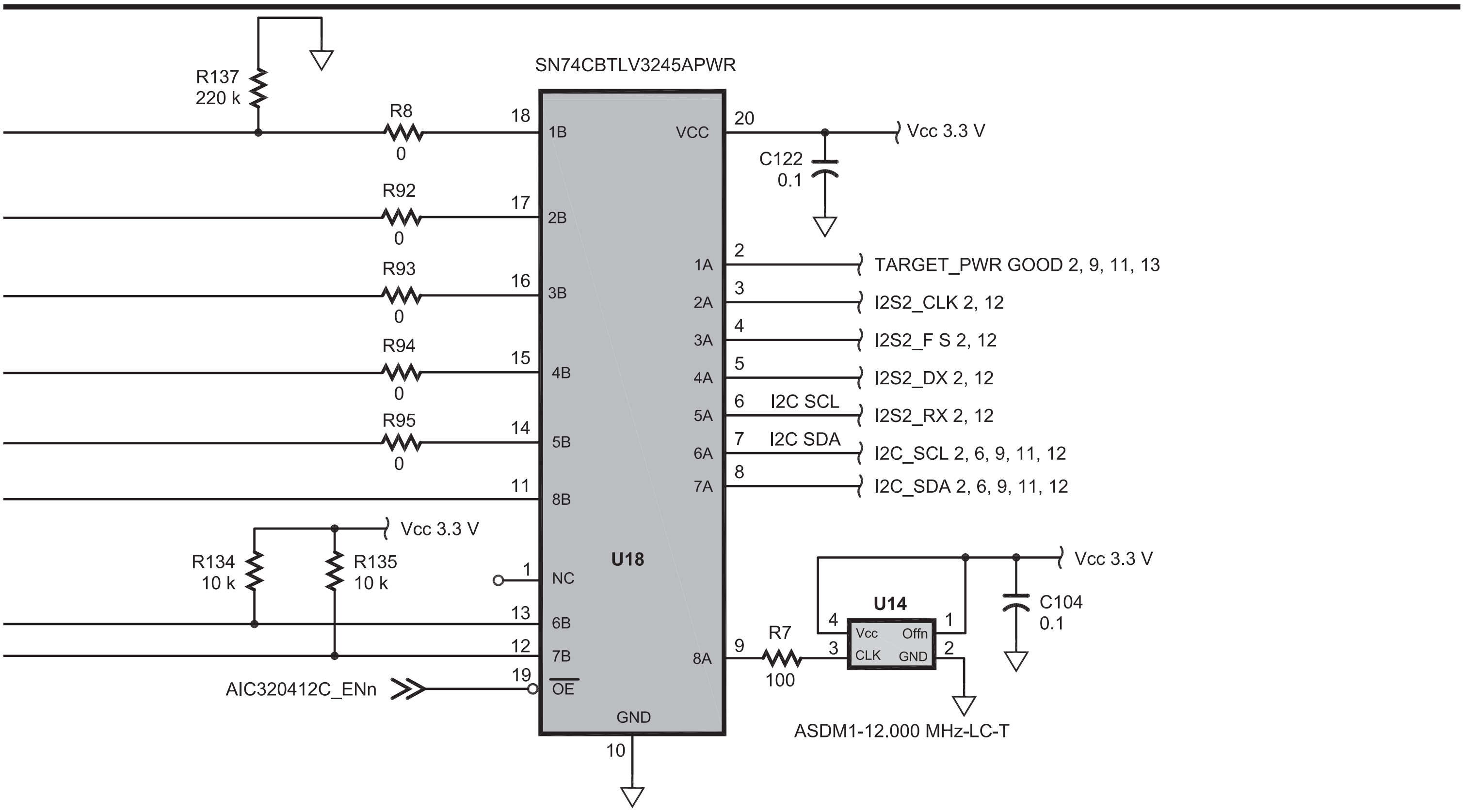
is a single ended stereo power amplifier intended to drive headphones to 40 mW in high power mode. Even though the DAC can be sampled at 200 kHz, the headphone amplifier is not intended to drive signals above 20 kHz. The codec implements decimation filtering on the ADC input and low-pass filtering on the DAC output. The filtering limits the input and output frequency ranges to around 80 kHz when sampled at 192 kHz. We can probably use the input connector for frequencies as high as 80 kHz or 90 kHz. The output connector is hooked to the headphone output of the codec, so we can probably only output frequencies up to about 20 kHz or so. The line out pins are not routed where we can access them easily, so there is no easy way to modify the board for higher frequency output.

The codec interface with the CPU is handled by two different buses. The audio input and output is handled by the I2S (Inter-IC Sound) bus. The I2S bus runs at very high speeds (more than 9.2 MHz when sampling at 192 kHz and 24 bits per sample). The I2S bus is implemented in full duplex, so input and output can happen simultaneously. The configuration registers are programmed using the I2C (Inter-Integrated Circuit) bus which runs at 400 kHz.

The Bare Metal DDS Transmitter Programs

The aic3204 loopback test makes an excellent platform for our experiments. It only needs one modification to fully fit our needs: the I2S library needs to be modified so that the input sampling drives both input and output. The function EZDSP5535_I2S_readLeft() waits for the I2S port to read a left sample. Once that sample is read, the right sample will be guaranteed to be ready. Likewise, the next left and right output samples can be placed in the buffer since the next left input sample cannot occur until enough time for a left and right sample has passed. The source code for the ezdsp5535bsl.lib is included in the test program distribution, so we can modify ezdsp5535_i2s.c so that the function EZDSP5535_I2S_writeLeft() does not wait on an interrupt.

I ported the DDS tone generator program almost unchanged (at least with respect to signal processing) from the *DirectX* PC version. It is included in the zip file of the code. (See Note 1.) I wanted to maintain the aic3204 test code directory in its original condition. That meant that I needed to create a new directory and copy over the necessary files. That also requires a new set of project files. This is an area where *Code Composer* is not very helpful. There is no tool for letting you clone a project. Fortunately, all of the files are plain text files that you can edit inside *Code Composer* or an editor of your choice. To create a new project, copy the .project, .cdtproject, .cdtbuild, and .cssproject files into your new directory. The .project file is easy since you just need to replace the old name (aic3204 in my case) with your new directory name. The .cdtproject file does not require modification. The .cdtbuild file is the one that requires the most skill to modify by hand. It references all of the source files in the old project. You need to remove all lines with the string "resourceConfiguration" and the lines in between. Each entry starts with a line with "<resourceConfiguration" and ends with a line containing "</resourceConfiguration>" by itself. The next step is to replace the old project name with your new directory name in the rest of the file. This file is the important one that holds all of the compiler options and the linker options. The .cssproject file also only has one entry of the old project directory to modify. You must also copy the file lnkx.cmd since it describes the memory layout



Decimal values of capacitance are in microfarads (µF); others are in picofarads (pF);
Resistances are in ohms; k=1,000, M=1,000,000.

for the executable file. This file is identical in all of the test programs. The last step is to create all of the source files for your new project and store them in the new directory. This can be as simple as creating the comments for the file header and storing the files in the new directory. *Code Composer* insists on including all files in your directory in the project and any files in subdirectories below, so you do not want to keep extraneous files in your project directory. Once you have the files you want stored in the directory, you can go to "File/Import…" and navigate to the directory that holds your project. Selecting that project and "Finish" will bring your new project in as the active project. It is important to copy and modify the project files because they contain the names of the *include* directories and the libraries. You can create an empty project using the new project wizard, but tracking down all of the directories and libraries is a time consuming process.

I separated the aic3204_loop_linein() function into a codec setup function and a sample loop function. The setup parameters do the work necessary to turn on power to the proper pieces and to set up the PLL to run the ADC and DAC at the 48 kHz sample rate based on the 12 MHz crystal. The data sheet lists the various parameters necessary to set the proper sample rate. The hardware is designed to take the 12 MHz crystal oscillator and place it on the MCLK pin. MCLK is the reference for the PLL. The data sheet does not give examples of other sample rates and the font for the PLL frequency equation is deceiving. This is the equation:

$$\mathrm{PLL\_FREQ} = \frac{\mathrm{PLLCLK}\times\left(\mathrm{R}\times\mathrm{J.D}\right)}{\mathrm{P}}$$

The PLLCLK is 12 MHz, R = 1, P = 1, J= 7, and D = 1680. The data sheet makes it look like they are mixing using "×" and "•" for multiplication. What is meant is that the numerator is 12 MHz × 7.1680 which yields a PLL frequency of 86.016 MHz. Fortunately, the data sheet gives the values for 12 MHz MCLK to generate 48 kHz and 44.1 kHz so we can do a sanity check. We are on our own when we want to use 96 kHz or 192 kHz sample rates, so knowing how to use the formula is important.

The signal logic for the DDS FM transmitter program is also ported intact from the *DirectX* PC program. It is really nice to be able to use the virtual serial port to do the same user interface that we had on the PC. We prompt for the desired carrier frequency and the deviation value. The modulator then is implemented by adding the input voltage to the frequency tuning word for each sample.

Next Issue

My plan for next issue is to experiment with implementing IF filters using finite impulse response filtering. We will read a center frequency and bandwidth value from the console, and then implement one of several demodulators.