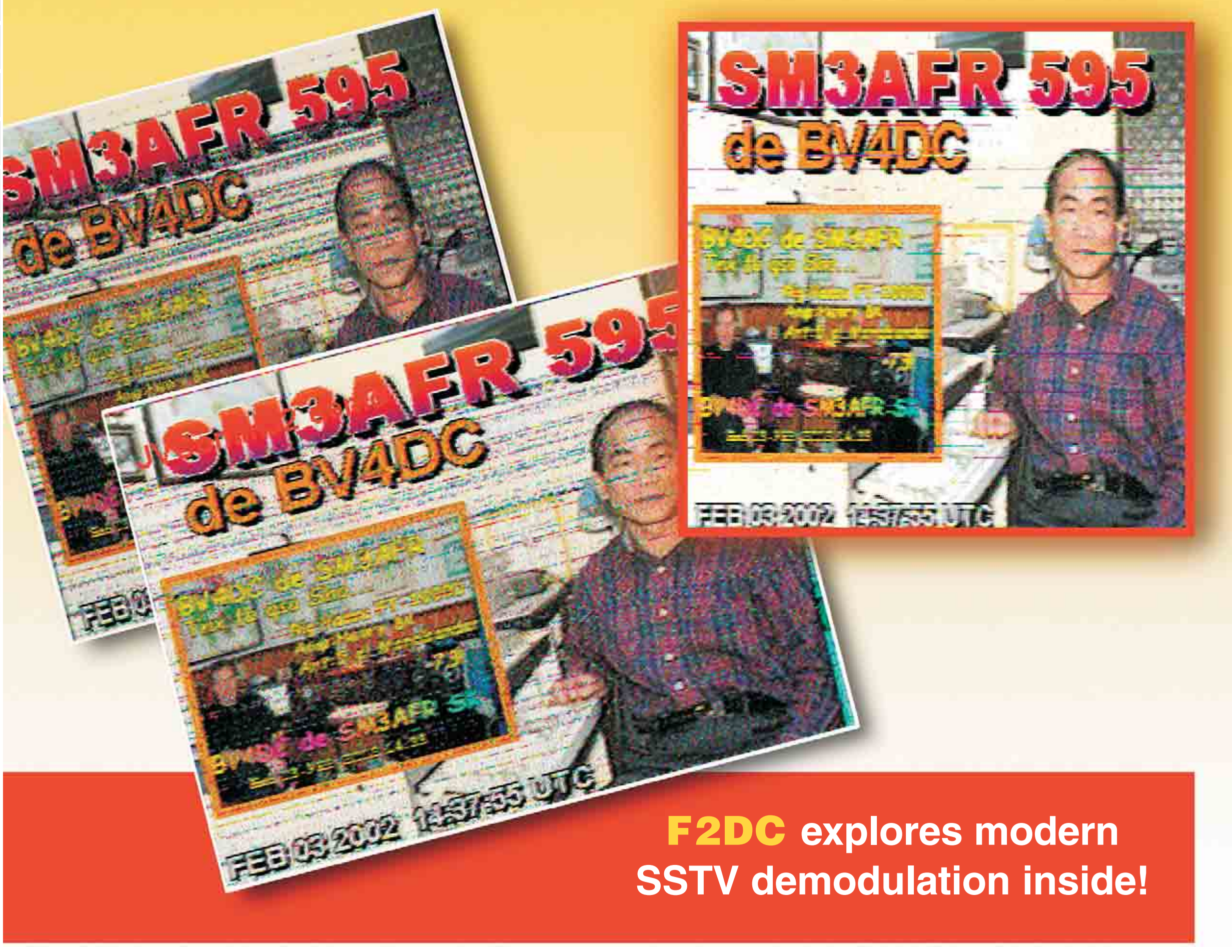
$5

QEX

INCLUDING: COMMUNICATIONS QUARTERLY

Forum for Communications Experimenters

May/June 2003
Issue No. 218

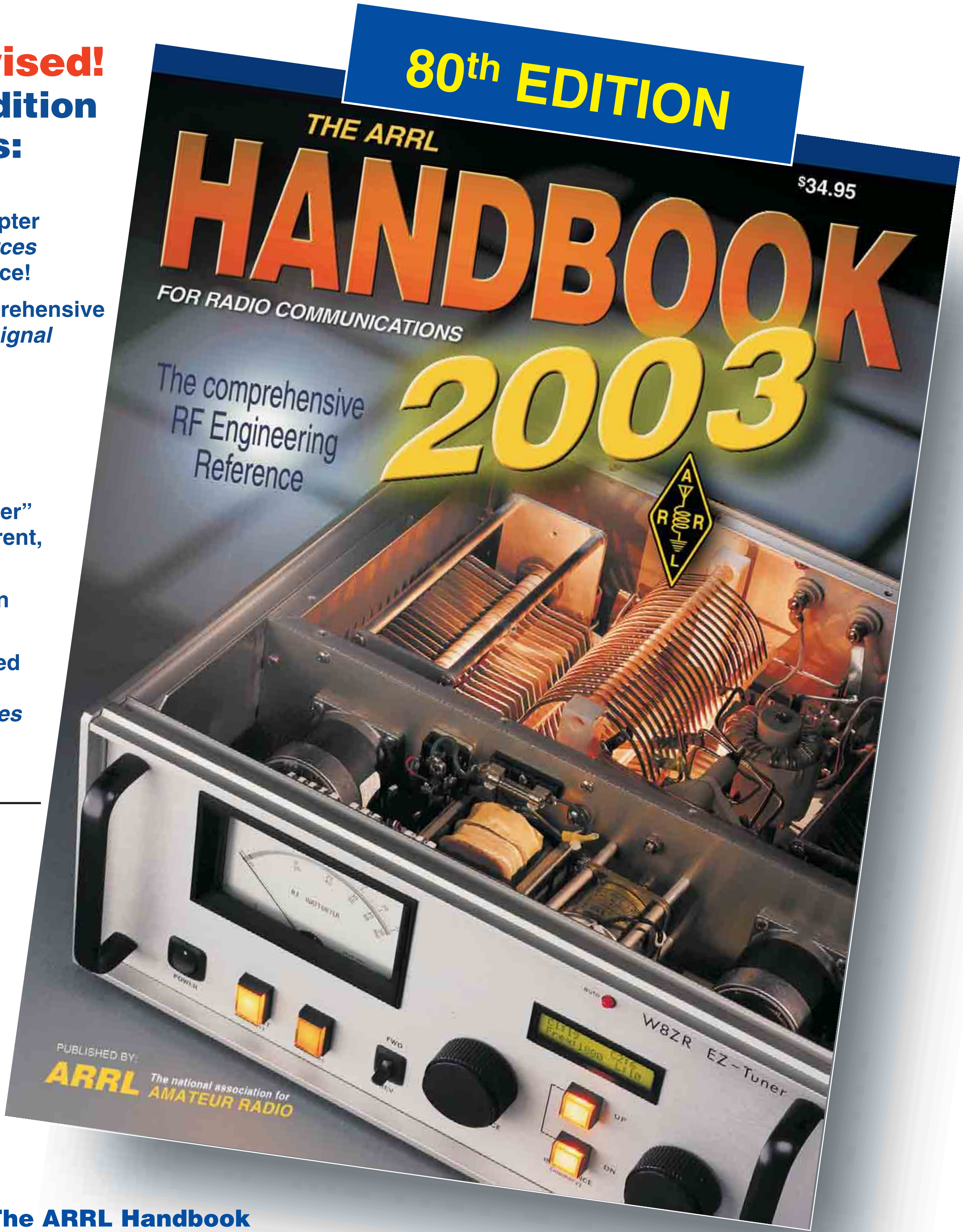


F2DC explores modern SSTV demodulation inside!

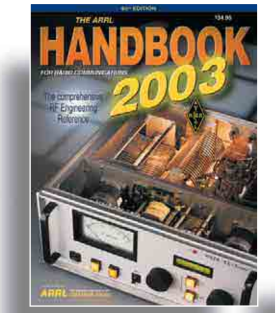
ARRL The national association for AMATEUR RADIO
225 Main Street
Newington, CT USA 06111-1494

Always revised!
The 2003 edition includes:

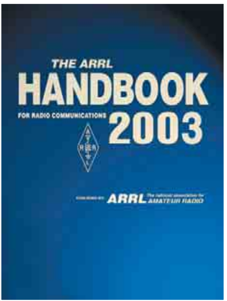
- An updated and comprehensive chapter on *Modulation Sources* including digital voice!
- A revised and comprehensive chapter on *Digital Signal Processing* (DSP) technology.
- A new high-power, automatic EZ-Tuner project by W8ZR.
- An "Ugly Transformer" project for high current, 120-V ac stations.
- A revised chapter on *Safety Practices*.
- A completely updated *Handbook* Address List in the *References* chapter.


80th EDITION
THE ARRL
HANDBOOK
FOR RADIO COMMUNICATIONS
$34.95
2003
The comprehensive RF Engineering Reference
PUBLISHED BY:
ARRL The national association for AMATEUR RADIO
W8ZR EZ-Tuner

THE STANDARD!



The ARRL Handbook for Radio Communications



Softcover—
ARRL Order No. 1921 .... **$34.95***
*plus shipping $7 US (UPS) /$9 International

Hardcover—
ARRL Order No. 1948 .... **$49.95***
*plus shipping $8 US (UPS) /$10 International

Available from ARRL publication dealers, everywhere!

Sales tax is required for orders shipped to CA, CT, VA, and Canada.
Prices subject to change without notice.

The ARRL Handbook CD for Radio Communications

Version 7.0—for *Windows* and Macintosh**
View, Search and Print from the entire 2003 edition book! CD-ROM.
ARRL Order No. 1956 .... **$39.95***

*plus shipping $5 US (UPS)/$7 International

**Some supplementary software utilities included—for *Windows* and DOS only.

Fully Searchable!

ARRL *The national association for AMATEUR RADIO*

225 Main Street, Newington, CT 06111-1494 tel: 860-594-0355 fax: 860-594-0303

In the US call our toll-free number **1-888-277-5289** 8 AM-8 PM Eastern time Mon.-Fri. **www.arrl.org/shop**

QEX

INCLUDING: COMMUNICATIONS QUARTERLY

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 218

Mark J. Wilson, K1RO
Publisher

Doug Smith, KF6DX
Editor

Robert Schetgen, KU7G
Managing Editor

Lori Weinberg, KB1EIB
Assistant Editor

Zack Lau, W1VT
Ray Mack, WD5IFS
Contributing Editors

Production Department

Steve Ford, WB8IMY
Publications Manager

Michelle Bloom, WB1ENT
Production Supervisor

Sue Fagan
Graphic Design Supervisor

David Pingree, N1NAS
Technical Illustrator

Joe Shea
Production Assistant

Advertising Information Contact:

Joe Bottiglieri, AA1GW, *Account Manager*
860-594-0329 direct
860-594-0200 ARRL
860-594-4285 fax

Circulation Department

Kathy Capodicasa, *Circulation Manager*
Cathy Stepina, *QEX Circulation*

Offices

225 Main St, Newington, CT 06111-1494 USA
Telephone: 860-594-0200
Telex: 650215-5052 MCI
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;

Elsewhere by Surface Mail (4-8 week delivery): ARRL member $31, nonmember $43;

Canada by Airmail: ARRL member $40, nonmember $52;

Elsewhere by Airmail: ARRL member $59, nonmember $71.

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.

QEX
INCLUDING: COMMUNICATIONS QUARTERLY
Forum for Communications Experimenters
May/June 2003
F2DC explores modern SSTV demodulation inside!
ARRL

About the Cover

The same SSTV frame is decoded by three methods. See the article on p 3.

Features

Columns

May/June 2003 *QEX* Advertising Index

American Radio Relay League: Cov II, 61, Cov III, Cov IV
Atomic Time, Inc.: 57
Buylegacy.com: 64
Down East Microwave Inc.: 35
Expanded Spectrum Systems: 35
Roy Lewallen, W7EL: 64
National RF: 51
Nemal Electronics International, Inc.: 51
Noble Publishing Corp: 43
Teri Software: 35
Tucson Amateur Packet Radio Corp: 64

THE AMERICAN RADIO RELAY LEAGUE

The American Radio Relay League, Inc, is a noncommercial association of radio amateurs, organized for the promotion of interests 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 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 two 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 at 225 Main Street, Newington, CT 06111 USA.

Telephone: 860-594-0200
Telex: 650215-5052 MCI
MCIMAIL (electronic mail system) ID: 215-5052
FAX: 860-594-0259 (24-hour direct line)

Officers

President: JIM D. HAYNIE, W5JBP
3226 Newcastle Dr, Dallas, TX 75220-1640
Executive Vice President: 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 on IBM or Mac format 3.5-inch diskette 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*. Further information for authors can be found on the Web at **www.arrl.org/qex/** or by e-mail to **qex@arrl.org**.

Any opinions expressed in *QEX* are those of the authors, not necessarily those of the Editor or the League. While we strive to ensure all material is technically correct, authors are expected to defend their own assertions. Products mentioned are included for your information only; no endorsement is implied. Readers are cautioned to verify the availability of products before sending money to vendors.

Empirical Outlook

A Busy Time for Amateur Radio

May and June bring several very important events for Amateur Radio: two are the Dayton Hamvention (May 16-18, **www.hamvention.org**) and the ARRL National Convention at Hamcom in Arlington, Texas (June 20-22, **www.hamcom.org**). Those gatherings will again give you opportunities to meet your Amateur Radio leadership in person, to have personal discussions with them and make your views known. You can also meet old friends, make new friends and catch the very latest in software and equipment. We encourage you to take advantage of those opportunities– be there!

The forum programs are vital parts of the events. Forums are where you can get up-to-the-moment information on operating techniques, regulatory issues and technical advancements from the experts. Material presented in them often cannot be found elsewhere. The ARRL Technology Task Force and its Working Groups will be represented at Dayton and Arlington, making their first combined presentations on software radio, digital voice and high-speed multimedia. Many other entertaining and informative speakers will be on hand. We hope to see you at one or both of those shows this year.

May 17 is ITU World Telecommunications Day. The World Radiocommunication Conference (WRC) 2003 is June 9-July 4 in Geneva. Perhaps of most interest to amateurs at the conference is agenda item 1.23, which involves several proposals regarding the 40-meter band. The proposal endorsed by the International Amateur Radio Union (IARU) would give hams a worldwide allocation at 7.00-7.30 MHz and international broadcasters would take 7.30-7.55 MHz. The change would be made in two steps, 100 kHz at a time, by 2010, thereby minimizing impact on broadcast and fixed services.

On the digital audio-broadcasting front, the Digital Radio Mondiale (DRM) consortium is making available some long-awaited software for demodulating international digital audio broadcasts (**www.drmrx.org**). For 60 Euros or about $65, you can own it. An open-source version is also available. You'll need a receiver with a 12-kHz IF output of 10-kHz bandwidth or greater and a reasonably fast PC with a decent sound card. The receiver can be the tough item unless you are prepared to modify one. At the DRM site, you will find a list of receivers that have been successfully modified and information about add-on down-converter modules for use with 10.7-MHz and 455-kHz IFs. I use an RX320, which already has the correct IF, with good results.

International HF digital audio broadcasting is just getting started. A few broadcasters are transmitting a regular schedule now from Europe and the Caribbean, but most are not aiming at the USA. Many of the broadcasts come in the wee hours for us. Nonetheless, some can be copied with good equipment and the number of DRM broadcasters will grow steadily over the coming months and years. Check it out, because the audio quality of these signals is nothing short of astonishing. Chip sets for embedded applications are being developed and the technology is available for licensing.

In This Issue

Roland (F2DC) and Lionel Cordesses discuss demodulation and signal processing of slow-scan TV images. They know their subject and treat it in quite some detail. Rod Brink, KQ6F, returns with a complete monoband transceiver design. It's the 17-meter Ragchewer, good for exactly what its name says—and much more.

Leif Åsbrink, SM5BSZ, continues his series on *Linrad*, his software-radio system for *Linux*. In this segment, Leif explains about how some of his software algorithms perform, how to install the software and get it working for you. Paul Wade, W1GHZ, tells us how he exploits his palmtop computer to make and log contacts. For portable operations, it seems like a good way to go! Read about how Paul does it and put the knowledge to your own use.

Robert Kavanagh, VE3OSZ, revives a method for building variable inductors—the soft ferrite core. He puts their B-H curves nicely to use in a remotely tuned receiving loop for LF and MF.

In RF, Zack Lau, W1VT, describes a 2-meter receiver useful as an IF for microwave work.—*73, Doug Smith, KF6DX;* **kf6dx@arrl.org**. □□

Some Thoughts on “Real-Time” SSTV Processing

How to improve our present SSTV programs.

By Lionel and Roland (F2DC) Cordesses

Picture transmission—SSTV and fax—has been a part of the Amateur Radio world for many decades. The receiving end changed dramatically with the availability of PC-based software at the beginning of the 1990s. However, the methods used to demodulate the SSTV signal, that is, to measure the frequency of the received tone carrying the luminance, are not so different from those used 10 years ago. Today, microprocessors are powerful enough to run efficient “real-time” algorithms that yesterday needed a digital signal processor (DSP). Modern available computing power allows better processing of raw SSTV signals and better picture quality.

In this paper, we present some unconventional approaches to process and use synchronization signals as well as to extract luminance information. These processing methods significantly improve picture quality when receiving conditions are poor—in the presence of noise, QRM and so forth. They also perform an on-line, accurate estimation of the sound-card sampling frequency, circumventing the calibration step needed in many—if not all—varieties of SSTV software. A dedicated program using the described algorithms has been designed and tested not only during simulations but also on real on-the-air SSTV signals.

An Overview of SSTV

While this paper focuses on the processing of the Martin M1 signal, specifications of which were kindly provided to us by its author G3OQD,[1] it is clear that ideas and algorithms presented here can be applied to other SSTV formats. We will first present some methods that are used to demodulate SSTV signals and then recall some technical specifications of the M1 mode.

SSTV Demodulation

The purpose of any color SSTV decoder is to recover the red, green and blue original information from the frequency-modulated signal and to accurately extract the video line with respect to the horizontal synchronization signal. This has long been done using analog approaches.[2]

Before the all-digital era, a hybrid approach was used. The demodulator was still relying upon an analog circuit based on filters, adders and so on. Once demodulated, this analog signal was digitized. Further processing, such as image processing, was then carried out on a computer. This was done back in 1970.[3] Later, digital processing of the demodulated signal became the basis of some computer-based SSTV solutions. See “Viewport VGA” and our own solution based upon the same hardware.[4, 5]

The all-digital approach first converts the analog signal from the receiver into a digital signal, using an analog-to-digital converter (ADC). Then, all the DSP demodulation and synchronization detection is performed on a computer. The two-level ADC, often a simple op amp, became one of the bases of much software—*JVFax*, for instance.[6]

Then came sound cards. Thanks to low-cost hardware, the sampling of

[1] Notes appear on page 19.

26 rue du Montant
F63540 Romagnat
France
roland.cordesses@free.fr

2 allee Dauphine
F78140 Velizy
France
cordesses@renagri.com

analog signals was possible. Most of the actual sound cards can reach a 44.1-kHz sampling frequency with 16 bits per sample. A new generation of software makes use of the sound card: *JVComm32* is one of them.

Our approach clearly belongs to the last one—the all-digital one. After 10 years of experiments with SSTV demodulation for one author and more than thirty for the other, we really think there is room for improvement. Perhaps a lot of room, we think.

When receiving conditions are poor and the signal-to-noise ratio (S/N) is low, much of the available software performs poorly. Sometimes it is not able to receive a faint picture you can hear in the background. Other times, it does receive something. Unfortunately, most software is unable to synchronize properly under these conditions. We decided to focus on these situations and attempt an answer to the underlying technical questions. So, if you are interested in SSTV, or just curious to see what DSP can bring to such an old standard, go on reading!

The Martin M1 SSTV Signal

Let's briefly recall the specifications of such a SSTV signal. The original picture is in color, described by its red, green and blue spectral components. The size of the picture is 256× 256 pixels. Each spectral component is transmitted in turn; that is, the 256 green pixels signal first, the blue and then the red. As in McDonald's original system (see Note 2), the modulation is a frequency modulation described by Eq 1.

$$f_l = f_{\text{black}} + \frac{(f_{\text{white}} - f_{\text{black}})lum}{max_{\text{lum}}} \qquad \text{(Eq 1)}$$

where: f_{black} is the frequency of a black pixel (1500 Hz), f_{white} is the frequency of a white pixel (2300 Hz), and max_{lum} is the maximum value of the luminance signal. A usual value is 255 for pictures coded with eight bits per component (and thus 8 × 3 = 24 bits for red, green and blue). *lum* is the value of the luminance and f_l [read "*f*" subscript lower-case "L"—*Ed.*] is the corresponding modulating frequency.

One line of the original picture is transmitted according to the following timing (see Fig 1). The horizontal line-synchronization signal (f_{sync} = 1200 Hz) lasts t_s = 4862 µs. During t_1 = 572 µs, a 1500-Hz tone is transmitted. The green component of the line is then transmitted. Each pixel lasts 572 µs, thus the green signal lasts t_G = 256 × 572 = 146,432 µs.

During t_1 = 572 µs, a 1500-Hz tone is transmitted. The blue component of the line is then transmitted. Each pixel lasts 572 µs, thus the blue signal lasts t_G = 256 × 572 = 146,432 µs.

During t_1 = 572 µs, a 1500-Hz tone is transmitted. Lastly, the red component of the line is transmitted. Each pixel lasts 572 µs, thus the red signal lasts t_G = 256 × 572 = 146,432 µs. During t_1 =572 µs, a 1500-Hz tone is transmitted. This is the end of the line.

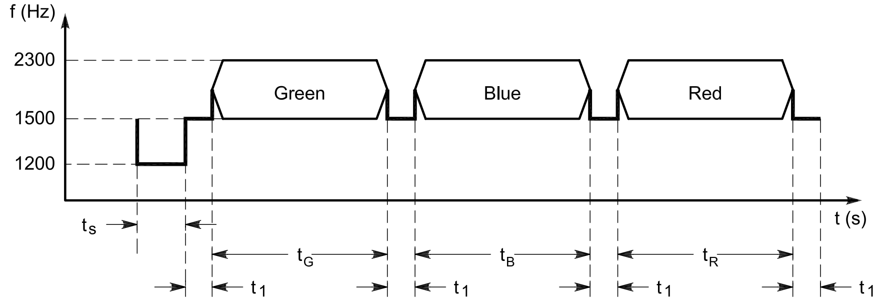


Fig 1—Martin M1 line timing.

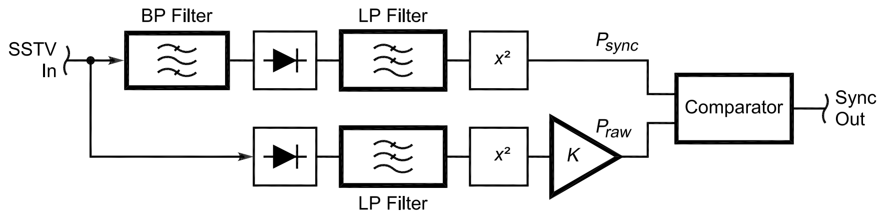


Fig 2—Synchronization detector overview.

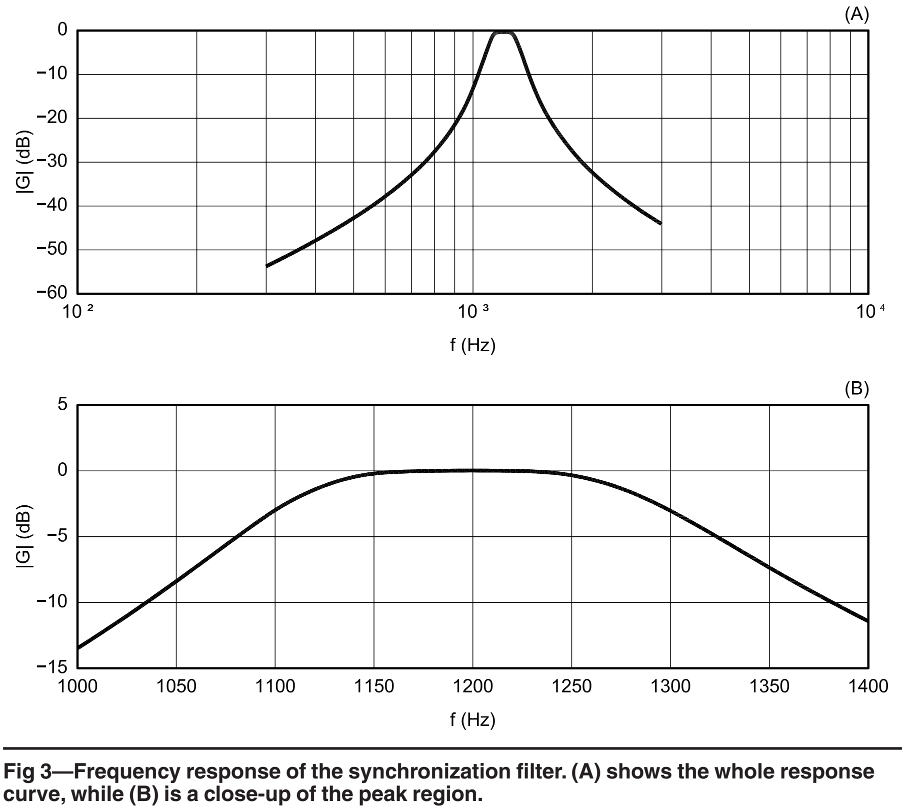


Fig 3—Frequency response of the synchronization filter. (A) shows the whole response curve, while (B) is a close-up of the peak region.

Horizontal Synchronization Method

This section is devoted to the horizontal synchronization algorithm we have developed. First, we consider the need for such a software module. We then describe the synchronization detector. It is based upon a band-pass filter followed by a linear Hough transform. The linear Hough transform is explained in a step-by-step approach.

The Need for Horizontal Synchronization

The reader may wonder why we need such a synchronization detector. Thanks to digital headers transmitted before the image signal (referred to as VIS code), software is able to detect the beginning of a new image. It also extracts from the digital header the mode used by the transmitter. Then, as the receiving software relies upon a calibrated time base, it *asynchronously* decodes the incoming signal, and it does not use the horizontal-synchronization signal any more.

Remember that we have decided to focus on realistic receiving conditions. The above mentioned method fails, for instance:

- When fading occurs during the VIS,
- When QRM prevents the software from detecting the start of image signal or
- When the operator misses the beginning of the transmission.

We will see that the forgotten horizontal-synchronization signal can drastically improve SSTV demodulation.

The Synchronization Filter: The analog SSTV signal is first digitized—thanks to the computer sound card—at $f_s = 44{,}100$ Hz. The resulting data are then processed by the synchronization detector presented in Fig 2. This incoming SSTV signal goes through a digital recursive band-pass filter (also known as an infinite-impulse response filter or IIR).[7] We have chosen a four-pole Butterworth filter for its burst time response.[8] The center frequency is $f_{\text{sync}} = 1200$ Hz and the bandwidth is 200 Hz. The coefficients of the IIR filter are given in Table 1. The output $y[n]$ of the IIR filter is computed using Eq 2.

$$\begin{aligned} a_0 y[n] &= b_0 x[n] + b_1 x[n-1] \\ &\quad + b_2 x[n-2] + b_3 x[n-3] \\ &\quad + b_4 x[n-4] - a_1 y[n-1] \\ &\quad - a_2 y[n-2] - a_3 y[n-3] \\ &\quad - a_4 y[n-4] \end{aligned} \qquad \text{(Eq 2)}$$

The frequency response of this filter is plotted in Fig 3. The upper curve displays the frequency response from 300 Hz to 3000 Hz, and the lower curve

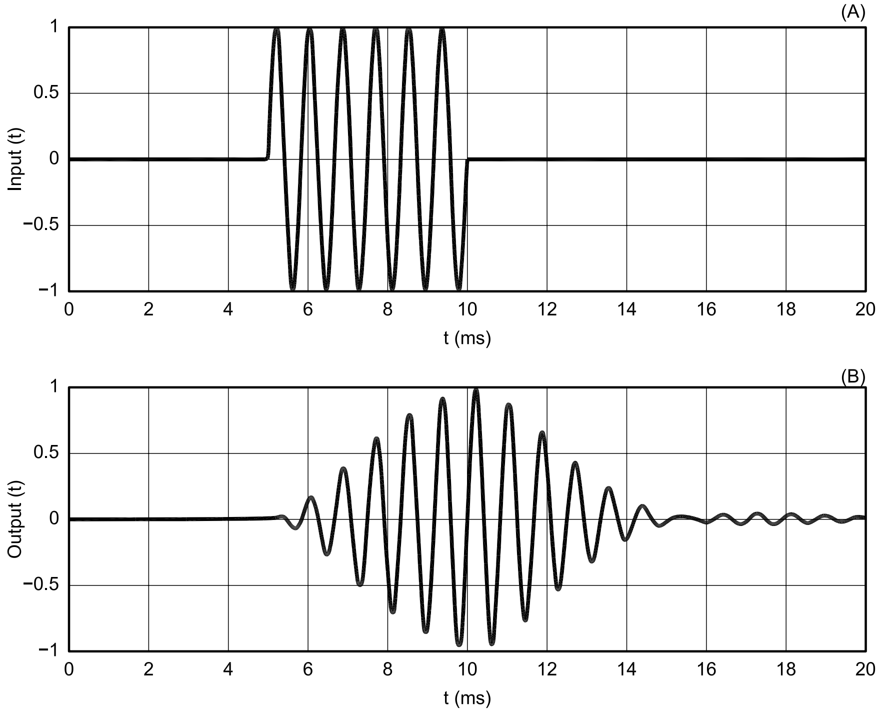


Fig 4—Burst response of the synchronization filter. (A) shows the 1200-Hz burst; (B) shows the burst as shaped by the filter response.

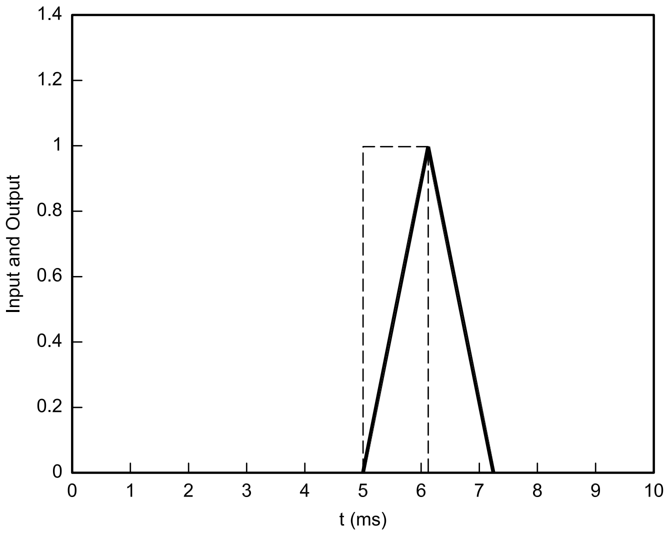


Fig 5—Time response (solid) of the FIR filter to a rectangular window dotted.

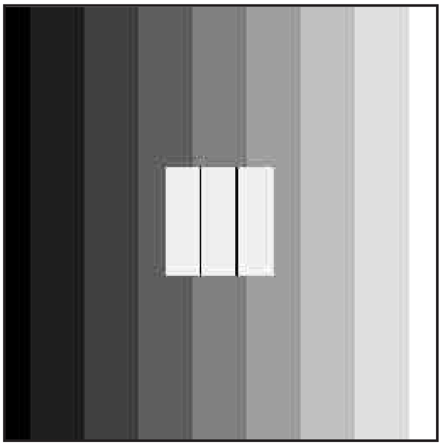


Fig 6—The reference picture.

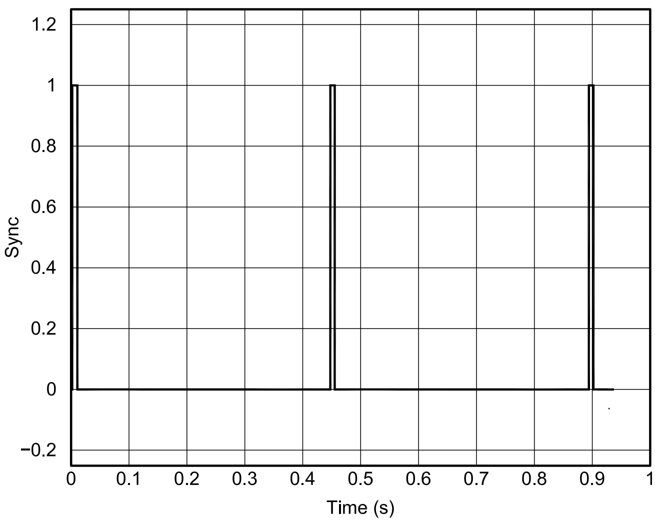


Fig 7—Output of the synchronization detector (M1 signal without noise).

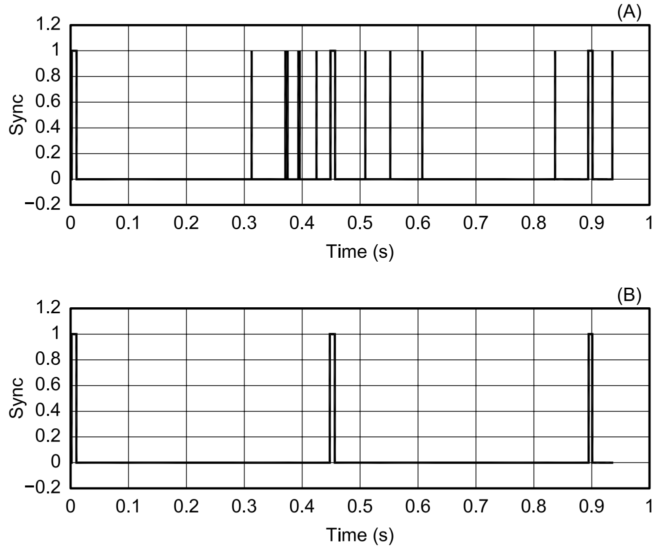


Fig 8—Output of the synchronization detector (M1 signal with noise, A, versus without noise, B).

is a closer look around the center frequency.

We are also highly interested in the behavior of this filter to a 1200-Hz sine-wave burst. We have therefore simulated a synchronization-like signal made of: 5 ms without signal, 5 ms with a 1200 Hz, unity-amplitude sine wave, and 10 ms without signal.

The choice of a 5-ms, 1200-Hz sine wave signal is realistic with respect to both the original standard (black and white) and the M1 (color) SSTV modes.[9] Both the input signal (Fig 4, upper curve) and the output of the IIR filter (lower curve) are plotted.

The 200-Hz bandwidth chosen during the design is the lower bound one can use. One can see in Fig 4 that the output signal reaches unity amplitude just before it starts decaying. This bandwidth is a trade-off between noise rejection and response time. It proved efficient during the many experiments carried out on-air with real SSTV M1 signals.

This filtered signal is detected thanks to an absolute-value function (the diode in Fig 2) along with the original signal. The remaining blocks are a poor-man's power-spectrum estimator. The 50-tap digital low-pass filter (a finite-impulse-response or FIR filter) is a moving-average type. It acts as a low-pass filter with a zero in its frequency response at f_s / 50 = 882 Hz.

Here's another way of explaining the behavior of this low-pass filter: It outputs the correlation of the detected signal with a rectangular window. This rectangular window, lasting 50 / f_s = 1.1 ms, is related to the desired time resolution of the synchronization detector. The correlation gives an indication of how much two signals look alike, along with the time of this likelihood measurement. A simulation result is displayed in Fig 5. The dashed line is the input signal, made of 5 ms without signal, 1.1 ms with a unity-amplitude signal and 3.9 ms without signal. It is a rectangular window of 50 samples. The bold continuous line in Fig 5 is the output of the filter. The maximum value (1 on Fig 5) is reached when the likelihood is at its maximum.

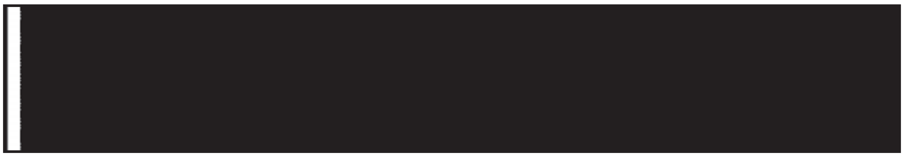


Fig 9—The synchronization signal—a 2D approach (no noise).

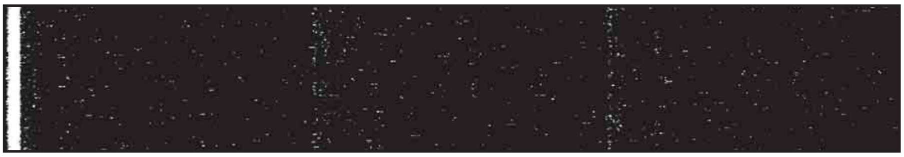


Fig 10—The synchronization signal—a 2D approach (with noise).

Now, look back to Fig 2: This FIR filter is applied to both raw and band-pass filtered detected signals. The detected value (the output of the diode in the figure) can be seen as a voltage. So squaring this value (the x^2 block in Fig 2) leads to a power, noted as P_{sync}. The raw power, P_{raw}, is normalized thanks to the gain K with respect to its assumed bandwidth (3000 Hz can be used as an upper bound for the average communication receiver). Now, both power channels can be compared against one another. The meaning of this comparison can be related to the following question: "Is there more power in the 1100 to 1300-Hz range than in the rest of the band?"

A simple threshold is set to decide whether the SSTV input signal is a synchronization signal or another type—video, for instance, or some QRM. During all the experiments, this threshold has been set to two: The incoming signal is said to be a synchronization signal when $P_{sync} > 2 \times P_{raw}$. Now, let us see how our synchronization detector performs.

Simulation Results: Why do we mention simulation results? A real, noisy, faded SSTV signal coming from the air would be nice; but it would be very difficult to analyze. Unless we have access to the original, clean signal, it is difficult to qualify the demodulator. Anyway, we will show some results with simulated data.

Simulation Signals: The original SSTV signal (the reference) has been created with a stand-alone program we have developed for this purpose. It reads a bitmap (see Fig 6) 24-bit color file and creates an M1-compliant monaural sound file (.wav). It also has the nice feature of adding noise to the pure M1 signal.

The noisy signal includes Gaussian noise. Its standard deviation from the mean is 1, and its bias (or mean value) is 0. This noise is then filtered by a low-pass filter, an 8th-order Butterworth IIR with a cutoff frequency of 2500 Hz. The value of 2500 Hz is a realistic one for SSTV signals.

This sound file is either used by our SSTV software or transferred to a CDROM for test purposes. This brand new audio CD, playing on a CD player, becomes the source of the SSTV signal that is digitized by the sound card. This solution has been extensively used when comparing the performance of our SSTV software against other programs.

Fig 7 plots the output of the synchronization detector for a clean SSTV M1 signal (without Gaussian noise). The horizontal-synchronization pulses are perfectly estimated. The period between two pulses, measured on Fig 7, is about 0.446 s. The theoretical value is 4862 + 572 + (572) (3) + (256) (572) (3) = 446,446 µs.

We have already mentioned that we were interested in the behavior of this synchronization detector when the S/N is low. We have therefore created a noisy M1 signal (S/N is almost 0 dB). The output of the same software is displayed on the upper curve of Fig 8 for this noisy signal. The reference signal is plotted on the lower curve to make comparison easier. At first sight, it is very difficult to accurately detect horizontal line synchronization out of this 1-D signal.

This first conclusion has led us to think about another representation of the *same* signal. The 1-D signal is indeed the output of the synchronization detector, so the 1-D representation may first come to mind. Yet this 1-D signal actually results from a 2-D signal; namely, the original picture. Let us plot the same output signal from the synchronization detector as a picture, thus as a 2-D signal. We use the following convention:

- A 0 in the 1-D signal (no synchronization signal) is displayed as a black pixel.
- A 1 in the 1-D signal (a synchronization signal) is displayed as a white pixel.
- The width of the resulting picture is X_{M1} = 1561 pixels.
- The height of the resulting picture is limited by the total amount of available synchronization samples.

The value X_{M1} = 1561 comes from the length of one M1 line, 446,446 µs, and the duration of one sample of 286 µs: 1561 = 446,446 / 286. The latter value of 286 µs has been chosen to get an integral number of samples for the synchronization signal (4862 / 286 = 17 samples) and for all other parts of the M1 line.

The resulting picture is displayed in Fig 9 for the M1 signal without noise (the same one used for Fig 7). A human being easily locates the synchronization signal on the left.

The case of the noisy M1 signal leads to the picture displayed in Fig 10. Even on this picture, coming from a very noisy M1 SSTV signal, the reader easily detects the synchronization signal.

The question is, "How can an algorithm accurately detect this vertical line?" Assuming it is possible, we would have built a robust synchronization detector based upon the whole synchronization signal.

From a 2-D Picture to a Synchronization Detector

The very first approach one can test is a least-mean-square (LMS) approach. The problem can be stated this way: *Knowing the 2-D coordinates of some points, the algorithm has to estimate the abscissa of the vertical line while minimizing some criteria.*

It is a simple curve-fitting approach[10] with a line described by Eq 3:

$$x = b \quad \text{(Eq 3)}$$

The parameter b can be estimated as the mean value of all the abscissas of the white pixels. It works perfectly when the M1 signal is clean (no noise),

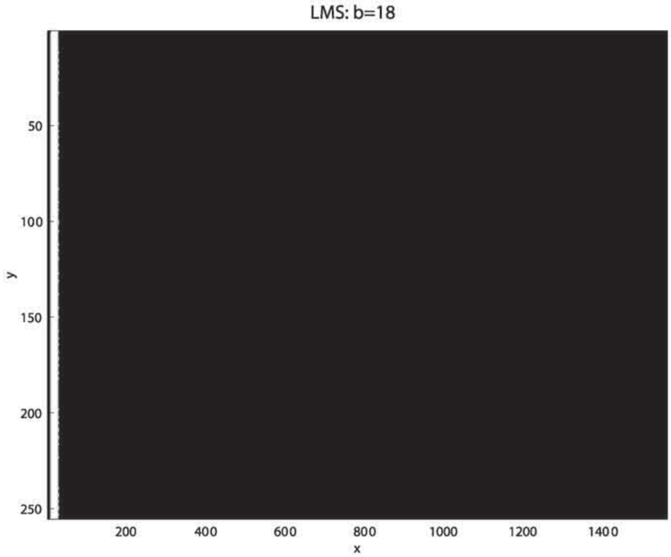


Fig 11—Synchronization signal estimated thanks to LMS (no noise, 250 lines).

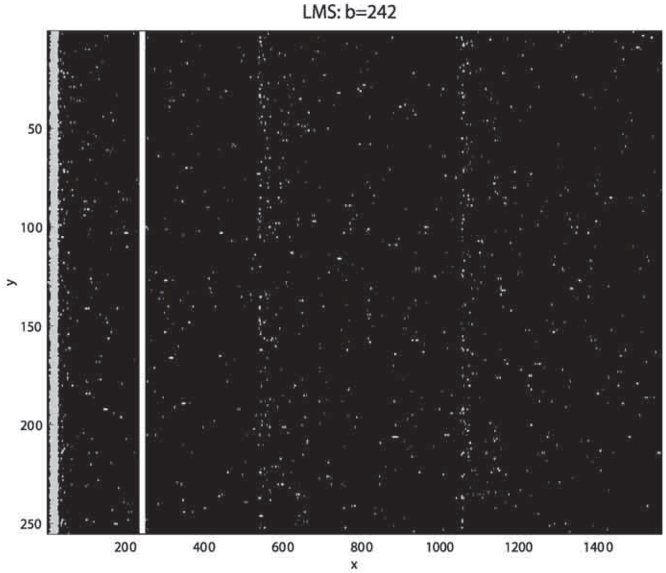


Fig 12—Synchronization signal estimated thanks to LMS (with noise, 250

and thus when the synchronization signal is perfectly recovered. The result from a LMS approach is displayed in Fig 11. The grey pixels are the output of the synchronization detector, and the white vertical line is the estimated one, thanks to the LMS. The estimated *b* is 18, which is correct.

This very same method totally fails when the signal is noisy (see Fig 12). The reader may wonder why such a popular method fails. The answer is simple: The parameter to be estimated, a constant, is corrupted by a noise with a nonzero mean. Here, the noise is produced by the synchronization detector. This is why we have used another approach that is robust against this kind of noise and is presented in the next section.

Using the Linear Hough Transform

The method we have used is robust against the kind of noise generated by the synchronization detector. It relies upon the linear Hough transform. We will first describe it using a step-by-step

```
// Input:
// pixel: an array of size (xMax+1,yMax+1)
// Output:
// bLine: the b parameter of the synchronization vertical line.
// maximum: the number of occurence of this b value.
//
// first step
for y=0 to yMax
   for x=0 to xMax
      if pixel(x,y) is white
         // estimate b parameter
         b=x
         // use the accumulator and take this new value
         // into account
         accumulator [b] =accumulator [b]+1
      end if
   end for
end for
// second step
maximum=0
for b=0 to xMax
   if (accumulator[b]>maximum)
     maximum=accumulator[b]
     bLine=b
   end if
end for
```

Fig 13—The Hough algorithm for a vertical line.

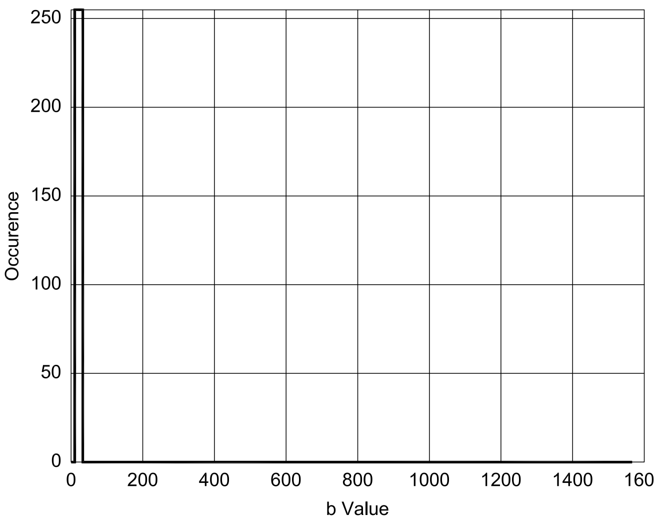


Fig 14—The accumulator corresponding to Fig 9 (M1 signal without noise) showing occurrences of *b* values (b_{line} = 9).

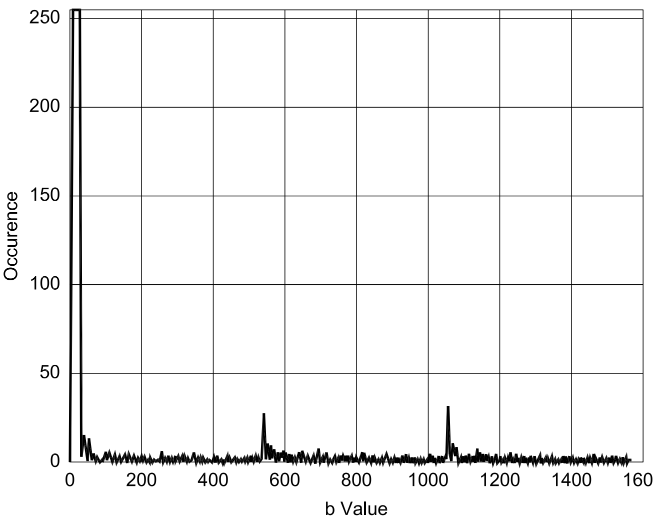


Fig 15—The accumulator corresponding to Fig 10 (M1 signal with noise) showing occurrences of *b* values (b_{line} = 11).

approach in a very simple situation. We will then introduce the linear Hough transform as presented in Reference 11. We will also illustrate it using another example in a step-by-step approach.

Estimating the b Parameter of a Vertical Line

This simple case deals with the same vertical line described by Eq 3. Let's have a closer look at either Fig 9 or 10. You can easily *see* where the vertical line is (on the left of both pictures). We could also write that *most of the pixels on the left belong to the* same *vertical line, whereas the remaining pixels do not belong to the same vertical line*. This idea can be translated into a two-step algorithm.

First step: For each white pixel, compute the line parameter b of the vertical line to which it belongs.

Second step: Among all the resulting b parameters, find the one with the highest occurrence.

The algorithm in Fig 13 performs these two steps. We have introduced an accumulator, which is an array. It is used to compute and store the number of occurrences of the b parameters.

This algorithm has been used on the very same data from the synchronization detector. Fig 14 plots the accumulator corresponding to the clean M1 signal (from Fig 9). Even in the presence of noise, the algorithm is still able to recover the synchronization. Fig 15 plots the accumulator corresponding to the noisy M1 signal of Fig 10. The b parameter is correctly estimated with this method: The error is 2/1561.

The Linear Hough Transform

This section could have been entitled "Estimating the Parameters of a Straight Line." We have seen in the previous section how to estimate the b parameter of a vertical line. It was a simple case of the linear Hough transform. We are now going to introduce the general linear Hough transform; we will illustrate its use through a step-by-step computation.

No More Slanted Pictures!

We have already seen that the results were convincing with this simple estimator, even in the presence of noise. The reader may wonder why we proceed further.

Have you ever seen a beautiful SSTV picture spoiled because it is slanted? How many times have you thrown away a rare DX picture just because of a badly calibrated time base?

A badly calibrated time base—at either the transmitter or receiver side—results in a slanted picture (see Fig 16, for instance). Much modern SSTV software allows the user to finely tune its time base to get perfect pictures. Actually, the received pictures are perfect as long as the transmitter time base is correctly calibrated.

Could we imagine an automatic system that relies only on the previously described "synch" detector? Such a system would receive a SSTV signal and output a vertical picture, even if the time base were not calibrated. We can design such a system. From a practical point of view, one can improve our SSTV synch detector—and thus the SSTV software—using the linear Hough transform.[11]

The Hough transform is named after its inventor (see the US patent).[12] It is widely used in the field of image processing.[13] We will see how to use for SSTV synch detection.

The slanted picture of Fig 16 is linked with the 2-D slanted synch signal (see Fig 17). The linear Hough transform allows estimation of the parameters describing a straight line. Eq 4, used to describe this line, is not the usual $y = ax + b$, but:

$$= -x \sin q + y \cos q \qquad \text{(Eq 4)}$$

Eq 4 is better for our application, as it can describe any straight line: vertical, horizontal or skew. The usual equation $y = ax + b$ fails to describe a vertical line (as long as a is finite). Moreover, Eq 4 closely relates the slope of the line and the parameter q, which is useful for the Hough transform.

The linear Hough transform relies upon a 2-D accumulator or array. The two dimensions of the accumulator are linked with the two parameters (d, q) of the line. The algorithm, also a two-step, is described in Fig 18.

A Step-by-Step Computation

An example illustrates the algorithm of Fig 18. Consider a very simple picture. Its size is 5×5. The round dots are the pixels of interest (see Fig 19).

During this computation, we are only looking for horizontal, vertical and diagonal lines. Thus, the corresponding values of q are 0°, 45°, 90° and 135°.

Step One

Let us start with the pixel located at coordinates (–1, –2). According to the algorithm, for each possible value of q we have to compute the corresponding d using Eq 4. The results are tabulated

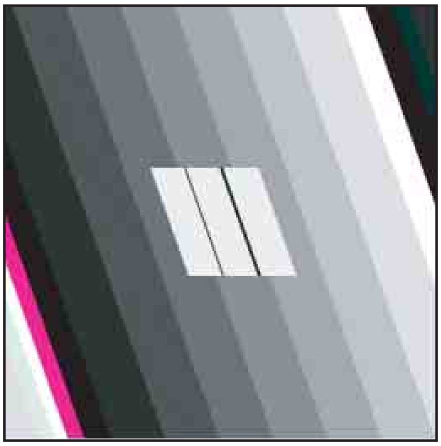


Fig 16—A slanted SSTV picture.

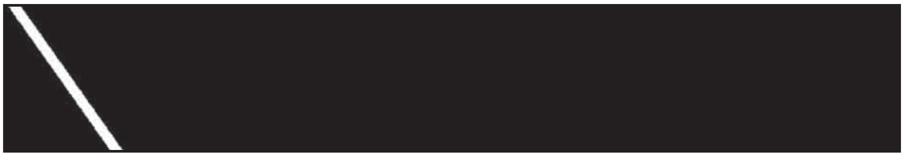


Fig 17—A slanted synchronization signal.

```
// first step
for each pixel (x,y) of interest
          for each possible q
             computes d = -x sin(q) + y cos(q)
             accumulate the set (d, q)
          end for
end for
// second step
find in the accumulator the set (d, q)
with the highest occurrence
```

Fig 18—The Hough algorithm for a straight line.

in Fig 20 (step 1). We must point out that *d,* computed using Eq 4, is a real number. It is represented by a floating-point number (see the third column). As this number is used as an index of an array (the accumulator), this floating-point number is rounded to the nearest integer (see the fourth column). The slight difference between the exact value of *d* and the value stored in the accumulator and used by the algorithm will cause an error in the estimated parameters. This error will be obvious when looking at the resulting estimated line. Nevertheless, it is not an important error, and it has no impact on our application.

The meaning of the first line of Fig 20 is as follows: *the pixel (–1, –2) belongs to the line whose equation is, according to Eq 4, –2 = –x* sin (0) + *y* cos (0). *A*lternatively, one can write that *the pixel (–1, –2) belongs to the line whose parameters (d, q) are (–2, 0)*. Now we could add one to the accumulator value located at (–2, 0). The accumulator would now reflect that there is one and only one line whose parameters are (–2, 0).

The resulting *d* values, along with their corresponding *q* values, are used to fill the accumulator. At the beginning, the accumulator is cleared—that is, filled with zeros. Using Fig 20, it is easy to build the accumulator at step 1:

q = 0° leads to $d = –2$; ACC[d, q] = ACC[d, q] + 1

q = 45° leads to $d = –1$; ACC[d, q] = ACC[d, q] + 1

q = 90° leads to d = 1; ACC[d, q] = ACC[d, q] + 1

q = 135° leads to d = 2; ACC[d, q] = ACC[d, q] + 1

The bold numbers in the accumulator represent the values different from the previous step. The picture on the left stands for the visual counterpart of the calculations. All the possible lines, corresponding to the tabulated values of *q*, are drawn on this figure.

Step Two

The same method is applied to the pixel (0, –1). The computed *d* for all the *q* values are tabulated in Fig 20.

Steps Three to Eight

The same method is applied to remaining pixels. The computed *d* for all the *q* values are tabulated in Figs 20-22.

The final values in the accumulator at (*d, q*) are the number of points belonging to the line (*d, q*). In our example, the highest value, 4 (see Fig 22) belongs to the line described by the set ($d = –1$; q = 45°). The line described by these parameters is plotted as a thick line in Fig 23, whereas the line calculated by LMS is the dashed line. Although not perfect (because of a rounding effect), the Hough estimation is better than the LMS.

The Case of the Slanted Picture

How to Estimate the Sampling Frequency of Your Sound Card

Unfortunately, this frequency is not as accurate as one might expect: the 44,100 Hz may be biased. A 40-Hz bias is common. This problem is well known in the SSTV community.

The following equations are based upon three hypotheses:

1. At time t = 0, the software is demodulating the very first video sample, located on the upper left corner (0 on Fig 24);
2. The M1 signal is perfect—that is, compliant with the timings defined by G3OQD;
3. The assumed sampling frequency, noted as f_S, is used to demodulate the M1 SSTV picture. This picture is available for the sampling frequency estimation (see Fig 24 for a model of such a slanted picture).

The coordinates of video sample B are (x_B, y_B); those of A are (x_A, y_A. Notice that one has $y_A = y_B$.

One video sample lasts t_1, that is 572 µs. During one second, the sound card digitizes f_S samples. Thus, one M1 video sample, lasting t_1, requires N_{M1} sound card samples, with N_{M1} defined by Eq 5:

(Eq 5)

Please note that N_{M1} is *not* an integer. The video sample B (see Fig 24) is received at time t_B. The total number of sound card samples corresponding to the video sample B is: $i_B \times N_{M1}$ with i_B the index of the video sample B, defined by $i_B = x_B + y_B \times X_{M1}$. Where X_{M1} is the number of video samples per M1 line. The video sample B is received at time t_B defined by Eq 6.

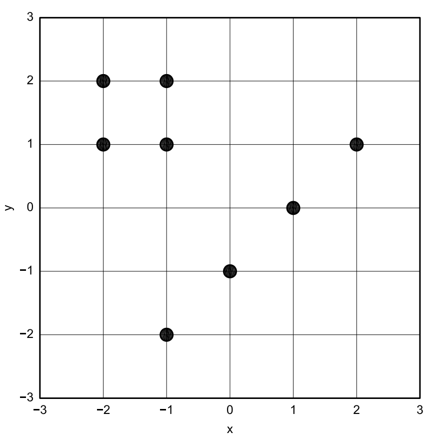


Fig 19—A simple 5×5 picture.

(Eq 6)

Where $f_{unknown}$ is the unknown frequency of the sound card (if B ≠ A, then $f_{unknown} \neq f_S$). From Eqs 5 and 6, we have:

$$t_B = i_B \times t_1 \times f_S \frac{1}{f_{unknown}} \quad \text{(Eq 7)}$$

As the received M1 SSTV signal is based upon the exact timings, we can infer that the video sample B should be located in A (see Fig 24).

The video sample A, which index $i_A = x_A + y_A \times X_{M1}$, occurs at time t_A:

$$t_A = i_A \times N_{M1} \frac{1}{f_S} \quad \text{(Eq 8)}$$
$$= i_A \times t_{M1}$$

As we know that the video sample B should be located in A, we can write Eq 9:

$$t_A = t_B \quad \text{(Eq 9)}$$

Substituting Eqs 6 and 8 into Eq 9 leads to:

(Eq 10)

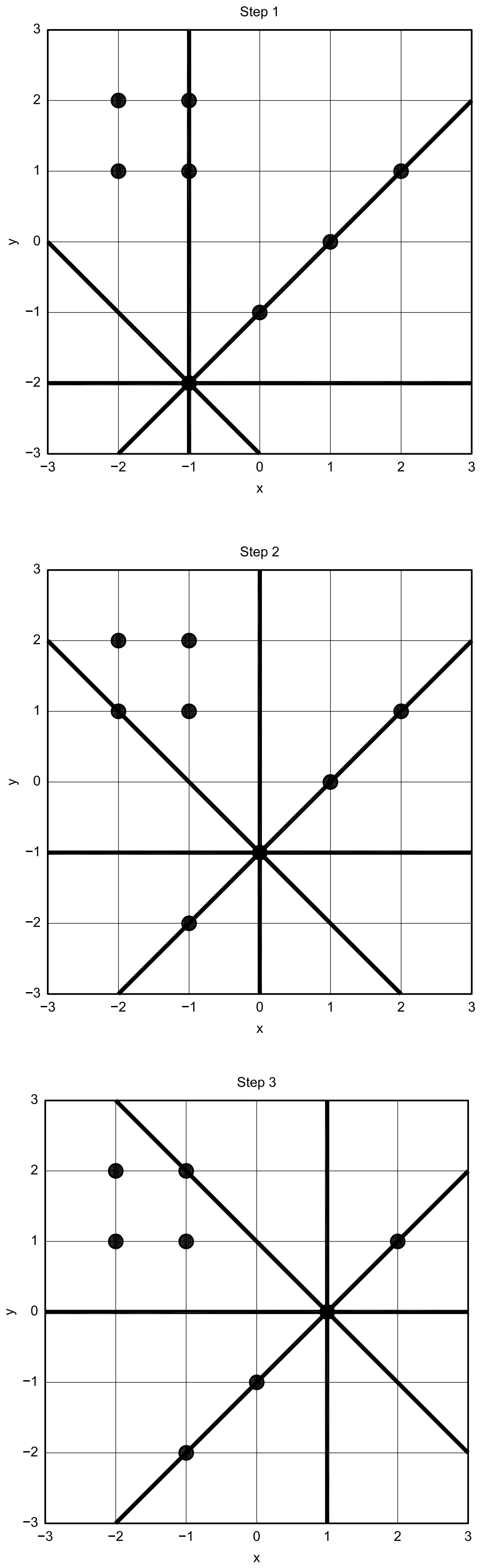
Finally, we can compute the unknown frequency $f_{unknown}$with Eq 11:

$$f_{unknown} = \frac{i_B}{i_A} f_S \quad \text{(Eq 11)}$$

Such a simple equation easily allows us to accurately estimate the sampling frequency of the sound card. In fact, the accuracy of our approach is only limited by the quality of the slope estimation. That explains why we needed a very-high-performance synchronization detector.

Experiments carried out with real noisy SSTV signals allowed us to estimate the sampling frequency of our sound card. The overall quality of the estimator is easily checked: The picture demodulated with the estimated $f_{unknown}$ is *perfectly* vertical.

Of course, a similar approach could be used when the sound-card sampling frequency is known, but the picture is nevertheless slanted. Such a situation occurs when demodulating a signal replayed on a CD player, whose frequency is not as accurate as one might think. The slope of the picture leads to an *ideal* value of f_S. When demodulating the same signal with this newly computed f_S, the formerly slanted picture now appears perfectly vertical. The very-high-performance



Step 1: Computing Parameter *d* for Pixel (–1, –2)

q	–x sin *q* + y cos *q*	*d*	*round(d)*
0°	1 × 0.00 –2 × 1.00	–2.00	–2
45°	1 × 0.85 –2 × 0.53	–0.71	–1
90°	1 × 0.89 –2 × –0.45	1.00	1
135°	1 × 0.09 –2 × –1.00	2.12	2

The Accumulator After Step 1

ACC	*0°*	*45°*	*90°*	*135°*
d = –2	***1***	0	0	0
d = –1	0	***1***	0	0
d = 0	0	0	0	0
d = 1	0	0	***1***	0
d = 2	0	0	0	***1***
d = 3	0	0	0	0

Step 2: Computing Parameter *d* for Pixel (0, –1)

q	–x sin *q* + y cos *q*	*d*	*round(d)*
0°	0 × 0.00 –1 × 1.00	–1.00	–1
45°	0 × 0.85 –1 × 0.53	–0.71	–1
90°	0 × 0.89 –1 × –0.45	–0.00	0
135°	0 × 0.09 –1 × –1.00	0.71	1

The Accumulator After Step 2

ACC	*0°*	*45°*	*90°*	*135°*
d = –2	1	0	0	0
d = –1	***1***	***2***	0	0
d = 0	0	0	***1***	0
d = 1	0	0	1	***1***
d = 2	0	0	0	1
d = 3	0	0	0	0

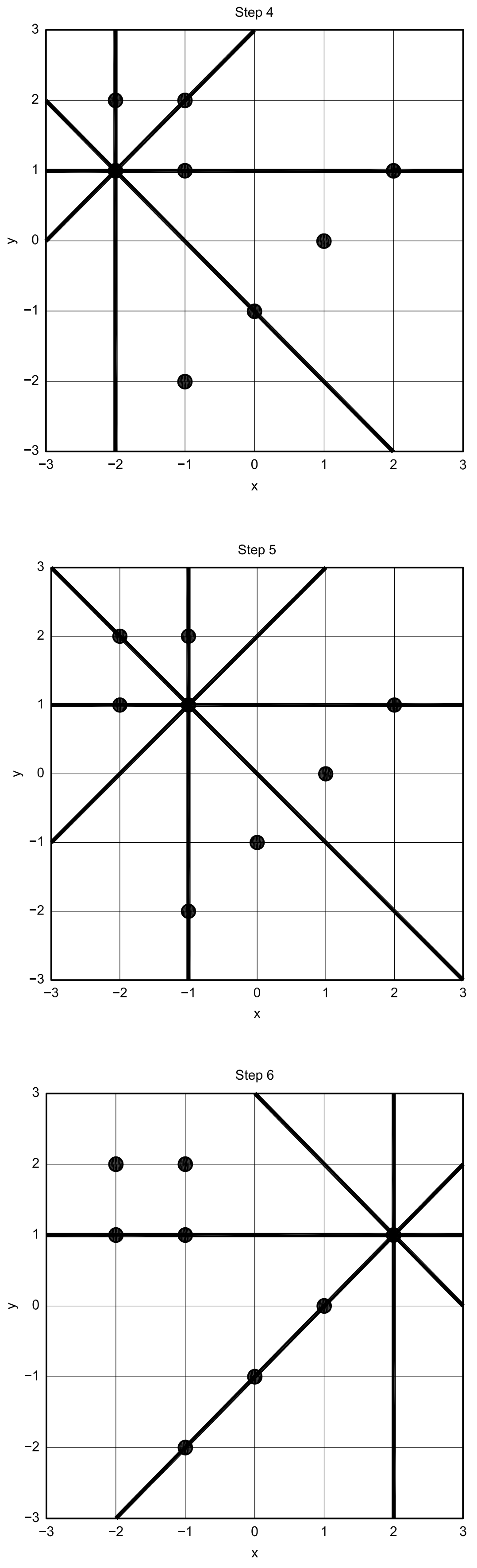
Step 3: Computing Parameter *d* for Pixel (1,0)

q	–x sin *q* + y cos *q*	*d*	*round(d)*
0°	–1 × 0.00 + 0 × 1.00	0.00	0
45°	–1 × 0.85 + 0 × 0.53	–0.71	–1
90°	–1 × 0.89 + 0 × –0.45	–1.00	–1
135°	–1 × 0.09 + 0 × –1.00	–0.71	–1

The Accumulator After Step 3

ACC	*0°*	*45°*	*90°*	*135°*
d = –2	1	0	0	0
d = –1	1	***3***	***1***	***1***
d = 0	***1***	0	1	0
d = 1	0	0	1	1
d = 2	0	0	0	1
d = 3	0	0	0	0

Fig 20—Hough transform—Steps 1 to 3.



Step 4: Computing Parameter *d* for Pixel (–2, 1)

q	–x sin *q* + y cos *q*	*d*	*round(d)*
0°	2 × 0.00 + 1 × 1.00	1.00	1
45°	2 × 0.85 + 1 × 0.53	2.12	2
90°	2 × 0.89 + 1 × –0.45	2.00	2
135°	2 × 0.09 + 1 × –1.00	0.71	1

The Accumulator After Step 4

ACC	*0°*	*45°*	*90°*	*135°*
d = –2	1	0	0	0
d = –1	1	3	1	1
d = 0	1	0	1	0
d = 1	***1***	0	1	***2***
d = 2	0	***1***	***1***	1
d = 3	0	0	0	0

Step 5: Computing Parameter *d* for Pixel (–1, 1)

q	–x sin *q* + y cos *q*	*d*	*round(d)*
0°	1 × 0.00 + 1 × 1.00	1.00	1
45°	1 × 0.85 + 1 × 0.53	1.41	1
90°	1 × 0.89 + 1 × –0.45	1.00	1
135°	1 × 0.09 + 1 × –1.00	0.00	0

The Accumulator After Step 5

ACC	*0°*	*45°*	*90°*	*135°*
d = –2	1	0	0	0
d = –1	1	3	1	1
d = 0	1	0	1	***1***
d = 1	***2***	***1***	***2***	2
d = 2	0	1	1	1
d = 3	0	0	0	0

Step 6: Computing Parameter *d* for Pixel (2, 1)

q	–x sin *q* + y cos *q*	*d*	*round(d)*
0°	–2 × 0.00 + 1 × 1.00	1.00	1
45°	–2 × 0.85 + 1 × 0.53	–0.71	–1
90°	–2 × 0.89 + 1 × –0.45	–2.00	–2
135°	–2 × 0.09 + 1 × –1.00	–2.12	–2

The Accumulator After Step 6

ACC	*0°*	*45°*	*90°*	*135°*
d=–2	1	0	***1***	***1***
d=–1	1	***4***	1	1
d=0	1	0	1	1
d=1	***3***	1	2	2
d=2	0	1	1	1
d=3	0	0	0	0

Fig 21—Hough transform—Steps 4 to 6.

demodulator is at the heart of such magic.

No more slanted pictures, we wrote. Now it is done! From a practical point of view, however, it is not an excuse for transmitting non-M1-compliant signals!

The Video Demodulator

A lot of work and time has been devoted to the synchronization detector described in the first part of this article. SSTV receiving software must also demodulate the video signal. Lastly, it must display it. We will now focus on video demodulation.

From Frequency to Luminance

Can we improve frequency estimation? The reason behind this question is straightforward: Each video sample can be recovered using Eq 12, which is based upon Eq 1:

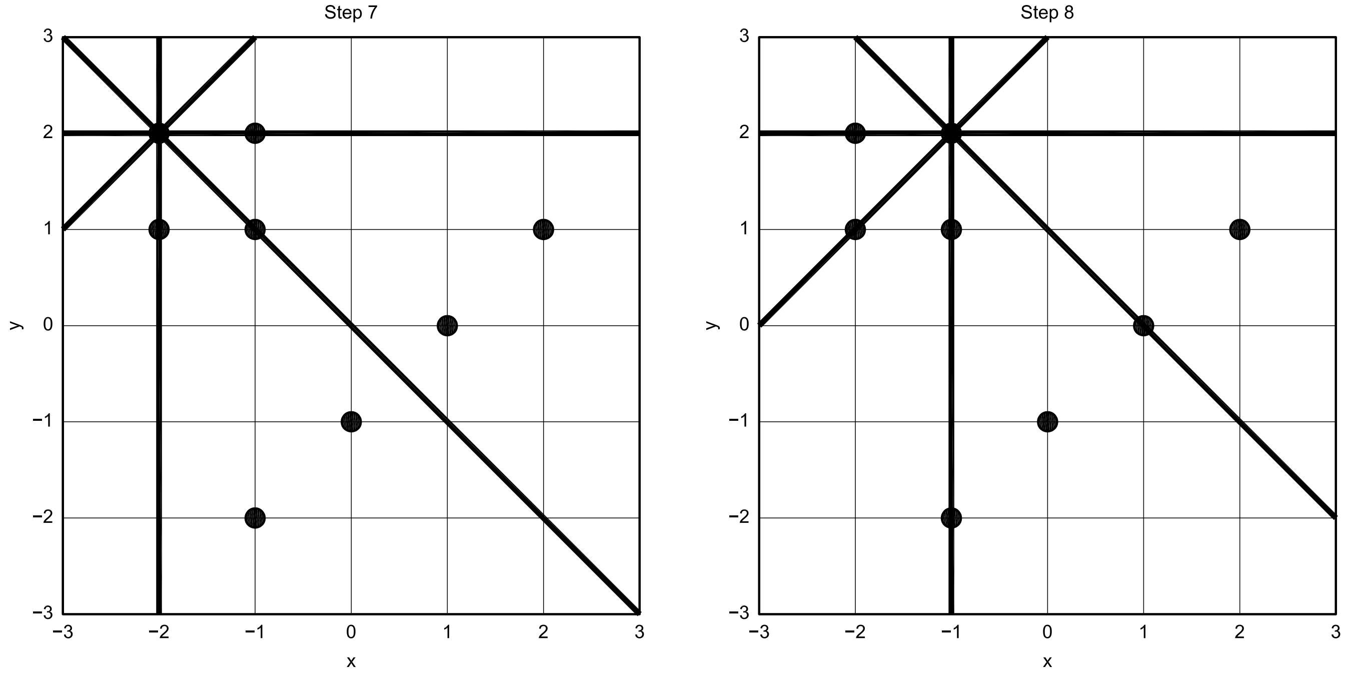
$$lum = \frac{f_1 - f_{\text{black}}}{f_{\text{white}} - f_{\text{black}}} max_{\text{lum}} \qquad \text{(Eq 12)}$$

Eq 12 will output the corresponding luminance, which will be stored in an array for further processing, such as for full-color display.

Unfortunately, such a beautiful equation leads to another question: How can we estimate this unknown frequency f_1? Many solutions have been proposed and used during the past decades. We have already summed up three important strategies at the beginning of this article:

- Analog filters can convert a frequency to a voltage. This voltage is used to drive either an SSTV monitor (see Note 2) or digitized and processed on a computer (see Note 4).
- The signal is crudely digitized thanks to a two-level (or 1-bit) A/D converter. The frequency is then estimated using, for instance, a period estimation (the method used in the Pasokon system).[14]
- The signal is digitized thanks to a decent A/D converter. The digital signal is then processed to estimate the frequency.

As we had already designed a robust synchronization estimator, the video demodulator performance had to



Step 7: Computing Parameter *d* for Pixel (–2, 2)

q	–x sin *q* + y cos *q*	*d*	*round(d)*
0°	2× 0.00+2× 1.00	2.00	2
45°	2× 0.85+2× 0.53	2.83	3
90°	2× 0.89+2× –0.45	2.00	2
135°	2× 0.09+2× –1.00	0.00	0

The Accumulator After Step 7

ACC	*0°*	*45°*	*90°*	*135°*
d = –2	1	0	1	1
d = –1	1	4	1	1
d = 0	1	0	1	***2***
d = 1	3	1	2	2
d = 2	***1***	1	***2***	1
d = 3	0	***1***	0	0

Step 8: Computing Parameter *d* for Pixel (–1, 2)

q	–x sin *q* + y cos *q*	*d*	*round(d)*
0°	1 × 0.00 + 2 × 1.00	2.00	2
45°	1 × 0.85 + 2 × 0.53	2.12	2
90°	1 × 0.89 + 2 × –0.45	1.00	1
135°	1 × 0.09 + 2 × –1.00	–0.71	–1

The Accumulator After Step 8

ACC	*0°*	*45°*	*90°*	*135°*
d = –2	1	0	1	1
d = –1	1	4	1	***2***
d = 0	1	0	1	2
d = 1	3	1	***3***	2
d = 2	***2***	***2***	2	1
d = 3	0	1	0	0

Fig 22—Hough transform—Steps 7 and 8.

match that high-quality level.

There are many well-known methods to estimate the frequency of a signal. One can use a period estimator, a phase-locked loop or a fast Fourier transform (FFT). There are many other methods, such as multiple-window spectrum estimation[15] and the wavelet transform. We have not thoroughly investigated the use of the last two approaches in the context of SSTV; it might be something worth looking at.

It is obvious that for high S/N many methods provide good results—and particularly, a small bias on the estimated frequency. When S/N becomes low, things are not so clear. Studies have been conducted by several authors to evaluate the behavior of estimators with a noisy signal.[16, 17] These papers show that the period estimator is among the worst methods, while the FFT is the least-biased approach.

The discrete Fourier transform (DFT), often implemented as the FFT, is an excellent frequency estimator:[18, 19, 20]

> "When the data consist of uniformly sampled time domain data containing some type of harmonic oscillations, the discrete Fourier transform is almost universally used as the frequency estimation technique. This is done for a number of reasons, but primarily because the technique is fast and experience has shown that the frequency estimates obtained from it are often very good."[21]

Even in the presence of noise, one can extract meaningful information from the power spectrum of the signal. Our frequency estimator relies upon both the discrete Fourier transform and the FFT (for performance issues only).

When the S/N is high, the user expects a high-quality picture. When the S/N is low, or when it suddenly decreases (due to fading, for instance), such quality cannot be maintained. A common thought is that the quantity of information transmitted per second decreases along with the S/N. This is perfectly true (see Shanon's famous paper,[22] for a mathematical explanation). Many everyday-life examples follow this rule: a CW signal is easier to recover using a narrow filter (200 Hz, for instance) than when using an SSB filter (3000 Hz, for instance). The signal can be recovered with the CW filter, whereas it could not with the standard SSB filter.

Keeping this idea in mind, we may wonder what a SSTV demodulator should do? Our answers are summarized below:

- It should provide a high-resolution picture when the S/N is high.
- It should gracefully degrade when the S/N decreases. The resulting video resolution would then be lower than in the theoretical case, as set by G3OQD.
- It should decide on its own which quality is the best during the SSTV demodulation. It should always provide the best trade-off between noise immunity and picture resolution.

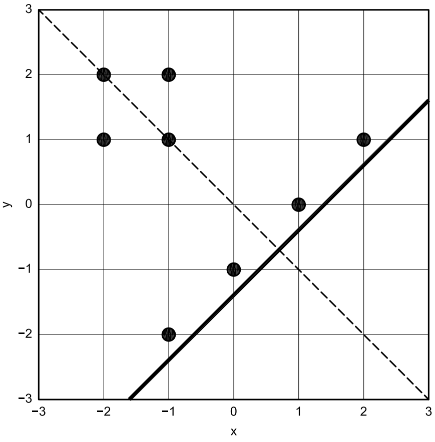


Fig 23—Hough versus LMS approximations.

Can a software demodulator do that? Fortunately, the answer is yes!

Remember that the frequency resolution of the DFT is closely related to the number of samples used to compute the power spectrum. When more samples are used the frequency estimation becomes more accurate.

The only requirement is to design a *magic box* that chooses the best length for the time series the DFT will process. Such a box must estimate the S/N of the SSTV signal. It then converts this S/N to the appropriate length thanks to a function $f(S/N)$. This function can be stored as a predetermined look-up table. Fig 25 shows the idea behind this adaptive scheme.

The S/N Estimator

This part has been very difficult to design. Some methods make use of silence to estimate the power of noise.[23] Our magic box cannot rely upon this approach because a typical SSTV signal lasts about two minutes and is continuous. We have therefore designed a S/N estimator that runs in real-time; that is, during actual SSTV reception.

A simplified SSTV power spectrum is depicted in Fig 26. It comprises the video signal plus some noise (the synchronization signal is not considered here). The purpose of a S/N estimator is to estimate: (1) the power of the SSTV-only signal for a given bandwidth; and (2) the noise power for a given bandwidth.

Let us split the spectrum into three bands, according to Fig 27:

1. A video-plus-noise band, ranging from 1500-2300 Hz.
2. A low-frequency noise-only band, ranging from 300-1100 Hz. The lower bound, 300 Hz, is realistic with an SSB receiver. The upper bound, 1100 Hz, has been chosen to avoid the synchronization signal (centered on 1200 Hz).
3. A high-frequency noise-only band, ranging from 2500-2700 Hz.

The total power in each band is computed thanks to a 2048-sample FFT (f_s = 44,100 Hz) after windowing the incoming signal with a Hanning window (see Reference 19,

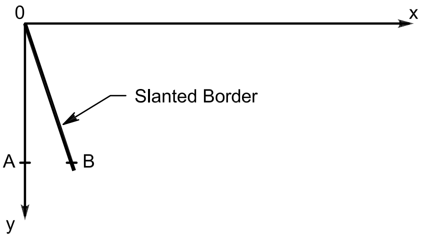


Fig 24—Model of a slanted picture.

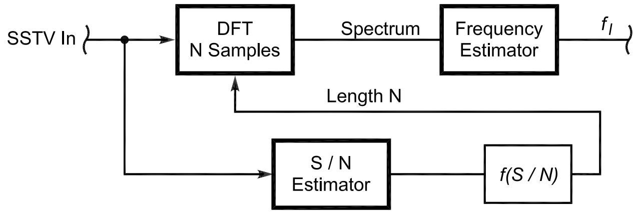


Fig 25—The adaptive scheme for frequency estimation.

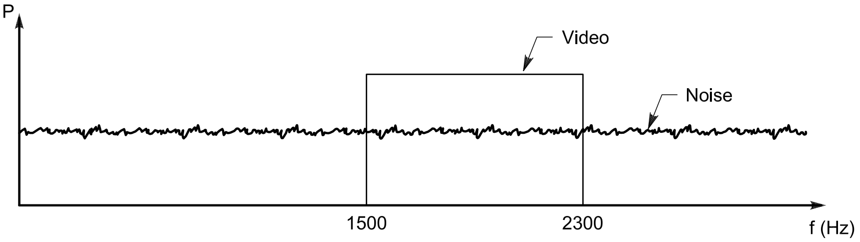


Fig 26—A simplified spectrum of a noisy SSTV signal.

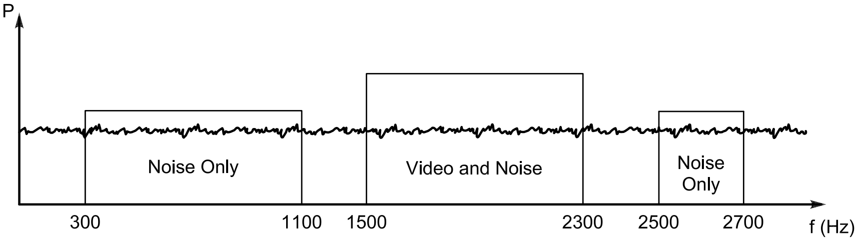


Fig 27—Splitting the spectrum into three bands.

p 20 for some window functions). We then introduce two new variables, namely $P_{\text{video\_noise}}$ and $P_{\text{noise\_only}}$.

- $P_{\text{video\_noise}}$ is the total signal power in the video band (1500-2300 Hz).
- $P_{\text{noise\_only}}$ is the total signal power in the lower band (300-1100 Hz) plus the total signal power in the higher band (2500-2700 Hz).

Under the following assumptions:

- The noise power is constant across all frequencies. If not, one can measure the global information filter response of the receiver and make up for it.
- The SSTV video signal lies from 1500 to 2300-Hz.

One can write:

$$P_{\text{video\_noise}} = P_{\text{signal}} + P_{\text{noise}} \frac{BW\left(P_{\text{video\_}}\right.}{BW\left(\mathit{recei}\right.} \qquad \text{(Eq 13)}$$

$$P_{\text{noise\_only}} = P_{\text{noise}} \frac{BW\left(P_{\text{noise\_only}}\right)}{BW(\mathit{receiver})} \qquad \text{(Eq 14)}$$

where:

$BW(\mathit{receiver})$ = receiver bandwidth (2700 – 300 = 2400 Hz)

$BW(P_{\text{video\_noise}})$ = bandwidth of the video signal (800 Hz for SSTV)

$BW(P_{\text{noise\_only}})$ = total bandwidth used for noise-power estimation. According to Fig 27, it is (1100 – 300) + (2700 – 2500) = 1000 Hz

P_{signal} = SSTV video signal power for a video bandwidth $BW(P_{\text{video\_noise}})$

P_{noise} = noise power for the receiver bandwidth $BW(\mathit{receiver})$.

From Eq 14, one can write:

$$P_{\text{noise}} = P_{\text{noise\_only}} \frac{BW(\mathit{receiver})}{BW\left(P_{\text{noise\_only}}\right)} \qquad \text{(Eq 15)}$$

From Eqs 13 and 15, we get:

$$P_{\text{noise}} = P_{\text{noise\_only}} \frac{BW(\mathit{receiver})}{BW\left(P_{\text{noise\_only}}\right)}$$

$$P_{\text{signal}} = P_{\text{video\_noise}} - P_{\text{noise}} \frac{BW\left(P_{\text{video\_}}\right.}{BW\left(\mathit{recei}\right.} \qquad \text{(Eq 16)}$$

And expanding Eq 16, one has:

$$P_{\text{noise}} = P_{\text{noise\_only}} \frac{BW(\mathit{receiver})}{BW\left(P_{\text{noise\_only}}\right)}$$

$$P_{\text{signal}} = P_{\text{video\_noise}} - P_{\text{noise\_only}} \frac{BW\left(P_{\text{vi}}\right.}{BW\left(P_{\text{n}}\right.} \qquad \text{(Eq 17)}$$

The S/N is then computed by:

$$S/N = 10\log\left(\frac{P_{\text{signal}}}{P_{\text{noise}}}\right) \qquad \text{(Eq 18)}$$

Notice that we can bound the minimum ratio P_{signal} / P_{noise} to 0.01 (–20 dB), a value far below realistic conditions for SSTV reception. This value will be used in the sequel.

Results

A Martin M1 signal is generated. Gaussian noise is then created and filtered by the same low-pass filter used for synchronization detector tests, an 8th-order Butterworth IIR with a cutoff frequency of 2500 Hz. It simulates the filters of the receiver.

Nine different amounts of v_{noise} were added to the pure M1 signal. We have arbitrarily chosen: 3, 2, 1.5, 1, 1/1.5, 1/2, 1/3, 1/6 and 1/10 times the original amplitude of the noise. Please notice that such a simulation only focuses on noisy SSTV signals and that it does not consider QRM. The estimated S/N is plotted for the nine experiments, along with the input noise power, in Fig 28.

Here are a few comments about that figure. The reference noise power, denoted as "o = input," is computed by 10 log$((v_{\text{noise}})^{-2})$. The amplitude of the signal is one. Notice there is an offset between the input noise power and the estimated S/N. That is caused by the way the noise power is computed; that is, without appropriate scaling. It is not important for our application, as a gain between these values is converted into an offset in the logarithmic scale.

We already know the estimator is biased. Another interesting simulation gives an idea about the standard deviation from the mean $s_{S/N}$ of this estimator.

Using the same algorithms 100 times for each experiment, one can compute the standard deviation $s_{S/N}$ from the mean value $\mu_{S/N}$ of the S/N. We have displayed the results in Fig 29: the mean value $\mu_{S/N}$ as "×," along with the upper "Δ" and lower "∇" bounds. These bounds are related to the mean value thanks to: $\mu_{S/N} + \sigma_{S/N}$ and $\mu_{S/N} - \sigma_{S/N}$.

We may conclude that the estimator performs well for S/N > 0 dB. The standard deviation from the mean increases for lower values—the results will be less accurate. Moreover, we can see in Fig 29 that the method tends to underestimate the S/N for very low values. Anyway, the dynamic range of the estimator extends to limits beyond realistic receiving conditions. It is indeed very difficult to recognize a SSTV signal at a very low S/Ns.

Estimating the Luminance

As explained above, the luminance demodulation is based upon a DFT of the incoming signal. The number of samples used to compute the DFT is driven by the estimated S/N, as a function $f(S/N)$. We are now going to describe the luminance demodulation system and the trade-off we had to face.

High-Quality SSTV

How many samples do we have to use to get an accurate estimate of the SSTV signal? We have already explained the idea behind our adaptive scheme. We now focus on the highest-quality demodulator. It is the upper bound of the luminance demodulator. Using a trial-and-error process, we finally decided to use a 37-point Chebyshev window (see Fig 30).

The windowed signal is then fed into a DFT. This DFT only computes the relevant part of the spectrum, thus saving many microprocessor cycles. The output power spectrum is then followed by a parabolic interpolation—a common practice in frequency estimation.

This choice leads to excellent frequency estimations for high S/Ns. Moreover, the horizontal resolution allowed by such a number of points is compatible with G3OQD specifications.

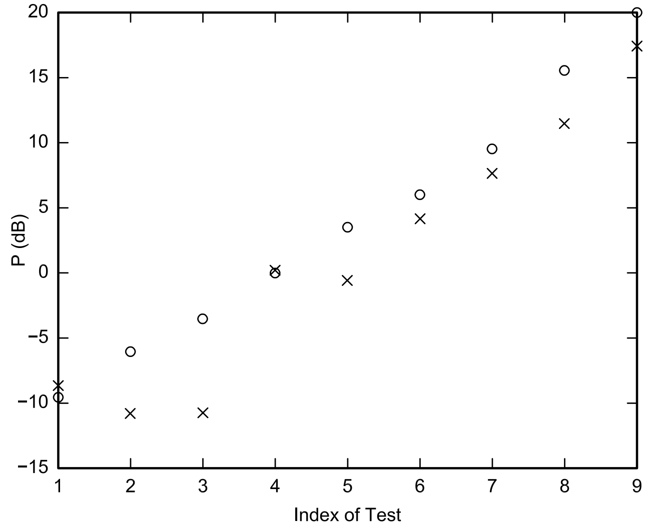


Fig 28—Original noise power (o) and estimated S/N (x).

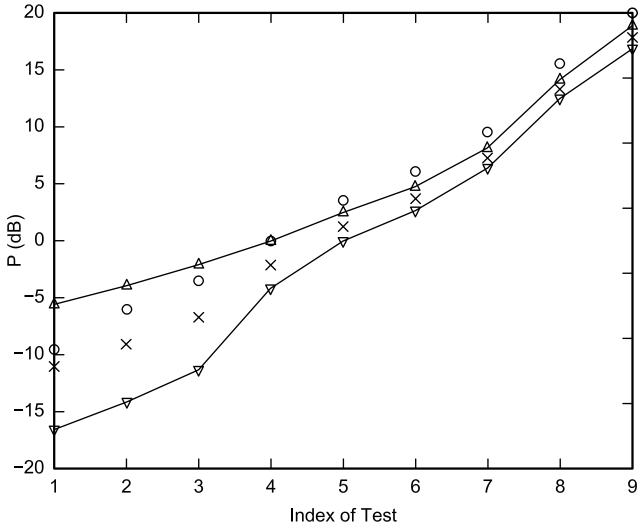


Fig 29—Standard deviation from the mean of the estimated S/N (o = input noise; x = estimated S/N).

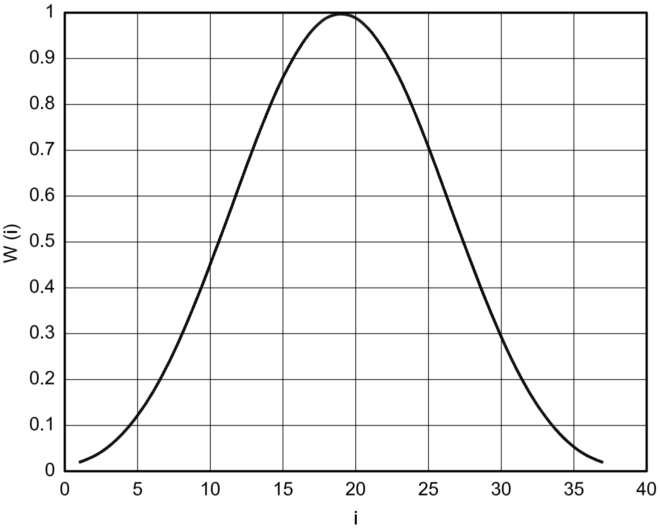


Fig 30—37-point Chebyshev window.

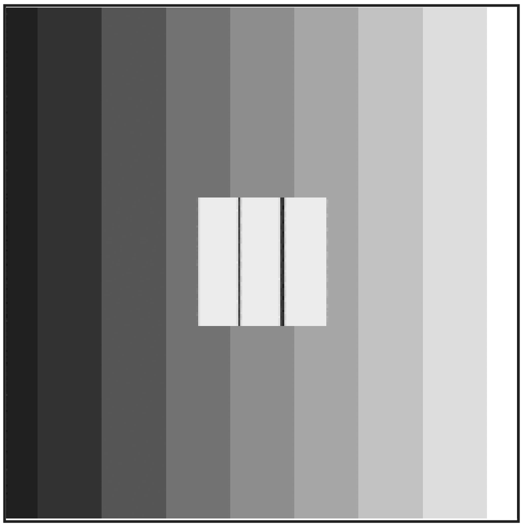


Fig 31—The reference picture

Some simulation results of this demodulator are presented below. We have used the same test picture as in Fig 6. The demodulated picture is presented in Fig 31. The luminance of the 128th line of the test picture is displayed in Fig 32. The luminance of the demodulated picture is displayed in Fig 33. One can check that the demodulated signal is very close to the reference. The only difference is a slight shift between the original and the demodulated pictures. It produces the black vertical line on the right. It will be fixed in a future release of the software.

This horizontal shift prevents us from using the usual peak S/N to estimate image quality. A far better quality estimate is to estimate bias and the standard deviation from the mean across a vertical line. The choice of a vertical line allows us to use samples that are not correlated. That topic is far beyond the scope of this paper.[24]

A Demodulator for a Low S/N

When the S/N is low, the software chooses a longer window—a Hanning window. The lengths used in the software range from 64-2048 points. The windowed signal is then followed by a FFT. Only the relevant part of the estimated spectrum is used to recover the frequency. Any maximum of the power spectrum outside of the video band (1500-2300 Hz) is excluded, thus helping to mitigate picture degradation induced by QRM.

The very same software has been used to demodulate a very noisy signal (more noise than SSTV signal). Two resulting pictures are presented: One was demodulated with a 37-sample DFT (Fig 34), and another demodulated with the adaptive, S/N-based demodulator (Fig 35).

We have also plotted the same 128th line using the following convention: The continuous line is for the adaptive scheme; the dots are for the 37-point DFT (see Fig 36). We can conclude the adaptive scheme, under these conditions, outperforms the basic demodulator.

Implementation Issues

The whole software has been coded in *C*++, some parts being *C* functions with a *C*++ wrapper. It was developed using Borland *C*++ *Builder* version 3

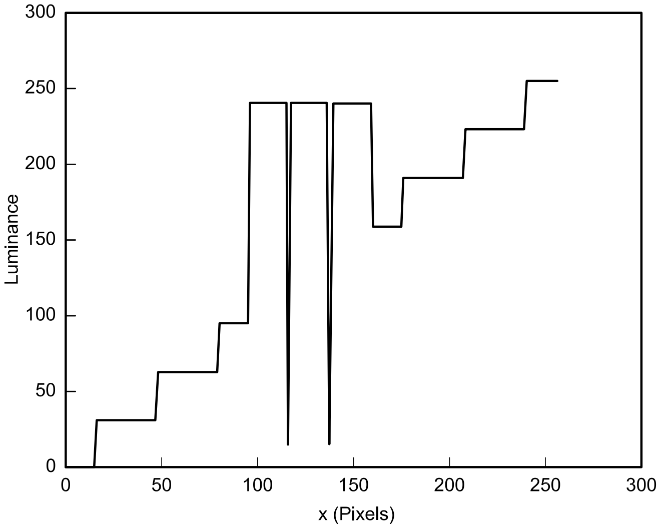


Fig 32—Picture demodulated with the 37-point DFT.

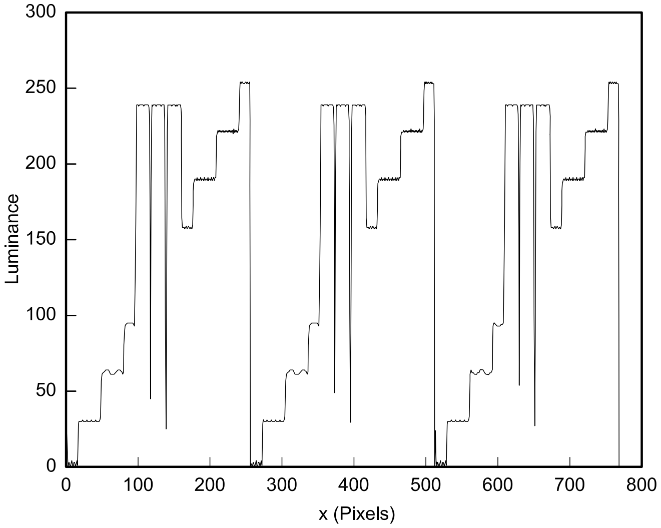


Fig 33—Demodulated signal, *N* = 37, line 128.

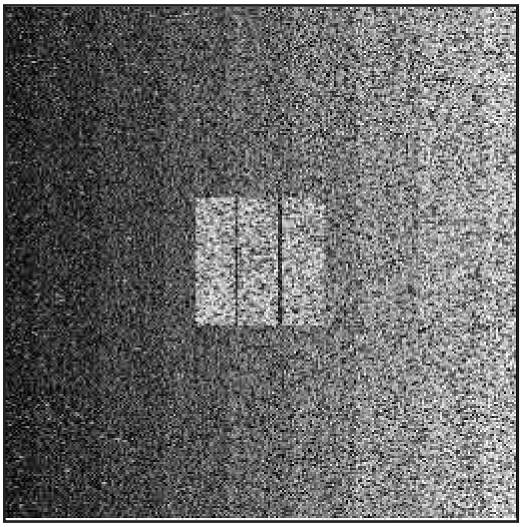


Fig 34—Noisy picture demodulated with a 37-point DFT.

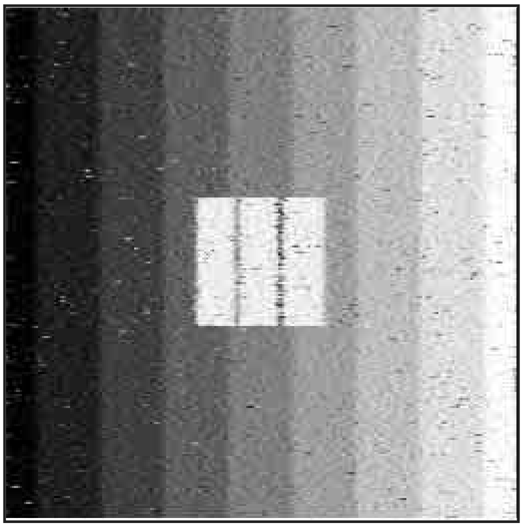


Fig 35—Noisy picture demodulated with the adaptive method.

(professional edition for students—the project started a while ago). More recently, it has also been compiled with version 5 of the same product without trouble.

The DFT relies upon two kinds of modules: a plain DFT and a FFT. The DFT is used for N = 37 points. Using some recent *C*++ coding techniques, the compiled code provides us with a very short execution time on a processor from the Pentium family. It relies upon a clever use of templates for the dot product of vectors.[26] As this topic is far from the scope of this paper, but of high interest to anyone "cooking" and coding DSP modules, we invite the interested reader to view references on the topic.[27, 28]

The FFT is used for both the video demodulator and the S/N estimator. We have used the "Fastest Fourier Transform in the West" library, also known as FFTW (see **www.fftw.org**).[29, 30] This software is licensed under the GNU General Public License (GPL, see **www.gnu.org**). The sound capture under the *Windows* operating system is a modified version of the code published in *Dr Dobbs Journal*.[31]As it uses

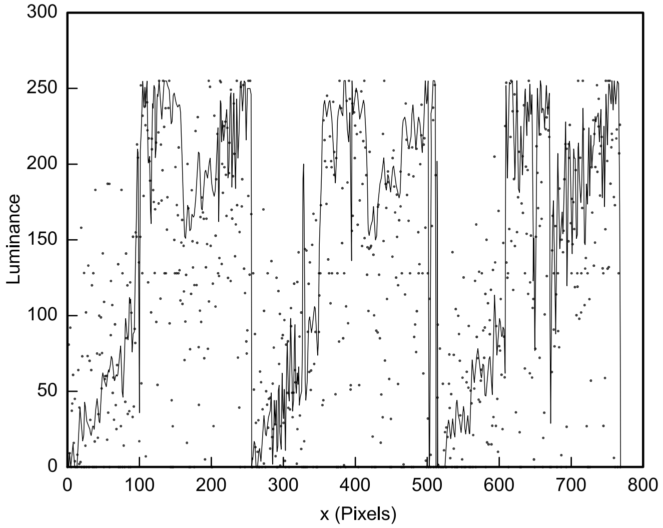


Fig 36—A noisy picture demodulated with both *N* = 37 and adaptive techniques, line 128.

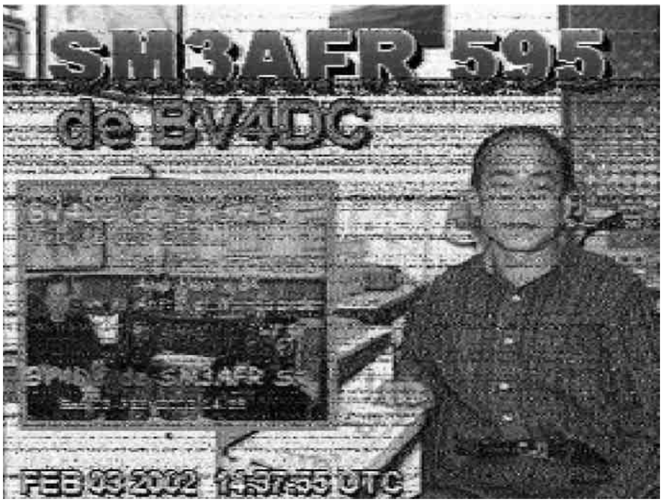


Fig 37—BV4DC's picture, *N* = 37.

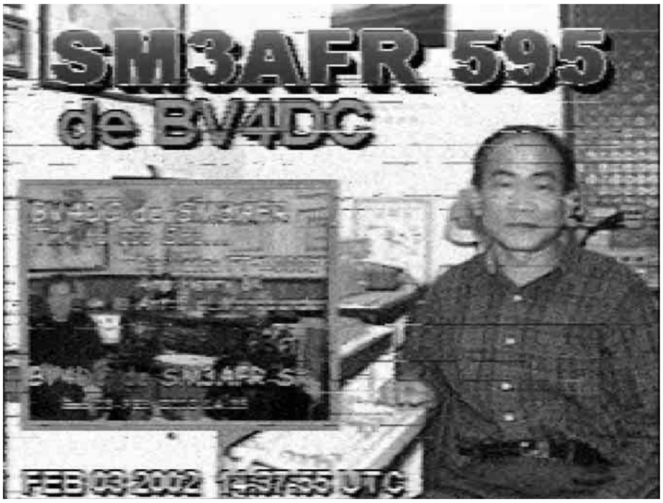


Fig 38—BV4DC's picture, adaptive.

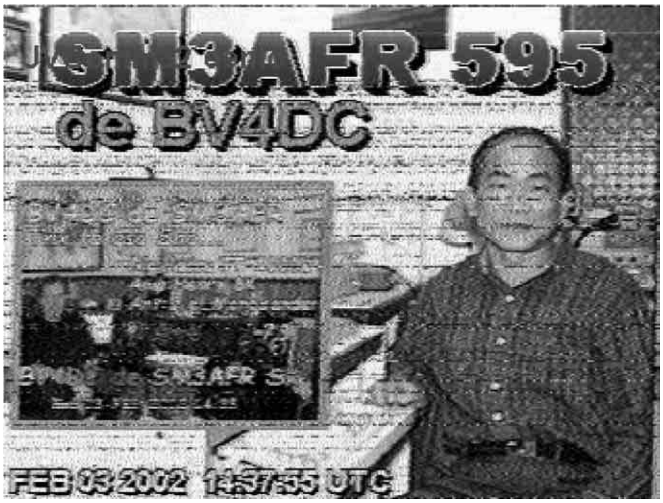


Fig 39—Same signal, demodulated with other common SSTV software.

Table 1—IIR Synchronization Filter Coefficients

b_0	b1	b2	b3	b4
1.9897E–004	0	–3.9794E–004	0	1.9897E–004
a_0	a_1	a_2	a_3	a_4
1	–3.9024E+000	5.7672E+000	–3.8245E+000	9.6050E–001

software released under the GPL, our SSTV software is also released under the GPL. Look for the source code at **roland.cordesses.free.fr/sstvrep** under "RealTime Processing."

Experimental Results

The program described here, running under the *Windows* operating system, can only receive M1 SSTV signals. While not having all the bells and whistles found in today's software, it has been designed to match various experimental conditions. The way the picture appears on the PC screen is close to any SSTV program, but we can point out some unusual features our software presents.

- The flux of SSTV lines is not constant and changes with the levels of noise and interference.
- When one tunes a SSTV transmission the beginning of which has been missed, some strange colors are first presented on the screen; but after a few seconds, the system resynchronizes itself even in presence of noise or QRM.
- At the end of a slanted picture, the program "unslants" it automatically.
- The result file presents some statistical information associated with a received picture and helps one understand how the program processed the SSTV signal.

Moreover, the user can store not only the received picture (BMP file) but also the associated audio signal (WAV file). It is therefore possible to replay the audio file as often as desired to experiment with different processing configurations. During our experiments, we produced many audio CD files. Thus we could test not only the various capabilities of our program, but also compare them to other popular SSTV software.

Figs 37 through 39 show examples of a picture transmitted by BV4DC on the 15-meter band. The propagation changed quickly and the signal, which was good at the beginning of the transmission, suddenly dropped with a slow and deep fading. Some interference appeared at the end of the picture.

Fig 37 presents the BV4DC picture as received with the 37-sample DFT, while Fig 38 shows the same signal demodulated with the adaptive, S/N based demodulation. It is clear that this last picture is far less noisy than the first, but it does not have the same sharpness. The adaptive algorithm is doing a nice job under those poor conditions, and we can accept easily the small loss of picture resolution.

Fig 39 is a picture from the same audio signal produced by one of the best available programs we have tested. The picture quality is slightly better than that of the 37-samples FFT of our software, but far behind that of the adaptive method shown in Fig 38.

Conclusion

We have presented some unconventional ways of processing SSTV synchronization signals and an adaptive approach to extract luminance information. The results obtained with our program making use of these algorithms are spectacular, especially when receiving conditions are poor.

As a final word, when we were reviewing the papers presented in the bibliography, we were very impressed by the conclusion in Reference 3: "We can be assured that picture and graphical processing by computer will shortly pervade many aspects of our lives." That sentence, written more than 30 years ago and before the introduction of the microprocessor, was really prophetic and has been subsequently verified!

Acknowledgements

First of all, I would like to thank Professor Jean Gallice. Back in 1996, he spent many hours casting a new light on frequency estimation. He is the man behind our DFT idea.

Thanks to G3OQD, the father of the popular Martin M1 mode. He kindly sent us the exact timings of the M1 mode (see Note 1).

And lastly, we must mention (in alphabetical order): Donovan, Bob Dylan, Tom Paxton; Peter, Paul and Mary. Thanks for the musical background!

Notes

[1] Martin Emerson G3OQD. M1 technical specifications in personal communication, February 1993.

[2] C. Macdonald, "A Compact Slow Scan TV Monitor," *QST*, Mar 1964, pp 43-48.

[3] Th. Cohen, H. L. Husted, and P. R. Lintz, "Computer Processing Slow-scan Television Pictures," *ham radio*, Jul 1970, pp 30-37.

[4] J. R. Montalbano, "The ViewPort VGA Color SSTV System," *73 Amateur Radio Today*, Aug 1992, pp 8-16.

[5] R. Cordesses and L.Cordesses, "SSTV couleur et Fax sur compatible IBM PC," (in French) *Radio REF*, juillet-aôut 1994, pp 23-27.

[6] *The ARRL Handbook for Radio Amateurs* (Newington, Connecticut: ARRL, 2000; ISBN 0-87259-183-2), Chapter 12, p 12.44.

[7] C. Marven and G. Ewers, *A Simple Approach to Digital Signal Processing* (Texas Instruments, 1994; ISBN 0-904 047-00-8), Chapter 4.

[8] An analog Bessel filter is a better choice than an analog Butterworth filter for this kind of application. Unfortunately, we are not able to design a digital Bessel filter.

[9] The so-called Scottie 1 or S1 color mode may use a longer horizontal synchronization signal. Unfortunately, we have never been able to get information from the creator of this mode. The author of *JVComm32*, DK8JV, has kindly sent us the timings he uses for his software for the S1 mode. Design and simulations should be done with a 10-ms burst for the S1 mode.

[10] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C* (Cambridge University Press, 1994; ISBN 0-521 43108 5) Chapter 15.

[11] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures, *Communications of the ACM* (Association for Computing Machinery), 15:11-15, Jan 1972.

[12] P. V. C. Hough, *Method and Means for Recognizing Complex Patterns,* US Patent Specification US3069654, Dec 1967.

[13] J. Illingworth and J. Kittler, "A survey of the Hough transform," ACM *Computer Vision, Graphics, and Image Processing*, Vol 44, Issue 1, pp 87-116, Aug 1988.

[14] J. Langner, "Slow-Scan TV: It isn't Expensive Anymore!" *QST*, Jan 1993, pp 20-30.

[15] D. J. Thomson, "Highlights of Statistical Signal and Array Processing, Multiple Window Spectrum Estimate," *IEEE Signal Processing Magazine*, Sep 1998, 15(5): 30-32.

[16] D. Sirmans and B. Bumgarner, "Numerical Comparison of Five Frequency Estimators," *Journal of Applied Meteorology*, Sep 1973, 14: 991-1003.

[17] F. Baudin, "Estimateurs de Fréquence pour Mesurer la Dérive Doppler Sodar," (in French) Technical report: *Note Technique CRPE/CNET 38*, Jan 1977.

[18] C. Marven and G. Ewers, *A Simple Approach to Digital Signal Processing* (Texas Instrument, 1994; ISBN 0-904 047-00-8) Chapter 5.

[19] D. Smith, "Signals, Samples, and Stuff: A DSP Tutorial (Part 3)," *QEX*, July 1998, pp 13-27.

[20] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C* (Cambridge University Press, 1994; ISBN 0-521 43108 5) Chapter 12.

[21] G. Bretthorst, *To Appear in Maximum Entropy and Bayesian Methods* (Netherlands: Kluwer, 2000), Chapter "Nonuniform Sampling: Bandwidth and Aliasing."

[22] C. E. Shannon, "Communication in the Presence of Noise," *Proceedings of the IRE*, 37: 10-21, 1949.

[23] A. Vizinho, P. Green, M. Cooke, and L. Josifovski, "Missing Data Theory, Spectral Subtraction and Signal-to-Noise Estimation for Robust ASR: an Integrated Study," in *Proceedings of EuroSpeech '99*, pp 2407-2410, Budapest, 1999.

[24] We have used the bias and the standard deviation from the mean to design and qualify the demodulator. Although it is a powerful tool, its results are hardly linked to the visual quality of the picture. The same problem applies to the aforementioned PS/N (see Note 25).

[25] M. J. Nadenau, S. Winkler, D. Alleysson and M. Kunt, "Human Vision Models for Perceptually Optimized Image Processing—A Review," submitted to *Proceedings of the IEEE*, Sep 2000.

[26] T. L. Veldhuizen, "C++ Templates as Partial Evaluation," In *ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation* (PEPM'99), pp 13-18, San Antonio, Texas; Jan 1999.

[27] Todd L. Veldhuizen, "Expression Templates," *C++ Report*, 7(5): 26-31, Jun 1995. (Reprinted

in *C++ Gems*, Editor Stanley Lippman.)

[28] T. Veldhuizen and K. Ponnambalam, "Linear Algebra with *C++*-template metaprograms," *Dr. Dobb's Journal*, 21(8): 38-44, Aug 1996.

[29] M. Frigo and S. G. Johnson. "FFTW: An Adaptive Software Architecture for the FFT," in *ICASSP 1998*, volume 3, pages 1381-1384, 1998.

[30] M. Frigo, "A fast-Fourier-transform compiler," *Proceedings of the ACM SIGPLAN '99 Conference on Programming Language Design and Implementation*, Atlanta, Georgia, United States, 1999, pp 169-180.

[31] R. Cook, "Real-Time Sound Processing," *Dr. Dobb's Journal*, Oct 1998.

Lionel Cordesses first discovered SSTV on his father's (F2DC) P7-phosphor SSTV monitor when Lionel was 6 years old. Ten years and a few homemade receivers later, he designed his own PC-XT based SSTV demodulation software.

1994 saw the beginning of a bench program to test image-decoding algorithms. Soon limited by the performance of the processor, the project revived and finally ended in November 2001, thanks to more number crunching power, and a stronger theoretical knowledge.

He is now an Electronics Engineer (1997) with a PhD in electronics (2001). His research interests include GPS-based control of farm vehicles and analog electronics. He then joined the R&D electronics department of RENAULT Agriculture, a company manufacturing farm tractors. He now focuses on control systems and embedded controllers. As an engineer, he always tries to explain complex theories with simple examples, trying to bridge the theory-practice gap. He is an IEEE and ACM member.

His hobbies include building model aircraft and their related electronics and flying indoor-light models. He also enjoys drawing Donald Duck and other characters of the Duck family in the Carl Barks' way. He is also fond of French and American film noir of the 1940s and 50s.

Roland Cordesses, F2DC, was licensed in 1962, and he has been a member of ARRL since 1964. He graduated as an Electronics Engineer and is presently a Research Engineer at OPGC, an Observatory devoted to atmospheric and earth sciences he joined in 1973.

For nearly 30 years, he has been working in the design and development of radar systems for remote sensing of the atmosphere. Recently, he designed and field-tested the world's first Doppler radar specifically devoted to the monitoring of volcanic eruptions. During his career, he has presented or published many papers related to electronics and geophysics.

Roland's Amateur Radio interests mainly deal with homebrewing: Over the years, he has designed and built many receivers, transceivers and SSTV projects.

Aside from Amateur Radio, he enjoys building and flying model aircraft (homemade transmitters and recei-vers, of course) and hiking in the mountains. □□

New Book

RF POWER AMPLIFIERS

By Mihai Albulet

Noble Publishing, Atlanta, Georgia, 2001; ISBN 1-884932-12-6; $75, hardcover, 376 pages.

This book covers the basics of solid-state RF power-amplifier design, including discussions of circuit classes A through S and many variations on those topologies. It provides excellent insight about the theory and physical basis of operation for such amplifiers. Vacuum tubes or valves are not covered.

The material is wholly theoretical in its presentation. As the author states, the book is not a practical guide to amplifier design but an analysis of design approaches and performance limitations. In his preface, he asserts, "Theory is the best practice and a good theoretical understanding is the quickest way toward achieving practical results." We agree that a good understanding of things is necessary, but practical considerations are equally important to producing working units, especially those intended for mass production.

While much of the book is replete with mathematical models, its treatment of amplifier stability lacks such rigor. The six pages devoted to that subject explain neither how to mathematically analyze designs for stability nor how to model the effects of feedback. The reader is referred instead to a Russian text for details of linear system analysis. Basic physical sources of instability are described though, giving the reader a good overview of how to combat oscillations in solid-state designs.

The author writes, "There is no evidence of successful use of large-signal *s* parameters in designing RF PAs above a few watts of output power." On the next page we find, "The most popular RF PA design technique is based on the large-signal input and output impedances of the active device." Several references are cited. He briefly describes the so-called "load-pull" technique, citing Motorola AN1526. He states that it can provide useful information about device performance but tells us neither what that useful information is nor how to use it. Instead of discussing load-pull, the author describes how to measure the load impedance that gives maximum gain at the desired output power. Gain-leveling, IMD and power-combined modular designs are also given sparse treatment.

Despite its lack of practical considerations, *RF Power Amplifiers* provides a fine introduction to the subject for those who know little or nothing about it. We recommend it to those who want to get the big picture. The material includes sufficient detail to get started on a design, especially for switch-mode amplifiers. For the nitty gritty, you may need to go to some of the many excellent references included at the end of each chapter. Mihai Albulet is an engineer with Microsoft, where he is involved with wireless design and development.

—Doug Smith, KF6DX □□

The 17-Meter Ragchewer

Here's a simple PC-controlled, single-band transceiver that gets rave reports for its transmit audio quality. It receives well, too. Come read about it.

By Rod Brink, KQ6F

I'm a 17-meter fan. Most 17-meter operators are gentlemen, and many of them like to dabble in enhanced audio, which makes their signals easy to listen to. This is the kind of coffee shop I like to visit regularly. So I decided to build a monoband rig specifically aimed at fitting in. To make the task a little easier, I chose to leave out some of the functions normally found in commercial transceivers: passband tuning, RIT, XIT and so forth. Even CW capability was left out. I'm a very simplistic operator and I rarely use those functions anyway.

25950 Paseo de los Robles
Corral de Tierra, CA 93908
rodbrink@redshift.com

My background is one of engineering in various areas including the computer industry. For years, I've been comfortable in front of a keyboard and monitor. So it was an easy decision to make the rig computer-controlled. Moreover, it made packaging easier, as I didn't have to worry about knobs, buttons and displays. At first I toyed with the idea of making a DSP rig, but that's an area in which I have no experience and little knowledge. The learning curve appeared to be very steep. In addition, the cost of a hardware/software system for developing the firmware seemed prohibitive. So instead, I opted for a more traditional approach involving mixers and crystal filters. Nevertheless, I vowed to make the very best crystal filters I could, and to throw in an occasional innovation.

Receiver Block Diagram

Refer to Fig 1. A look at the receiver block diagram reveals the single-conversion architecture. Single-conversion offers various advantages:

1. Less components and easier to build than multiple-conversion designs
2. Fewer spurious responses (birdies) owing to fewer mixers
3. In general, single-conversion designs have greater dynamic range, again owing to fewer mixers. The mixer chosen in this particular design partly negates that advantage. More on that later.

On the other hand, single-conversion radios generally lack the selectivity offered by multiple conversion. To get some of that lost selectivity back requires better-than-average bandpass filters. So I chose 14-pole crystal filters over more conventional 6- or 8-pole filters.

The ubiquitous NE602 [SA602 is a commercial equivalent—Ed.] is used three places in the receiver and twice in the exciter. This part has been around for a number of years, but still offers several advantages.

1. Combination double-balanced mixer and oscillator

2. 17 dB of conversion gain (as opposed to 6 dB conversion *loss* for diode mixers)

3. Reasonable noise figure (5 dB)

4. Works with only 300 mV P-P LO input. There is no need for LO amplifiers, as is the case with diode mixers.

5. Self-biased for a low parts count.

Unfortunately, the NE602 has

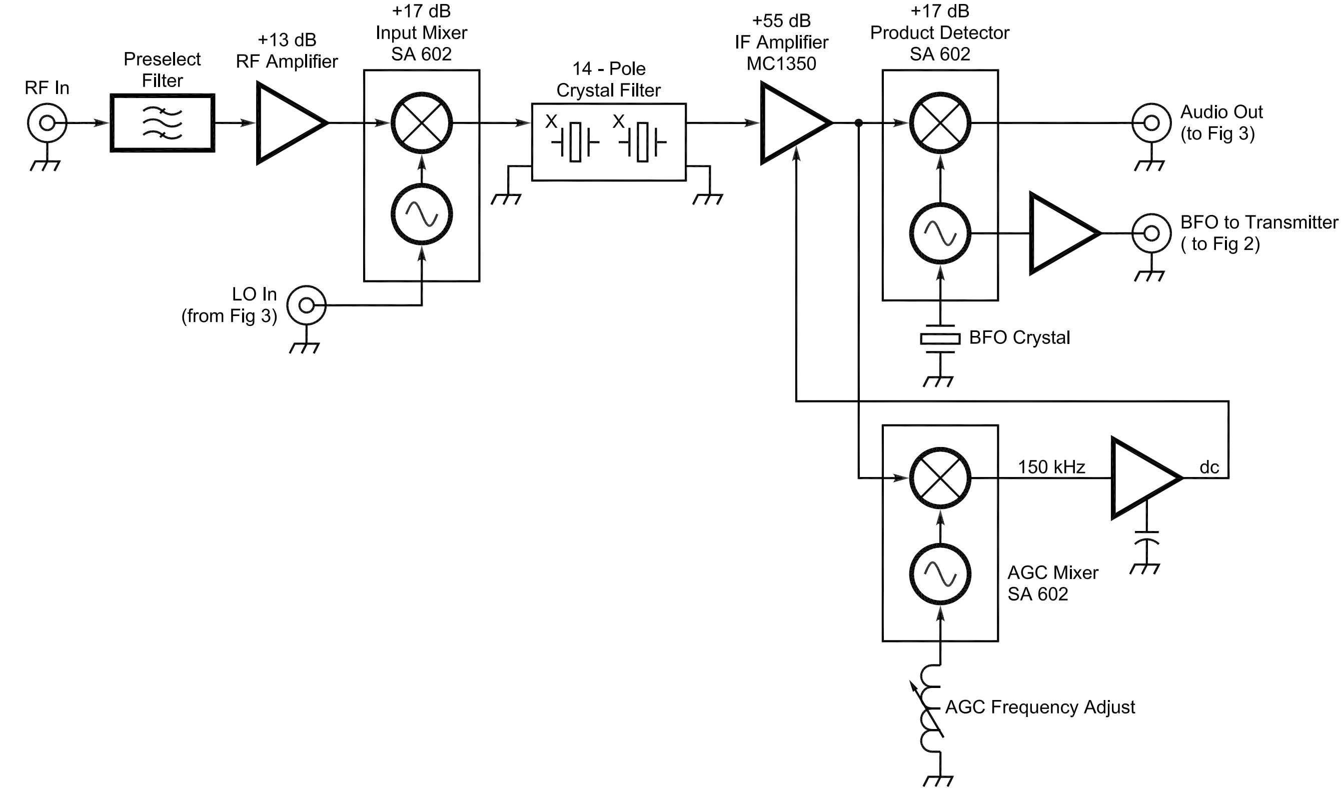


Fig 1—Receiver block diagram.

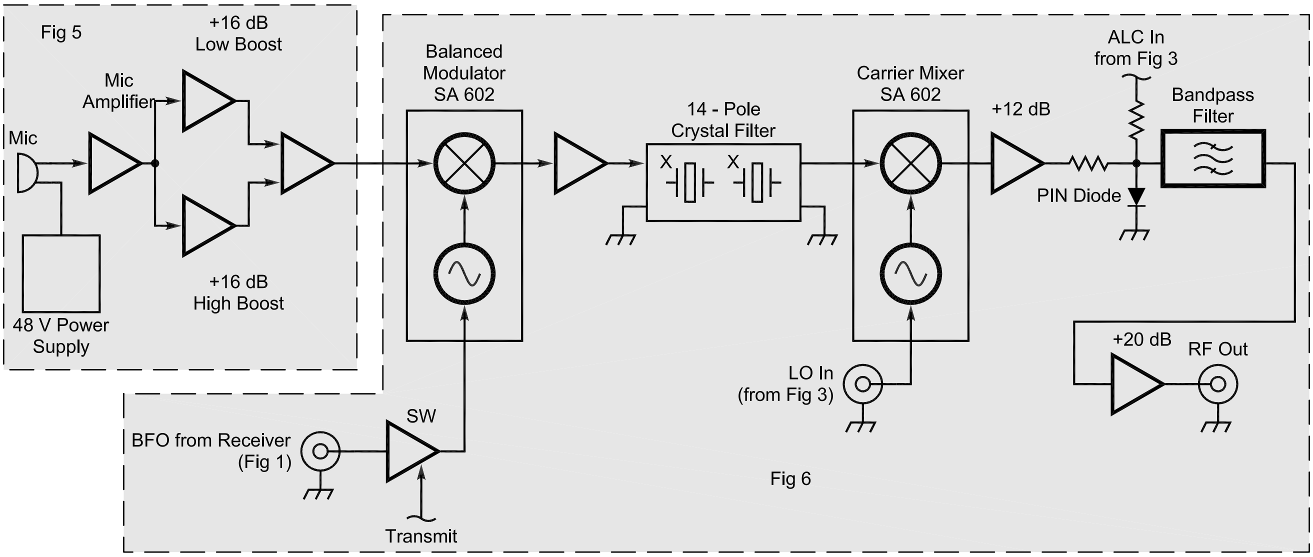


Fig 2—Transmit exciter block diagram.

fairly limited dynamic range owing to its –13 dBm third-order intercept point. For this reason, I was sorely tempted to use the MC1496, which is a more robust part. In the end, the circuit simplicity offered by the NE602 won out. Fortunately, signals on the 17-meter band are not that loud (at least with my antenna), so having a huge dynamic range is unnecessary.

The IF is 8 MHz, which came out of the crystal filter design (covered later in detail). This is a high IF and requires some tricks in the design of the AGC circuit. Nevertheless, it has the advantage of placing the image frequencies far enough away from the desired signal to ease the design of the preselector filter.

For the AGC to work at low signal levels requires gain in the AGC amplifier. The AGC input is at 8 MHz; unless something else is done, the AGC amplifier would have to be an 8-MHz amplifier, similar to the IF amplifier. Having two high-gain, high-frequency amplifiers next to one another on the PC board would likely present some serious instability and cross-talk problems. Instead, I chose to heterodyne the IF signal to approximately 150 kHz using another NE602. Now, the AGC amplifier can be relatively mundane. The BFO is crystal-controlled and is shared by both the receiver and exciter.

Transmit Exciter Block Diagram

See Fig 2. The exciter is designed to produce nice-sounding SSB audio. It includes:

1. A 48-V dc-to-dc switching power supply, for supplying voltage to a studio-quality capacitor microphone.
2. A built-in equalizer circuit for boosting both low and high audio frequencies.

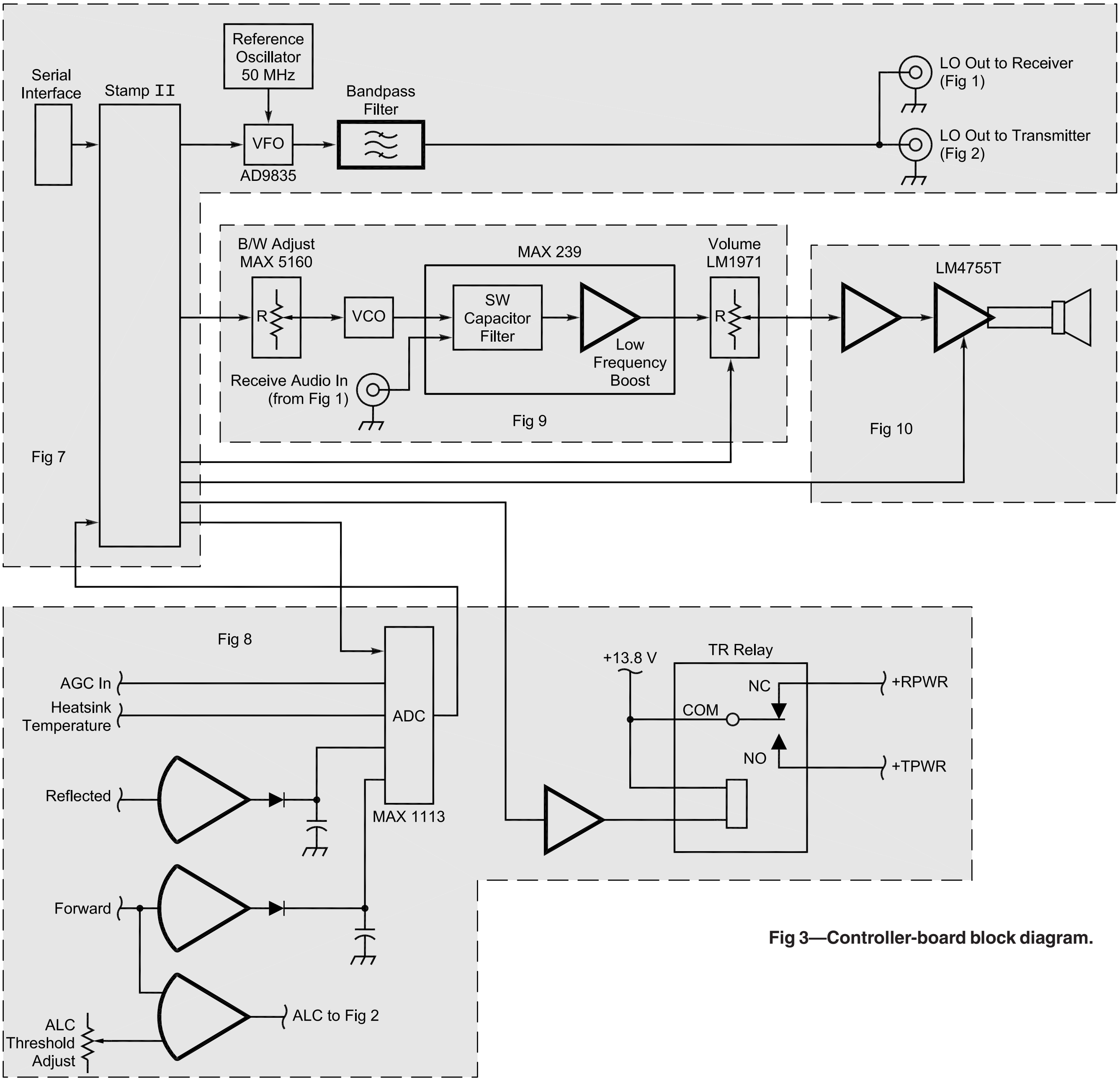
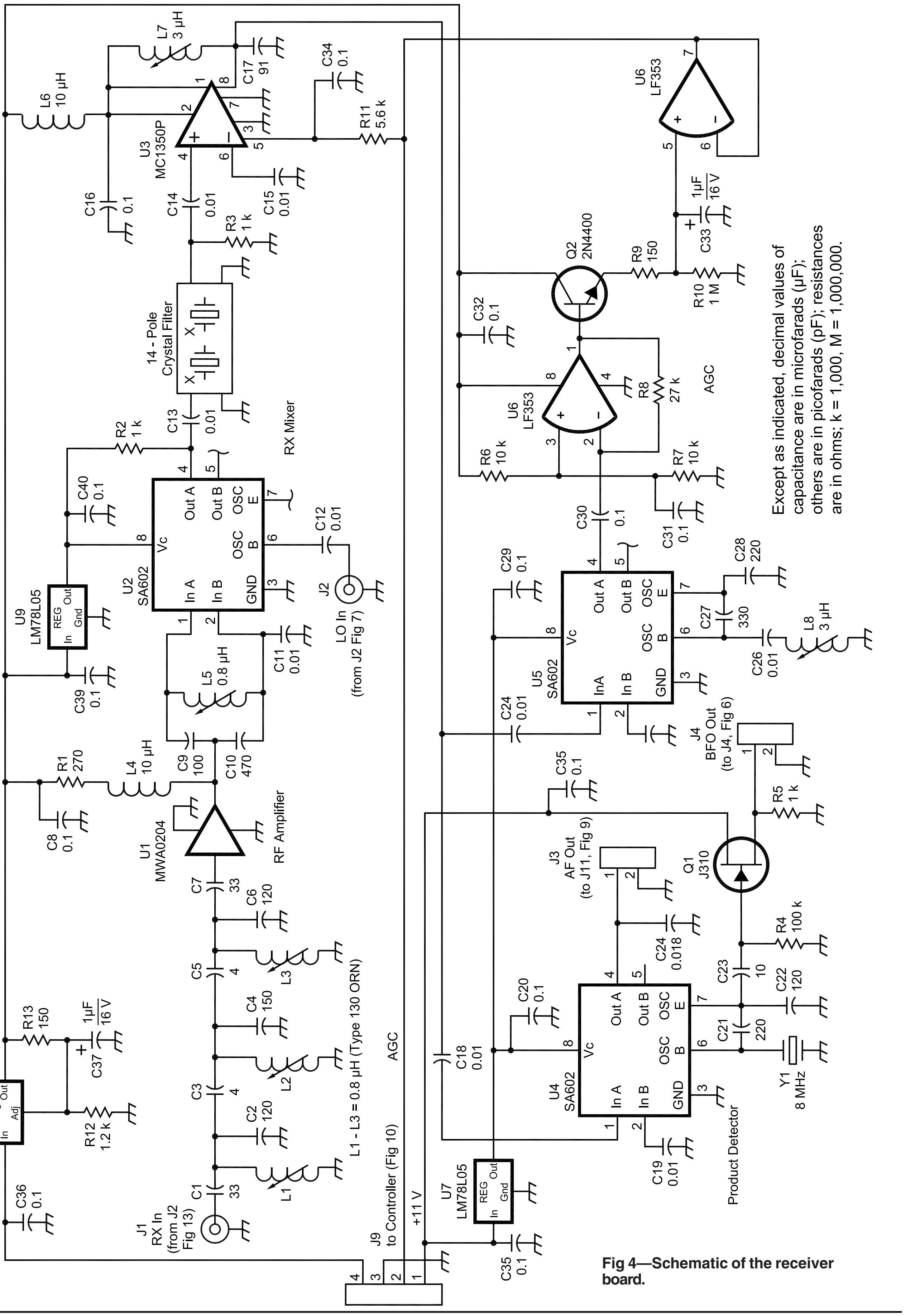


Fig 3—Controller-board block diagram.



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

Fig 4—Schematic of the receiver board.

3. Careful attention to interstage signal levels to minimize IMD.

4. 3-kHz-wide crystal filters.

Single-conversion topology was again chosen for simplicity.

Controller Board Block Diagram

Refer to Fig 3. At the controller heart is the Stamp II module. This is a very nice hybrid part that includes a microcontroller, 2-kByte EEPROM, clock, power-supply regulator and serial interface, all in a 24-pin package. Its internal firmware includes a BASIC-style interpreter that makes writing and debugging programs a snap. Its manufacturer has free downloadable software that runs on a PC for program development. The instruction set includes some very powerful commands that work flawlessly. In short, it's a dandy part and a joy to use.

Another nice part is the AD9835 from Analog Devices. This is a complete DDS chip that runs from a 50-MHz clock and generates synthesized sine waves with 32-bit resolution. This chip directly generates the VFO signal with only a simple band-pass filter at its output. I have found that when operated within the narrow range of the 17-meter band and with the band-pass filter, the VFO output is virtually free of any discernible birdies.

The following parameters are measured by an ADC chip and displayed on the computer monitor:

1. AGC voltage from the receiver—converted by software into S-units.

2. Heat-sink temperature—turned out to be pretty much unnecessary. With the heat sink I use and a whisper fan continuously drawing air over it, its temperature never rises above 96° F.

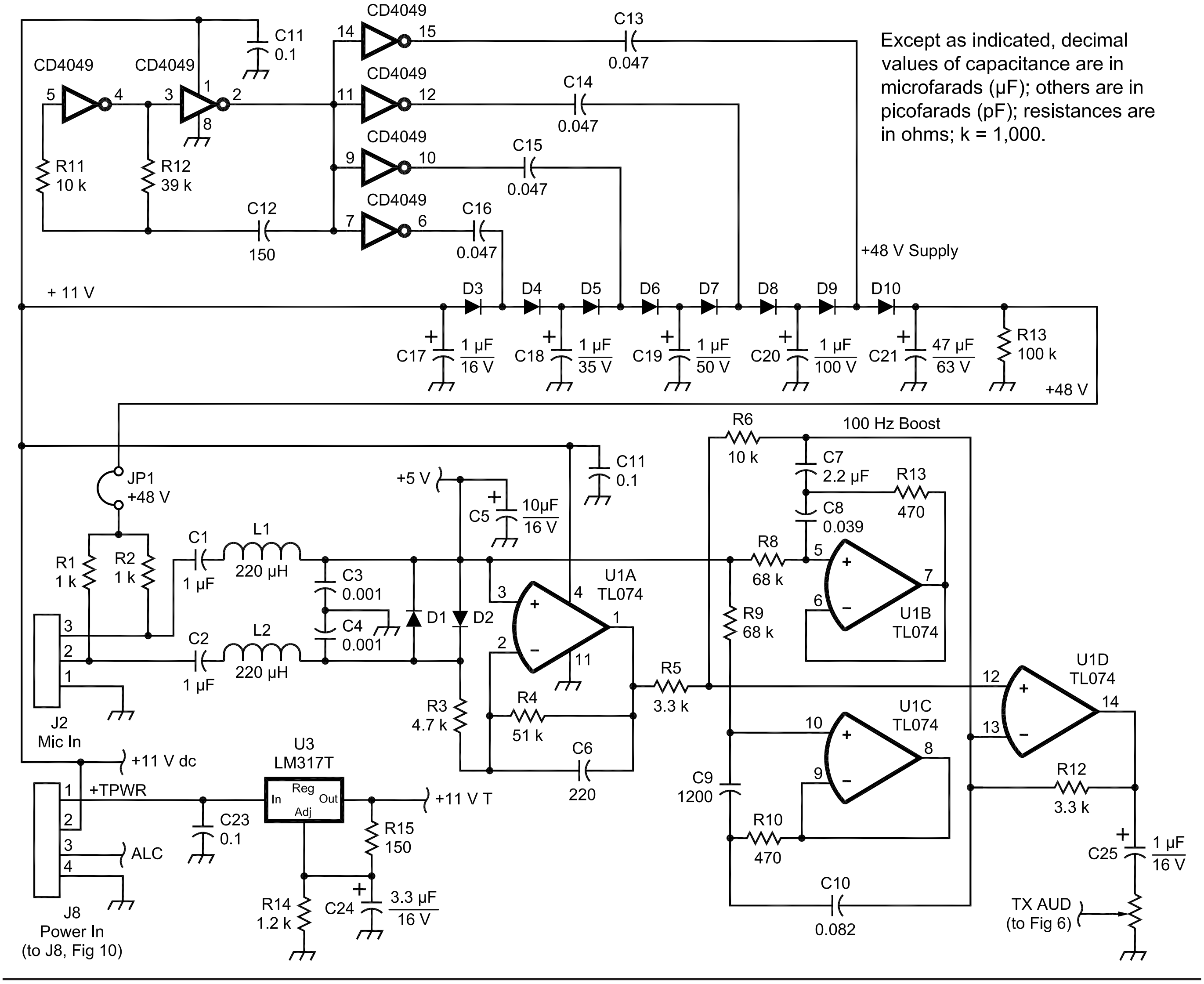
3. Forward and reflected power—these are run through peak detecting circuits before measurement.

Switched-Capacitor Filter

In spite of the sharpness of the crystal filter, frequent occurrences of QRM can be bothersome. To resolve this, an 8th-order elliptical low-pass switched-capacitor filter (MAX 293) in series with the audio path. The corner frequency is varied from about 2.0 to 3.0 kHz by controlling the clock frequency to the chip, which runs at 100 times the corner frequency. The clock is generated by a VCO whose frequency is varied with a digital pot.

The MAX 293 contains an uncommitted op amp, connected to provide a low-frequency boost to the audio. The rise is 15 dB at 50 Hz. I find this adds a nice mellowness to the receive audio.

The filter's cutoff does a good job of cutting QRM. I've found it to be nearly



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

Fig 5—Partial schematic of transmitter board showing the mic amplifier, equalizer and mic gain circuits.

as effective as the DSP filter on my Ten-Tec Pegasus.

Digital Volume Control

Since I was determined to avoid any physical controls in favor of virtual ones, it was necessary to use a digital pot to control the receive audio level. The LM1971 varies the volume from 0 to –62 dB in 1-dB steps. It has an audio-taper characteristic, and it is controlled by the Stamp II.

Transmit/Receive Power Control

When going between transmit and receive, the 13.8-V dc input power is switched by a relay. Transistor switches would operate faster; but since QSK operation isn't in my specifications, I chose the relay for simplicity and reliability.

Receiver Circuit Description

Refer to Fig 4. The preselector, consisting of L1 - L3 and associated capacitors, is a three-resonator bandpass filter. Typically, preselection is done with a simple stagger-tuned two-resonator stage; but in my case, there is a strong VOA station that operates near me just below the 17-m band. I wanted the extra rejection the triple stage provides. The RF amplifier makes up for the filter insertion loss and provides about 6 dB of additional gain. Mixer U2 receives the RF amplifier output signal and low-side injection from the LO (carrier frequency *minus* the IF). Regulator U9 provides +5 V for U2 and also noise rejection from the +11-V supply. U2's output impedance is 1.5 kΩ and R2 is connected in shunt to provide the 750-Ω source resistance required by FL1.

IF, AGC Amplifiers and Product Detector

The IF amplifier is the MC1350. This common, easily obtained part provides about 55 dB of gain [this part is still available through ECG—*Ed*.]. At first glance, this may not seem like enough IF gain, but the receiver sensitivity requirement is met in large part because of the conversion gain provided by the NE602s in the receiver-mixer and product-detector stages. In fact, the receiver

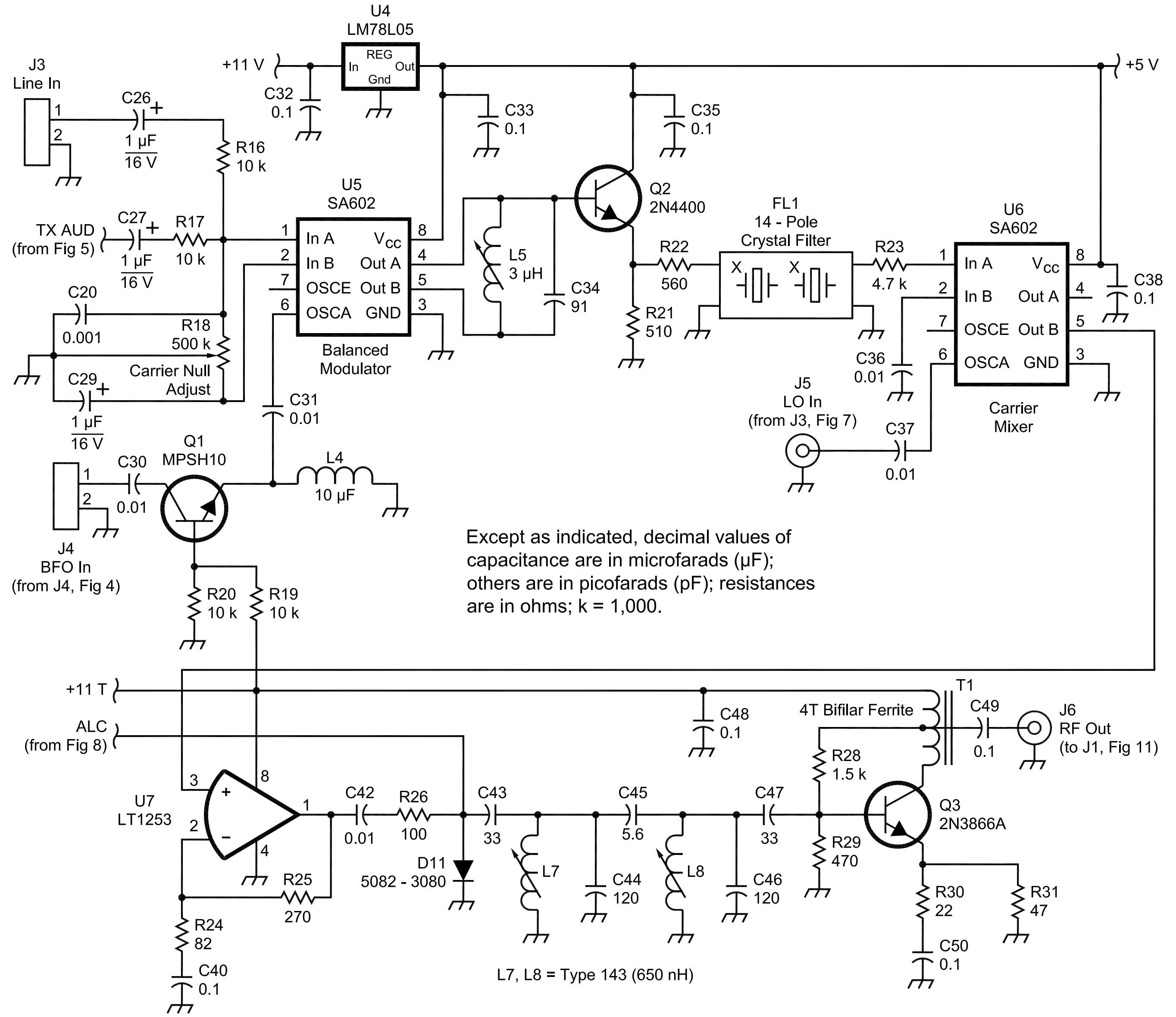


Fig 6—Partial schematic of transmitter board showing the exciter circuit.

gain overall is spread among the RF amplifier, receive mixer, IF amplifier and product-detector stages. This helps to ensure that no one stage becomes overloaded in the presence of strong input signals.

The IF output goes to the product detector, U4, and to U5, where it is heterodyned down to 150 kHz with adjustment of L8. The 150-kHz signal envelope appearing at U5A pin 1 is amplified by U6A, peak detected, buffered by U6B and applied to the AGC-control pin on the IF amplifier, U3.

BFO

The BFO is the oscillator section of the product detector and employs an 8-MHz crystal, Y1, that is "pulled" to 7.9965 MHz, a frequency that corresponds to that about halfway down the lower skirt of the crystal filter. Placing the BFO frequency that close to the filter edge allows most of the IF energy from the lower audio frequencies to get through the filter. This helps make for fuller, mellow-sounding audio for both receive and transmit since the same BFO is shared by both.

Notice that the product detector and AGC circuit operate from uninterrupted +11 V dc. This is because the BFO is required in both receive and transmit modes. In the case of the AGC, if its power were interrupted, the long time constant in the peak detector would introduce a significant delay in receive audio when switching from transmit to receive.

Exciter Circuit Description

The microphone 48-V supply is shown in Fig 5. This charge-pump circuit runs at about 70 kHz. Also shown are the microphone amplifier, equalization amplifiers and a three-terminal regulator U3 that generates the +11-V supply voltage for the exciter board.

The +48 V is fed to the mic when jumper JP1 is installed—the jumper is left out when using a dynamic mic. Any RF on the mic cable is suppressed by L1, L2, C3 and C4. Diodes D1 and D2 prevent damage to U1 in the event a microphone is plugged in while the phantom voltage is present. (This would cause large negative spikes to be coupled through C1 and C2.) The U1A output feeds two boost circuits in parallel. U1B provides a 16-dB boost at 100 Hz and U1C, a 16-dB boost at 2.7 kHz. This may seem like a lot of boost; but for my voice at least, it produces a nearly flat audio spectrum from about 80 Hz to 3.0 kHz on a receiver set for 3-kHz audio. There is a slight low-frequency hump, but most listeners think it sounds better with the hump.

The mic-gain pot (R15) is mounted on the exciter board inside the rig and is not accessible from the outside. The pot is adjusted once for normal speech with the intended mic and then let alone. This was purposely done to eliminate the possibility of running the mic gain too hot and causing distortion from balanced-modulator overload. Remember that we're doing everything we can to make this rig sound good.

Referring to Fig 6, the audio signal is fed to the balanced modulator, U5. Q1 operates as a back-to-back diode switch that turns on the BFO signal to U5 when in transmit mode. It's necessary to operate the balanced modulator continuously because the large audio-coupling capacitors would take too long to charge if power were interrupted. The coupling capacitors were made intentionally large to extend response in the lower audio spectrum. R18 is adjusted for minimum carrier output with no audio.

The DSB signal output from U5 is buffered by Q2 and fed to the crystal filter. Notice that this filter is identical to, but separate from, the one on the receiver board. In other designs, the crystal filter is usually shared between the receiver and exciter. In this case, it is considered too cumbersome to do so because the receiver and exciter boards are separate. Also, the filters are homemade from $0.49 crystals; they are cheap compared to commercial ones.

The crystal-filter output is fed to

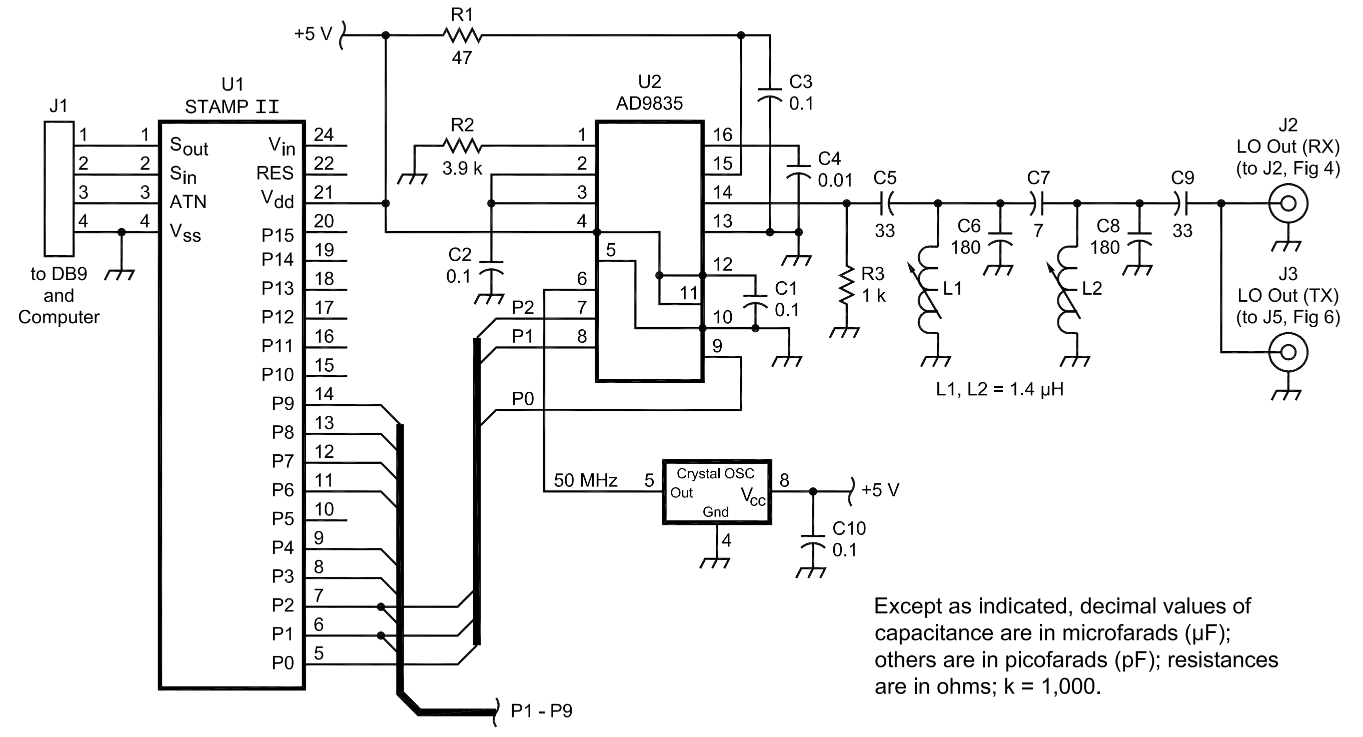


Fig 7—Partial schematic of controller board showing the microprocessor, DDS local oscillator and LO filter.

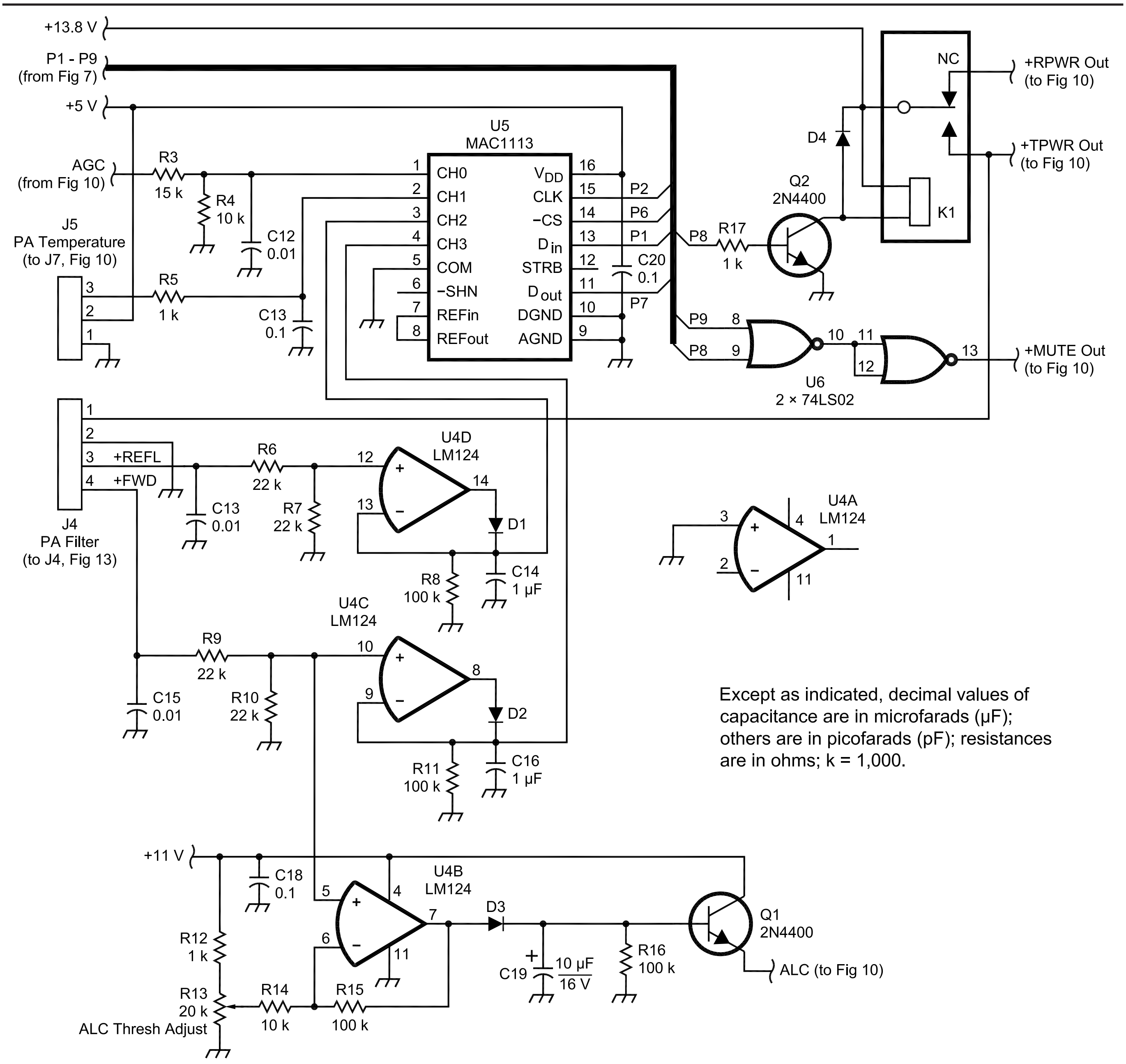
the carrier mixer (U6) through dropping resistor R23. This resistor is necessary to prevent overloading U6—remember it has conversion gain. Low-side injection from the LO is again employed to produce the sum and difference frequencies: The sum is the desired carrier frequency. The U6 output is amplified 12 dB by U7 to drive the ALC limiter consisting of R26, R27 and PIN diode D11. The signal then passes through a simple band-pass filter tuned to 18.1 MHz to remove the difference frequencies before it's further amplified 20 dB by Q3. The 50-Ω output from Q3 is fed to the driver/power amplifier via J6.

Controller Circuit Description

The Stamp II module shown in Fig 7 is the heart of the controller board. It connects to the PC via the 9600-baud serial port. Its 16 I/O pins can be programmed as either inputs or outputs.

Pins P0, P1 and P2 connect to the VFO chip U2. The VFO contains various registers, including four eight-bit holding registers that accept the 32-bit frequency word. Those four registers must be reloaded each time the carrier frequency is changed. The sequence is as follows:

1. Operator changes the frequency at the PC.
2. PC software issues a frequency change command to the Stamp II.
3. Stamp II lowers P0 to signal the VFO that a new frequency word is to follow.
4. Stamp II generates an eight-bit control word followed by the first eight data bits of the 32-bit word and shifts these 16 bits out on P1 along with 16 clocks on P2. The control word tells the VFO which of the four holding registers is to be loaded.
5. Stamp II repeats step 4 until all 32 frequency bits are loaded.
6. Stamp II issues a 16-bit write command word to enable the new frequency word.



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

Fig 8—Partial schematic of controller board showing the ADC, mute and power relay.

Thus, a total of 64 + 16 = 80 bits are required to change the VFO frequency.

The VFO is clocked from a 50-MHz crystal-oscillator module. On the PC board, I provided for a positive-temperature-coefficient (PTC) thermistor to be attached to the module case to keep it nearly constant at 95°F. I've found that frequency drift with that particular module is negligible, so the thermistor is unnecessary.

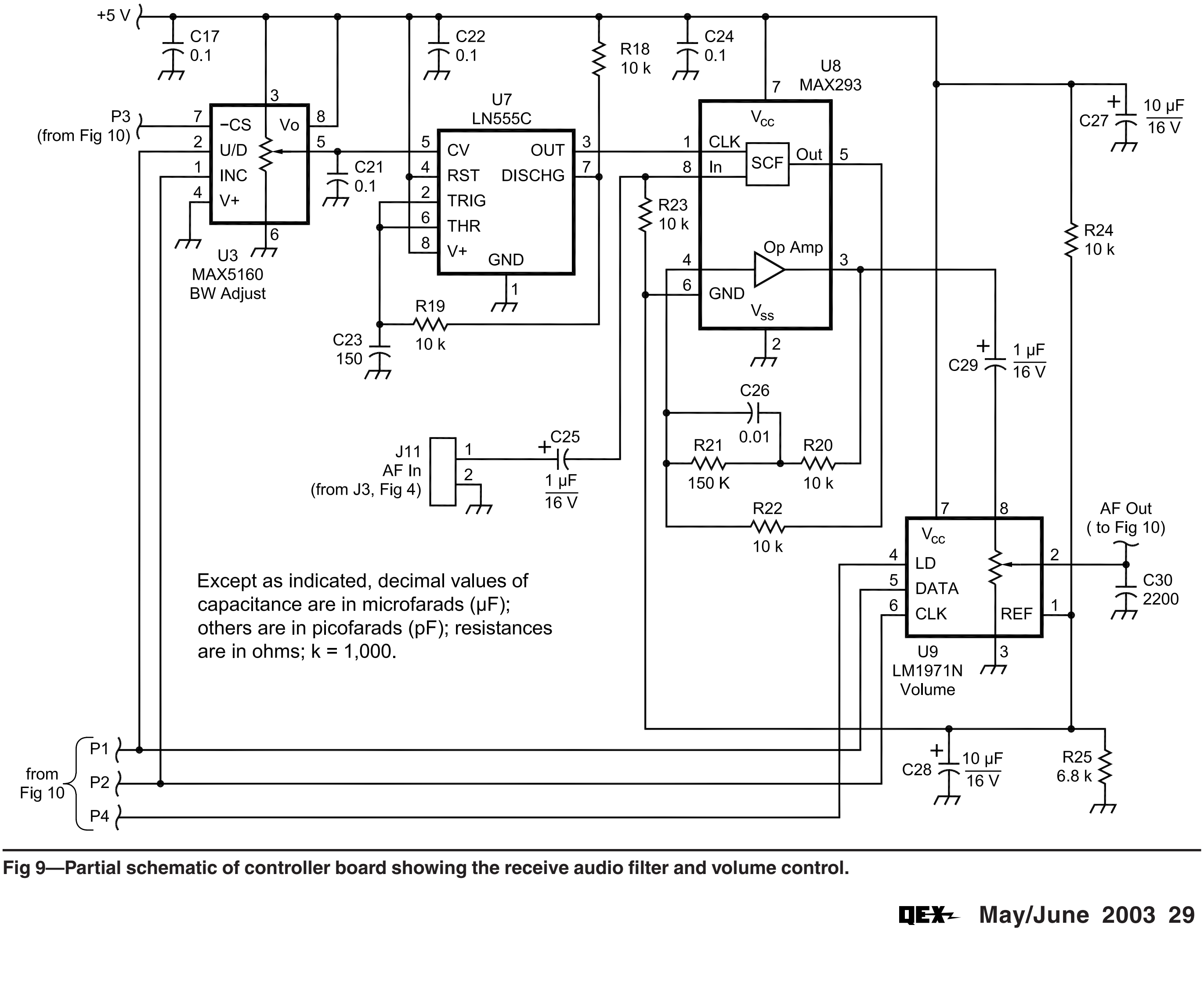
With 32-bit resolution and running at 50 MHz, the VFO can be stepped in 0.01164 Hz increments. This is much greater resolution than is required. In my software, the minimum step value is 10 Hz. Load resistor R3 sets the level to about 700 mV P-P at the bandpass filter output. This is more than sufficient to directly drive both receive and transmit mixers.

Fig 8 shows the analog measuring circuits. The AGC voltage, which varies with receive signal strength, is divided down by R3/R4 to fit within the 5-V measurement range of the ADC chip U5. Heat-sink temperature is measured by an LM34 sensor mounted on the heat sink and passes into an ADC input via the R5/C13 noise suppressor. Reflected power from the directional coupler on the PA Filter board is divided by R6/R7, peak-detected by U4D and associated circuitry, and measured by the ADC. Forward power is handled the same way and input to the ALC-control stage U4B and associated components. When forward power exceeds the level set by ALC Threshold pot R13, the voltage at the emitter of Q1 rises and causes D11 in Fig 6 to conduct, thereby limiting exciter output. The limiting action is soft, eliminating any distortion that might be caused by hard ALC action. The +13.8 V to the transceiver is switched to either the receive or exciter boards through relay K1 controlled by signal P8 from the Stamp II.

U6C and U6D are gates for muting the receive audio. Muting occurs while transmitting and when P9 is set high by the operator pressing the mute key (F1).

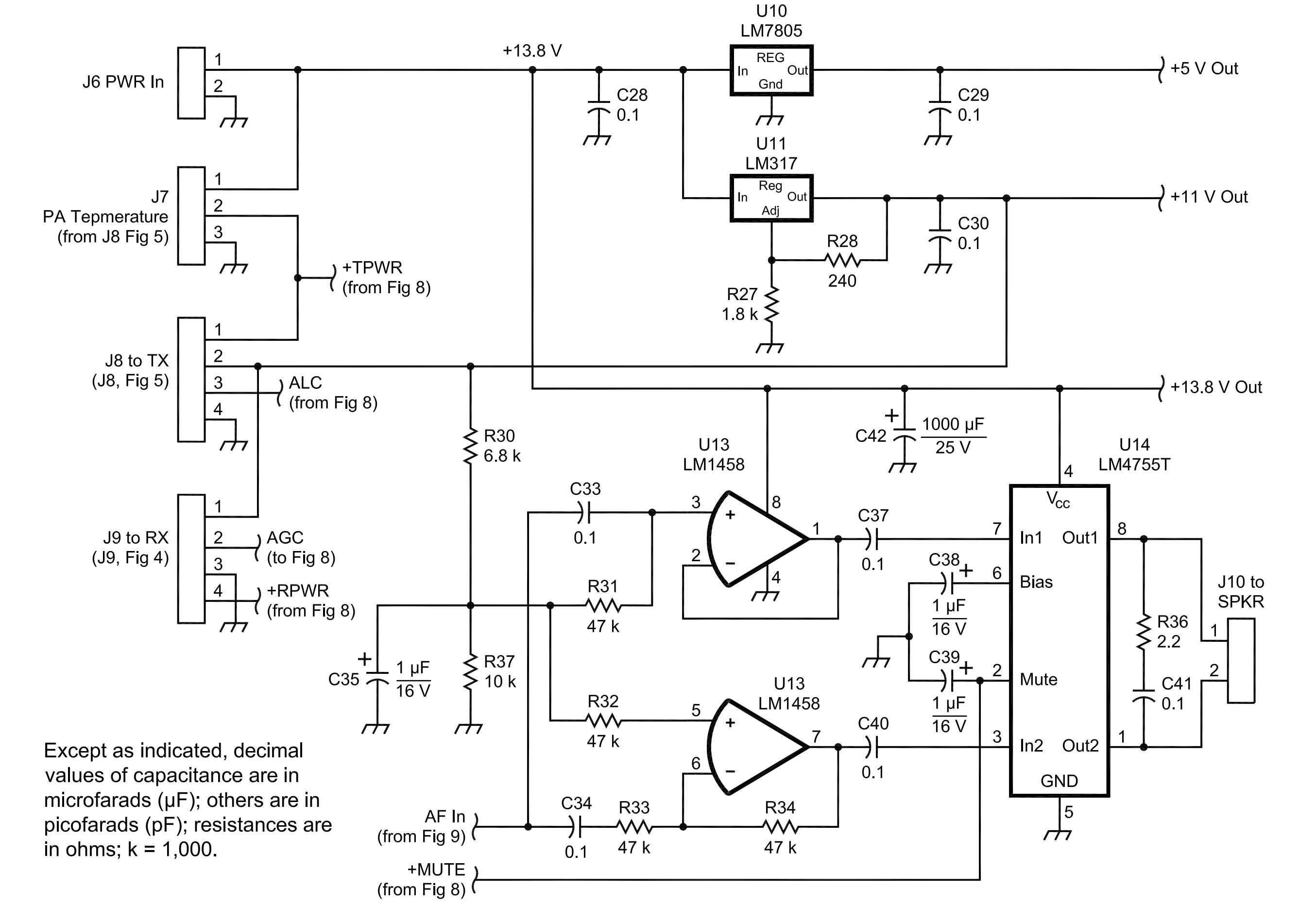
Fig 9 shows the switched-capacitor circuits. Digital pot U3 adjusts the control voltage to U7, which is connected as a multivibrator that produces pulses in the range from 200-300 kHz. These pulses clock filter U8 at 100 times the desired audio cutoff frequency (2.0-3.0 kHz). The audio (J11 Fig 9) signal from the product detector passes through the SCF low-pass filter and then to an uncommitted op amp within U8. This op amp is connected to produce a low-frequency boost beginning at about 700 Hz and rising to about 18 dB as frequency decreases to 50 Hz. Again, this might seem like excessive boost, but it really warms the receive audio. The output signal is fed to a digital pot U9 that changes the receiver's audio level in 1-dB increments under software control.

Referring to Fig 10, the pot output feeds two unity-gain amplifier stages U13: one inverting and one noninverting. These, in turn, feed the inputs



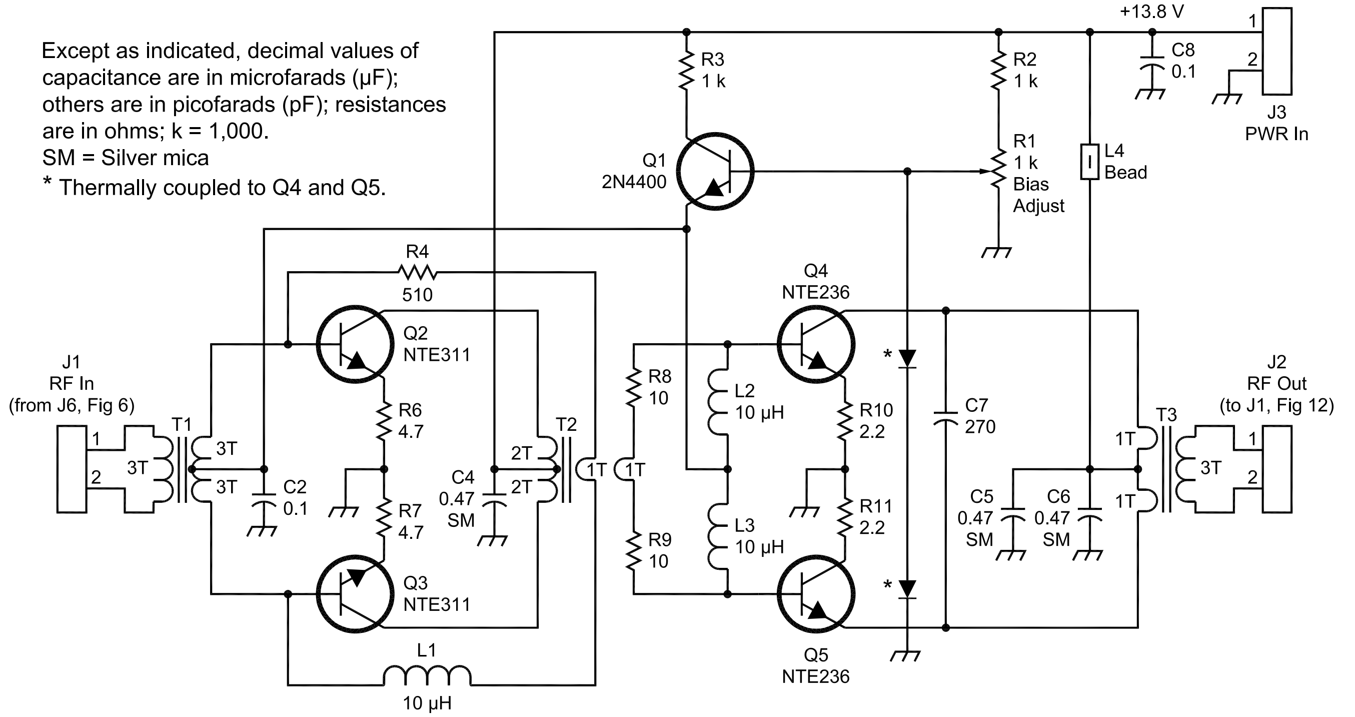
Except as indicated, decimal values of capacitance are in microfarads (µF); others are in picofarads (pF); resistances are in ohms; k = 1,000.

Fig 9—Partial schematic of controller board showing the receive audio filter and volume control.



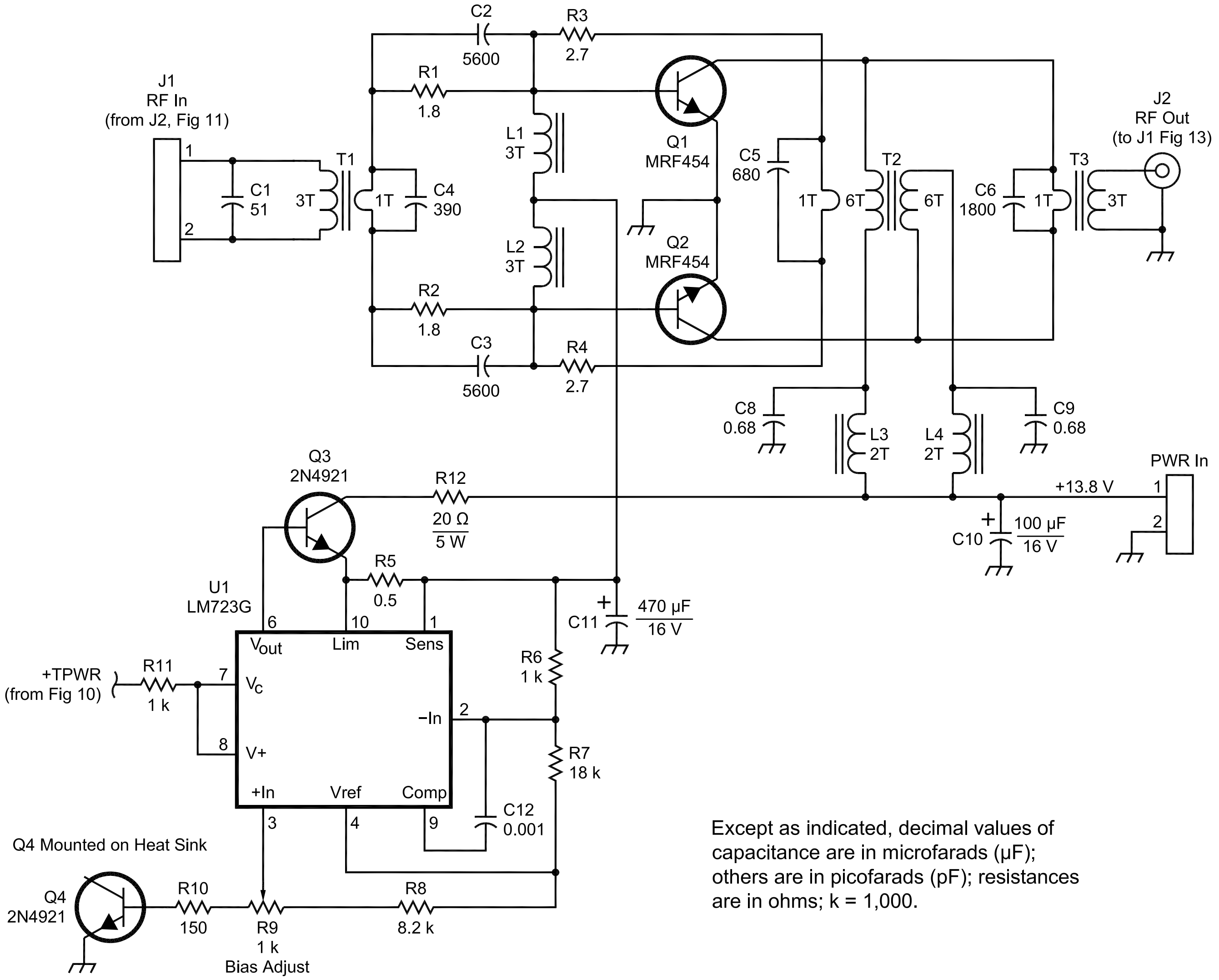
Except as indicated, decimal values of capacitance are in microfarads (µF); others are in picofarads (pF); resistances are in ohms; k = 1,000.

Fig 10—Partial schematic of controller board showing the voltage regulators and speaker amplifer.



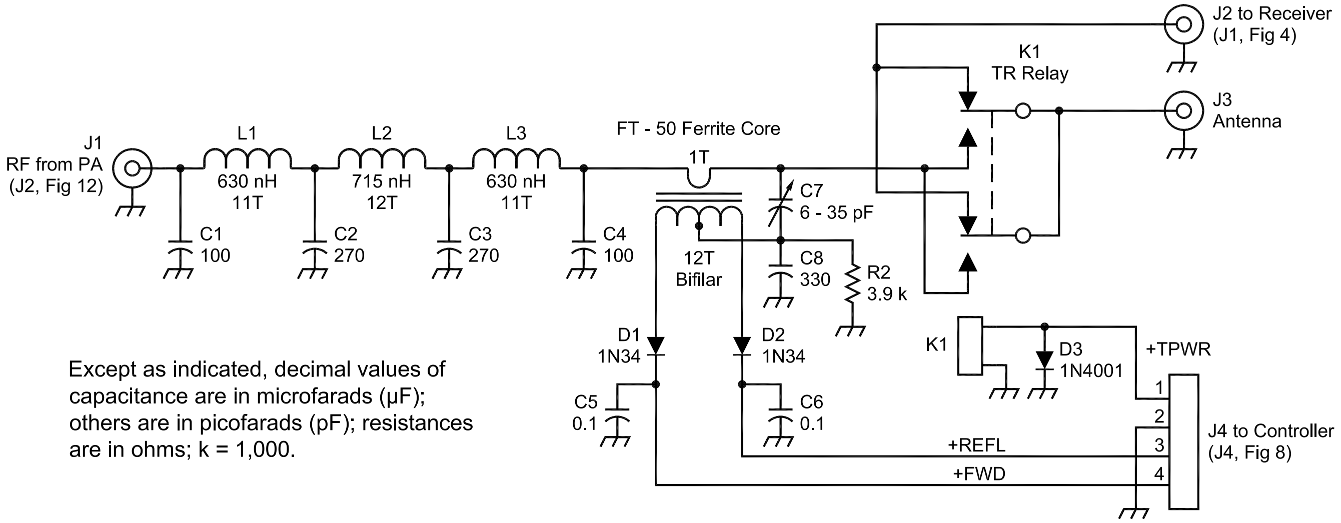
Except as indicated, decimal values of capacitance are in microfarads (µF); others are in picofarads (pF); resistances are in ohms; k = 1,000.
SM = Silver mica
* Thermally coupled to Q4 and Q5.

Fig 11—Schematic of the RF driver circuit. T1 and T2 are 0.525×0.575-inch ferrite binocular balun cores of #43 material. T3 is a 0.56×1.10-inch ferrite binocular balun of #43 material.



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

Fig 12—Schematic of the RF power amplifier. T1-T3 are Stackpole parts: T1 is a 57-1845-24B core; T2 is a 57-9322 core; T3 is two 57-3238 ferrite sleeves. The MRF454 is now the Microsemi MS1001 from APT.



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

Fig 13—Schematic of PA filter, directional coupler and TR switching relay.

Designing 14-Pole Crystal Ladder Filters

Crystal band-pass filters with extremely steep skirts and almost ruler-flat passband characteristics can be easily built using inexpensive microprocessor crystals. Nevertheless, it takes some careful measurements and some nifty software to do it. Thanks to a program recently developed by Wes Hayward, W7ZOI, the task becomes nearly trivial.

Just any off-the-shelf crystal won't work well, however. Crystals with series-resonant frequencies in the range of 6-12 MHz work best. You need many of them, out of which you sort the "good" ones. All crystals must first be measured to obtain their series-resonant frequency, equivalent series capacitance (sometimes called *motional* capacitance), equivalent series resistance (ESR) and parallel or holder capacitance. This all sounds horrendous, but a very simple measurement jig, along with an accurate signal generator and a 'scope, does the whole job.

Using the G3JIR article in *QEX* (Jan/Feb 2002, pp 7-11) as a guide, I constructed the test jig shown in Fig A. The measurement procedure goes as follows:

1. With the dual-channel scope connected to TP1 and TP2 and with S1 closed, adjust the signal generator until the two waveforms are exactly in phase. This is the series-resonant frequency; record it as F_s. (I set up a spreadsheet program for this data and making the necessary calculations).

2. The measurement is now repeated with S1 open and the new frequency recorded as F_{cap}. The equivalent series capacitance is:

$$C_s = \frac{2C1\left(F_{cap} - F_s\right)}{F_s} \qquad \text{(Eq A)}$$

where *C1* is the 100-pF capacitor in shunt with S1 and any stray capacitance. It's important to measure this total capacitance with a good LCR meter!

3. Next, connect the scope to TP3, close S1, set S2 to position 1 and adjust the signal generator for maximum signal output. Record this signal level. Then switch S2 to position 2 and adjust the 50-Ω pot to obtain the same signal level. Now measure the pot resistance. This is the ESR of the crystal; record it as R_s. The crystal *Q* is:

$$Q = \frac{1}{\omega C_s R_s} \qquad \text{(Eq B)}$$

where $\omega = 2\pi F_s$

Record the value of *Q*.

4. Calculate the motional inductance:

$$L_m = \frac{1}{\omega^2 C_s} \qquad \text{(Eq C)}$$

and record it.

Sorting Out the "Good" Ones

I bought a bag of 8-MHz crystals from Jameco for $0.49 each and measured them all. Then I screened out about 20 with inferior *Q*s (less than 60,000). The average *Q* of the good ones was about 110,000. For best filter sharpness and lowest insertion loss, you want to use the higher *Q* crystals.

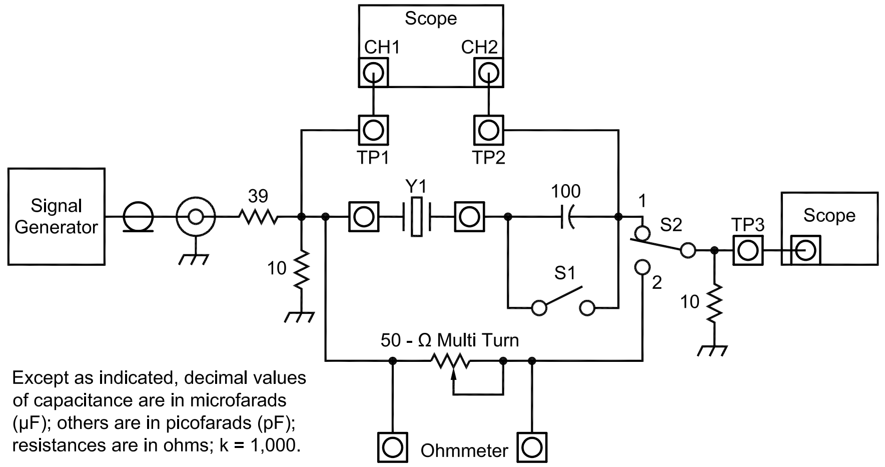
Using the W7ZOI Software

XLAD is a program for designing crystal ladder filters. My filters were designed using early beta-test versions of the software. Since then, Wes Hayward has done further work on the program. For this reason, I won't describe the program in detail, but entering the crystal data was a snap and the program did the rest. The program requires the following parameters:

1. Nominal crystal frequency (8 MHz)
2. L_m (for both L_m and *Q*, the values of the chosen crystals were averaged)
3. *Q*
4. Holder capacitance (This was measured using an LCR meter connected across the crystal terminals. All crystals measured the same, 5.6 pF).
5. Desired bandwidth (3.0 kHz)
6. Number of crystals (14)
7. Chebyshev or Butterworth response (Chebyshev with 0.2-dB passband ripple)
8. End resistance (This is the resistance of both the filter source and its termination. I chose 750 Ω to yield reasonable values for the end capacitors. I would have preferred 1.5-kΩ end resistance to permit direct connection to the NE602, but the resultant end-capacitor values were impracticably small.)

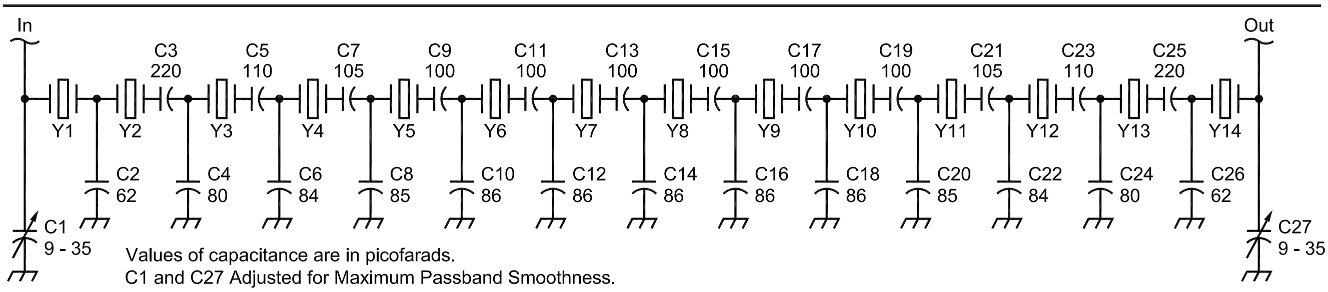
Once these parameters are entered, the program instantaneously calculates and displays the values for both the shunt coupling capacitors and the series-tuning capacitors.

There is also a separate program for fine-tuning the filter. F_s varies from crystal to crystal and the specific F_s for



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

(A)



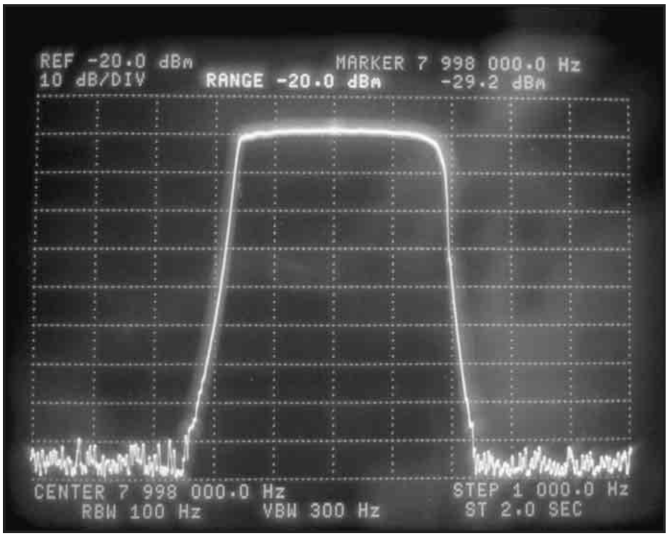
(B)

each one used in the design can be plugged into this program. The program then adjusts the series tuning capacitor for that crystal to accommodate the frequency offset. I tried using this program, but the results didn't change the final filter performance much, so I didn't pursue it any further.

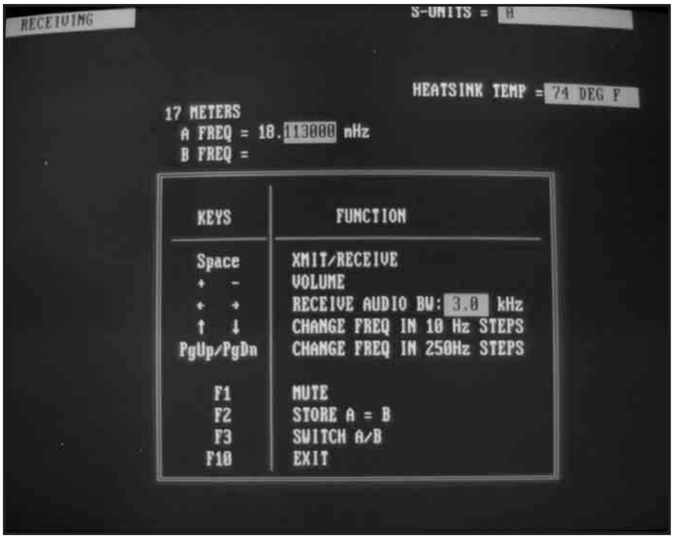
Building the Filter

After choosing the batch of 14 crystals I averaged their series-resonant frequencies and calculated each offset frequency from the average. I then plugged each crystal into the filter in the following manner. Notice from the schematic in Fig B that the filter is symmetrical about its center. I put the crystals with the lowest offset in the center and arranged the others with progressively larger offsets toward both ends. Thus, Y1 and Y14 have the largest offsets. This seemed like "the right thing to do." However, I can't prove that it made the filter perform any better.

The measured response of one filter is shown in Fig C. Notice there is no undesirable "blow-by" coupling. This is so because with 14 crystals, the end-to-end distance on the PC board is wide enough to preclude it.—Rod Brink, KQ6F



(C)



RECEIVING

S-UNITS = 8

HEATSINK TEMP = 74 DEG F

17 METERS
A FREQ = 18.113000 mHz
B FREQ =

KEYS	FUNCTION
Space	XMIT/RECEIVE
+ -	VOLUME
← →	RECEIVE AUDIO BW: 3.0 kHz
↑ ↓	CHANGE FREQ IN 10 Hz STEPS
PgUp/PgDn	CHANGE FREQ IN 250Hz STEPS
F1	MUTE
F2	STORE A = B
F3	SWITCH A/B
F10	EXIT

Fig 14—The operator-interface screen.

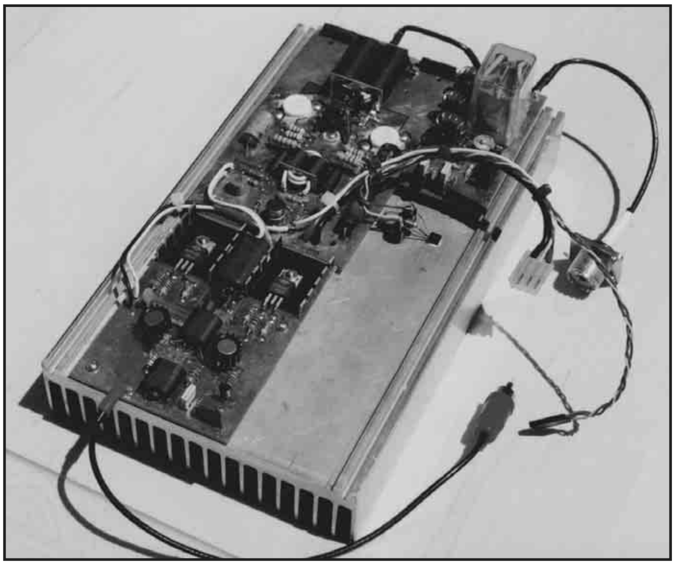


Fig 15—Heatsink with RF driver, PA, PA filter and TR relay. Notice the temperature sensor below the PA filter board.

of a stereo power amplifier U14, connected in a bridge configuration. I use an 8-Ω hi-fi speaker and found the extra power afforded by this bridge configuration is necessary to get comfortable listening volumes without distortion. Also shown in Fig 10 are two three-terminal regulators for developing +5 V and +11 V to power the radio.

RF Driver and PA Stages

The RF Driver is shown in Fig 11. It consists of two cascaded push-pull stages and provides about 23-dB gain.

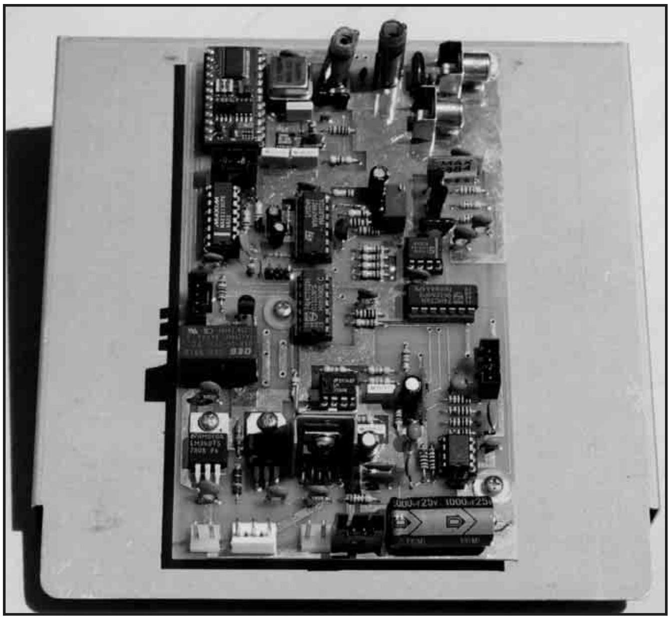


Fig 16—Control board mounted on shelf.

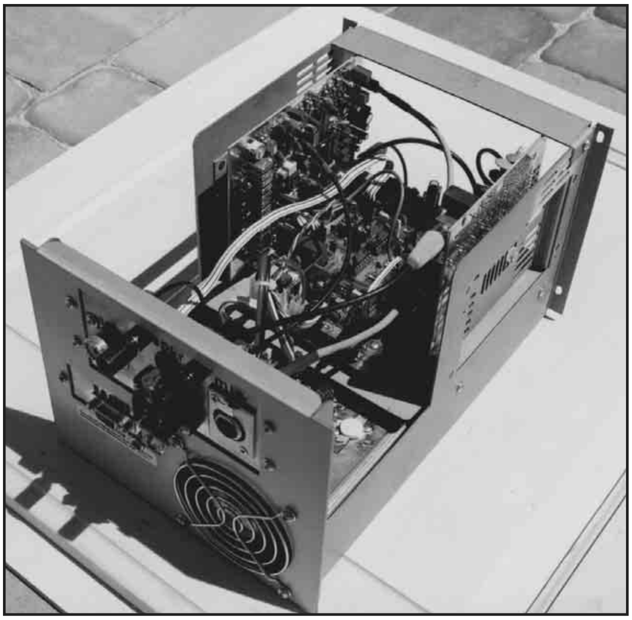


Fig 17—Rear view with covers off.

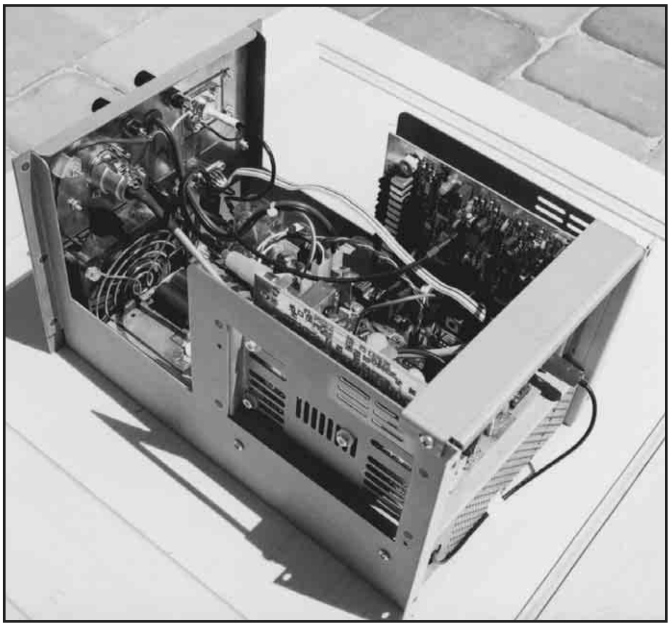


Fig 18—Top/front view with covers off.

Fig 19—Complete transceiver with covers and faceplate.

Emitter feedback is employed in both stages to help stabilize dc biasing. This feedback also acts to broaden the frequency response. R4 and L1 further broaden the response. Actually, I didn't work very hard to achieve broadband response, since operation is over a very narrow frequency range. Nevertheless, a certain amount of broadband response might be necessary in some future application.

Bias compensation is provided by diodes D1 and D2 that are temperature coupled to the Q4 and Q5 heat sinks. Quiescent bias levels are determined by pot R1 and are set to 15 mA for Q2, Q3 and 60 mA for Q4, Q5.

The power amplifier, shown in Fig 12, develops 100 W into a 50-Ω load. It's a straightforward push-pull design using MRF454 transistors mounted to a somewhat oversize heat sink. Transformer-coupled feedback is employed to achieve a broadband response, although again, that is not required. Transistor Q4 is mounted on the heat sink to provide bias compensation. Regulator U9 makes the bias very stable. R9 is adjusted for about 0.5 A of total collector current. I found this level necessary for good IMD performance.

PA Filter, Directional Coupler, TR Relay

Refer to Fig 13. I designed these onto a small PC board mounted alongside the PA. In attaching the board to the heat sink, I used nylon standoffs to insulate it from the heat sink. The heat sink carries circulating currents from the PA emitters, and I don't want those currents, which are rich in harmonics, to get into the filter board.

The output filter is a three-stage Butterworth design with a 21.5-MHz corner frequency. The response at any second harmonic within the 17-meter band is better than –48 dBc. The directional coupler is straightforward. C7 is adjusted for a null when operating into a 50-Ω resistive load.

Software

The firmware that runs in the Stamp II employs a *BASIC* instruction set and was developed using the PC-based editor supplied by Parallax, the Stamp II manufacturer. There are about 115 lines of code, including the variable-statement field.

The PC software is about 250 lines of *QBasic*. The operator screen is shown in Fig 14. At this writing, the software is being rewritten in Borland's *Delphi* to give it a graphical interface.

Packaging the Transceiver

This is where I really lucked out. In a local surplus warehouse, aptly named "Weird Stuff," I found a lovely little chassis that once held a couple of external hard drives. There was a small switching power supply in the bottom and a fan on the rear panel. I saw that my power-amplifier/heat-sink assembly would fit nicely in place of the power supply and directly in front of the fan. A built-in shelf would later hold my control board, and there is plenty of space along the sides for mounting the receiver and exciter boards. Figs 15 through 19 show the whole story. What a perfect fit! The cost was $10.

Summary

Homebrew lives! Your comments are welcome.

Rod was first licensed in the early 1950s, but got busy with school and forgot all about ham radio. After retiring in the 1995, he re-entered Amateur Radio as KE6RSF, then AD6TV and now KQ6F. He holds BS(EE) and MS(EE) degrees and worked 37 years for various electronic companies in the Silicon Valley, in charge of their engineering departments for the last 20 years of his work life. He now spends much of his leisure time designing and building computer-operated radios.

Linrad: *New Possibilities for Communications Experimenters, Part 3*

Linux *and the* Linrad *software package.*

By Leif Åsbrink, SM5BSZ

Linrad is a computer program that runs on PC computers under the *Linux* operating system. I have chosen to write this software under *Linux* because under *Linux*, all development tools are free. I want to encourage others to play with the code and make additions and modifications. The disadvantage is that one has to install *Linux*, which is seen as a major obstacle by many potential users. Installing *Linux* is quite simple on a reasonably modern standard PC. It is possible to run *Linrad* on a fast 486 computer, but it is not trivial to install *Linux* on such old machines. The easy way to install *Linux* is to use a computer that can be booted from a CD. Just insert a *Linux* distribution CD, boot from it and follow the instructions—not really difficult, but if you have never done it before, it helps a lot to have someone more experienced at your side.

Jaders Prastgard 3265
63505 Eskilstuna, Sweden
leif.asbrink@mbox300.swipnet.se

To run *Linrad*, you must install *svgalib*, a package that contains drive routines for video boards. Unfortunately, *svgalib* does not support all board types; if the video board is integrated on your motherboard, there is substantial risk that *svgalib* will not support it. The sound system under *Linux* is not quite stable yet. Some *Linux* distributions have sound that works automatically, but most need the installation of a new sound package. I am using OSS, which is not free; but the free ALSA sound system should be compatible, although I have not been able to make it run myself.

Your sound package must support the sound card in your computer, which is often a problem with sound cards integrated on the motherboard. I am not an expert on hardware compatibility issues. Everything I have tried has worked automatically but I have had several reports about problems with integrated video or sound. This article does not further concern itself with *Linux* issues that are not specific to *Linrad*.

Basic Functions of *Linrad*

Linrad displays a portion of the RF spectrum on the computer screen. The bandwidth (kilohertz) displayed is determined by your hardware. The operator can see all signals within the on-screen passband and can tune to any particular station by clicking it with the mouse. The signal is then processed and presented to the operator via the headphones or as text on the screen. *Linrad* is planned to support the following modes:

- CW (including weak-signal and meteor-scatter)
- SSB
- FM
- Various digital modes

Linrad is still at a reasonably early development stage. Only the weak-signal CW routines are implemented and the only output is sound to the headphones. The weak-signal CW algorithms are currently used for the other modes, but they are not necessarily optimal. When the user has pressed the button, *Linrad* looks back in time and decides what the optimum center frequency is and how it changes with time. An internal local oscillator is set up and used to convert the frequency of the incoming spectrum for the desired signal to be centered in the final passband selected by the user. This AFC function makes it possible to use narrower filters than otherwise possible on weak and unstable CW signals. The output of the final filter can be used in several ways. It can be

- Routed directly to the headphones
- Coherently processed (only the component of the signal that is in phase with the carrier is sent to the loudspeaker)
- Filtered through a narrower filter for one ear or delayed.

All tricks I know of to enhance the reception of weak signals are there.

For normal CW or SSB, one can disable the AFC, set the bandwidth to 2 kHz and use *Linrad* as an ordinary receiver. There is no AGC, however, since an AGC is inappropriate for weak-signal CW. When the algorithms for other modes are in place, they will have AGC and some interference-fighting functions.

Linrad can combine signals from two antennas. The spectrum presented on the screen will then contain all the signals from both antennas and when the mouse is clicked, the two signals are combined to maximize the desired signal. This is an obvious strategy for weak-signal CW, where the two antennas are typically the two polarizations of a crossed Yagi array. The effect of combining the signals is identical to that obtained when the crossed Yagi array is aligned with the polarization of the incoming wave. The user is then presented with the best possible signal. This function is very useful for EME. It eliminates loss of signal caused by Faraday rotation. For other modes, it may be better to combine the two signals for minimum interference, which can be done by hand (but which will be automated in the future).

Combining two antennas with complete freedom in phases and amplitudes means electronic lobe control. Someday, when the antenna signals can be digitized more or less directly at low cost, many more channels will be extremely useful on crowded HF bands, but *Linrad* has no provisions for more than two channels yet.

Linrad contains a very special noise blanker, such that *Linrad* can be used to hear signals that are impossible to receive with any other radio. Power-line noise may have repetition rates of several kilohertz. The interference source is a spark gap in series with a capacitor, that is, a defective insulator in series with some good ones. When the mains frequency is applied at a high voltage, pulse trains are emitted at twice the mains frequency. Each pulse train typically consists of 10 pulses. A noise blanker must resolve the individual pulses well and therefore the noise blanker must work with a bandwidth of 10 kHz or more. Many ham transceivers have such blankers and they work fine until some strong undesired signal is present within the blanker bandwidth. The *Linrad* blanker has automatic notch filters that remove all strong signals in the blanker passband. The *Linrad* blanker finds out as much as it can from both of the receive channels and subtracts the most probable interference waveform from the incoming data. That means that much less information is lost because only a few data points at the peak of the pulse must be blanked. The pulse tails are accurately reduced to well below the noise floor by the subtraction process.

The practical implementation of *Linrad* is based on Fourier transforms. The description above is what the algorithms do, not the way they are implemented. The Fourier transforms are needed anyway for various reasons and it saves a lot of CPU time to use them for the actual processing.

Linrad has no prejudice about how you want to use it or to the hardware to which it is connected. The user must select parameters that make a good radio receiver from the building blocks. If you ask for a 20-Hz filter that rejects signals well only 0.1 Hz outside the passband, the processing delay will be on the order of 10 seconds for the filter. If a simple sine window is chosen to save CPU time and large decimation rates are selected, aliasing spurs will degrade the dynamic range, and so on. There is nothing wrong with any of this, and it has nothing to do with *Linrad* being a DSP system. The propagation delay through a filter is related to the *Q* regardless of whether an analog or a digital implementation is chosen. There may be perfectly valid reasons to design a radio either way, and *Linrad* allows you to do it. I have tried to make the control functions so that it is intuitive to set up a good, normal radio receiver.

Linrad Block Diagram

The block diagram of Fig 1 illustrates signal flow through the different processing steps of *Linrad*. Blocks labeled *fft* and a number go from the time domain to the frequency domain via a fast Fourier transform routine. Blocks labeled *timf* and a number go from the frequency domain to the time domain by an inverse-fast-Fourier transform routine. Filtering and resampling are extremely easy in the frequency domain. One just removes irrelevant parts of the spectrum to get rid of undesired frequencies. The smaller transform size automatically reduces the sampling rate of the time-domain function that follows.

There are no conventional DSP filters inside *Linrad*. The filter shapes are not controlled by FIR filter coefficients. The actual processing is the same, however; the same sums of products are taken, but in another order. The window function used for the FFT becomes one component in the effective filter function that is used to prevent aliasing spurs in the resampling transformation.

The theory may seem frightening; but when actually operating *Linrad*, it is not difficult to understand what is going on. For example, you may try a very bad (and very fast) FFT with no window and few points. The resampling spurs that then surround any strong signal are easily seen. It is not difficult to figure out what causes them and how to remove them. It is exactly as in any other radio. If the frequency conversion reduces the frequency by a large factor, one has to put more effort into the filters to avoid spurs. The same thing happens when reducing the sampling speed. A large reduction rate requires a narrower filter because resampling spurs come closer.

The processing is controlled by certain parameters. These parameters fall into four categories:

- Hardware related parameters.
- Receive-mode related parameters.
- Operator's current preferences for a particular station.
- Dynamic parameters calculated by *Linrad* itself.

The hardware parameters are set for a particular hardware combination as a part of the *Linrad* installation.

The receive-mode related parameters enable or disable certain functions and control how much memory *Linrad* will allocate for various purposes. They also allow experimentation with transform sizes and other things one usually would not like to change when operating in a particular mode. What the parameters are and how to set them up is described in conjunction with detailed descriptions of the different processing blocks below. The operator's current preferences are given to *Linrad* by mouse clicks on the screen. Some parameters like the angle and phase that control how the two antenna signals are combined can be either category 3 or 4. A parameter of category 3 "Adapt" or "Fixed" is used to select that. Most of the real-time parameters are obvious. There is a help function; pressing the F1 key gives information about where the controls are and how to use them. The F1 key can also be used to get information about the mode related parameters in the mode-setup routine.

How to Install *Linrad*

First, you need a computer with *Linux* installed on it. You then must install *svgalib-1.4.3* or later if it is not already included in your *Linux* installation. You also must install *nasm*, an assembler that reads Intel-style assembly language exactly as written without making all sorts of unpredictable (to the innocent beginner) assumptions. *Linrad* contains fast routines that use the Intel multimedia instructions MMX and XMM and they are written in assembly language. Finally, you must install *Linrad* itself.

Linrad, *nasm* and *svgalib* are all available on the Internet. The *Linrad* home page is at **antennspecialisten.se/~sm5bsz/*Linux*dsp/*Linrad*.htm**, with mirrors at **www.g7rau.co.uk/sm5bsz/*Linux*dsp/*Linrad*.htm** and **nitehawk.com/sm5bsz/*Linux*dsp/*Linrad*.htm**. You can find links to *svgalib* and *nasm* there. At the *Linrad* home page, you will also find links with descriptions for the novice about how to install the *svgalib*, *nasm* and *Linrad* packages. *Linrad* comes only as source code, so you must compile and link it to obtain an executable program. Everything is automated. You need only issue two commands: **configure** followed by **make**. This is a normal procedure for the installation of a *Linux* package.

Before starting *Linrad*, you must get sound going. Installing OSS, the sound package from 4Front Technologies is always easy if your *Linux* distribution is modern enough to still be supported. Some distributions have sound included that works directly with standard sound cards. Sound under *Linux* is still a bit messy, but presumably, ALSA will be a free and well-working part of *Linux* in the near future if it is not already.

My information about *Linux* sound has a link from the *Linrad* home page, but my information is getting old. Since I have paid the OSS license, I just go on using OSS without much concern about what happens to *Linux* sound in general. There is a *Linrad* mailing list where users can interchange information. You will find a link to it at the *Linrad* home page.

Setting Computer-Related Parameters

When starting *Linrad* for the very first time, the initial screen in text mode may contain warning messages about multimedia instructions that you will never see again unless you remove the parameter file *par_userint*. Your hardware may support multimedia instructions while your *Linux* kernel does not. *Linrad* allows you to use multimedia instructions even if the system flag says they are illegal. This works fine under *Mandrake* 8.1. I do not know if it is because *Linrad* is the only software using the multimedia registers, so it therefore does not matter whether these registers are handled properly on task switching by the kernel.

From the initial screen, you can go

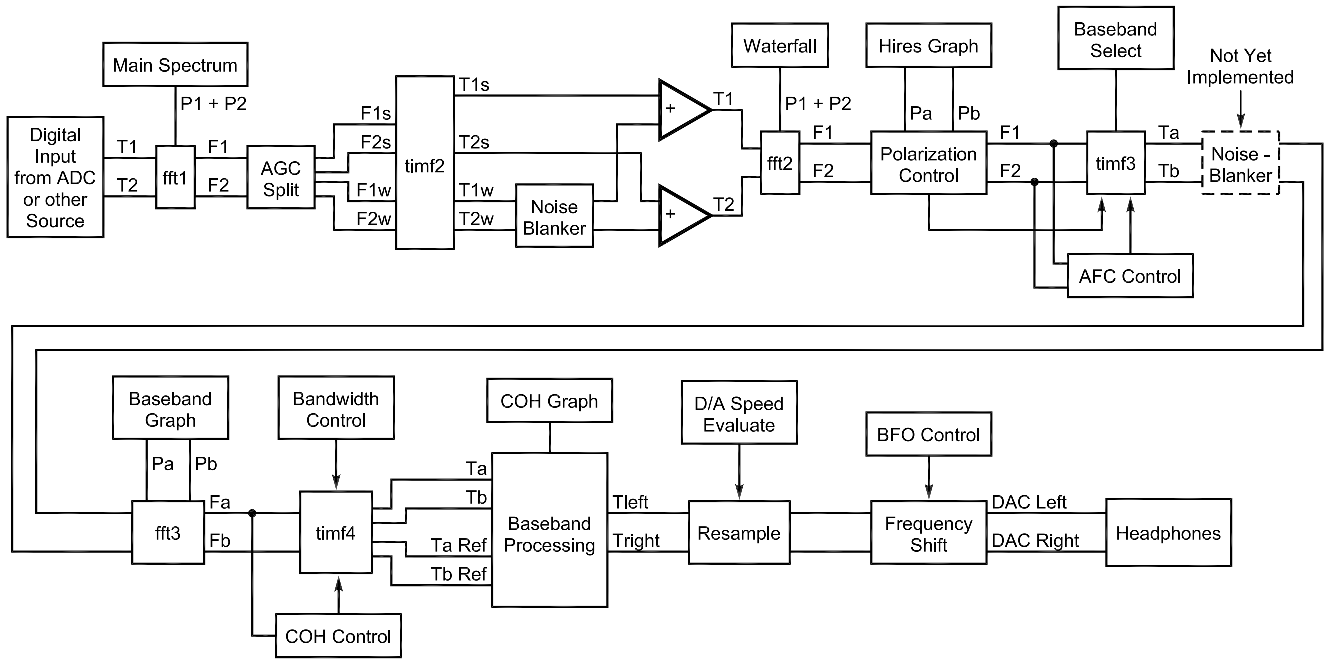


Fig 1—The block diagram of *Linrad* with two receive channels and the second FFT. T1 and T2 are signals in the time domain from two antennas 1 and 2. F1 and F2 are the corresponding signals in the frequency domain. Ta and Tb are linear combinations of T1 and T2 that make the desired signal zero in Tb and consequently maximizes the desired signal in Ta. TaRef is a time function constructed from a much narrower bandwidth than Ta. For Morse coded signals, it will be the CW carrier that is useful for coherent processing.

only to the video-mode selection where one of the graphic modes reported by *svgalib* can be selected. *Linrad* needs 256 colors and a minimum screen width of 640 pixels. I recommend a 1024×768 screen. If you do not see the video modes expected for your hardware, something is wrong with *svgalib*. It is possible to change the device drivers included in *svgalib* by changing its *Makefile.cfg* file and then recompiling.

After selecting screen size, you need to select a font size. Start with the minimum size. Finally, you are prompted for mouse speed, where 64 is a suitable starting value. If your mouse type is not the one presented on the screen, you need to edit the */etc/vga/libvga.config* file. After you hit ENTER, *Linrad* will switch to the graphic screen you have selected. If the screen looks okay, save the settings you've made so far by pressing *W*. The screen you should have at this point is the main menu. In case the screen does not look okay, it is possible to instruct *svgalib* to change the video signals by editing the */etc/vga/libvga.config* file. You may also try another screen size.

Setting the SoundCard Parameters

Linrad assumes you always use the same hardware to feed the computer with digital data. Therefore, the sampling speed and data format is set only once, from the main menu. Press U to set the soundcard parameters. If you try something else, you will be prompted to this setup anyway if parameters are undefined.

Linrad opens all device drivers it can find. Some of them may be defective: They may belong to some other sound system than the one you actually have running. There may be several soundcards in the computer. The user should know which drivers belong to which soundcard, but you may try what seems reasonable. On a simple, standard system where there are no alternatives, *Linrad* selects the only possible alternative automatically.

After having selected the input device, you need to select the sampling speed and the input format. Do not set a higher speed than necessary. Oversampling will load the CPU, but it will not improve performance much. When everything is set up, you may check that the sampling speed is correct for your hardware by checking how a strong signal is attenuated as you tune it upward in frequency. When it reaches the Nyquist frequency (half the sampling frequency), it is folded

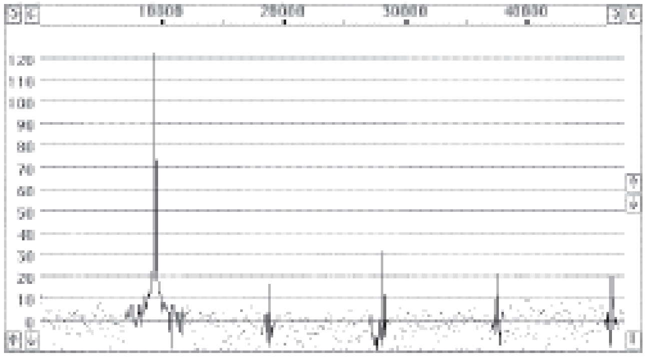


Fig 2—Unwindowed Fourier transform in 512 points. The bin bandwidth is 93.75 Hz and the near saturating signal occupies a single bin only because the frequency is carefully adjusted to be exactly on the center of an fft bin. This graph like Figs 3 to 8 are *Linrad* screen dumps on which straight lines are drawn between pixels for better visibility.

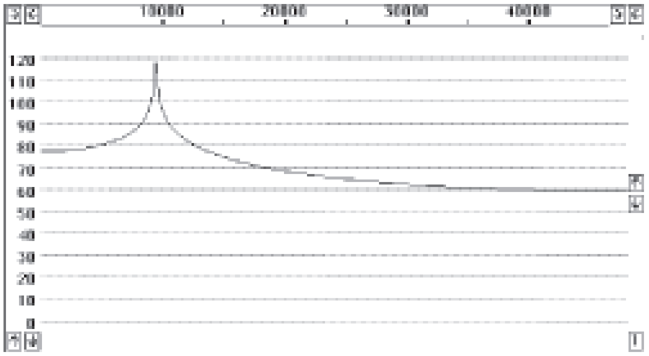


Fig 3—Same as Fig 2, but this time the frequency is adjusted to a point right between two frequency bins. Unwindowed ffts provide poor dynamic range and a poor shape of the spectrum peaks.

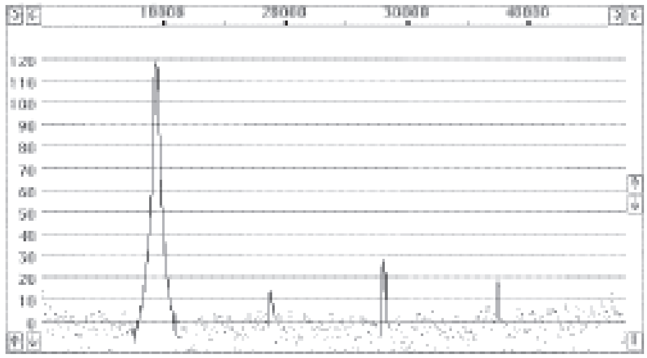


Fig 4—A sine^4 window provides a nice spectrum with a peak shape that is nearly independent of how the frequency is related to the fft bins. The peak is about 2.5 bins wide at the 6-dB points. This is because the window makes so many data points very small so the length of time during which the window does not attenuate very much is only 40% of the total transform time.

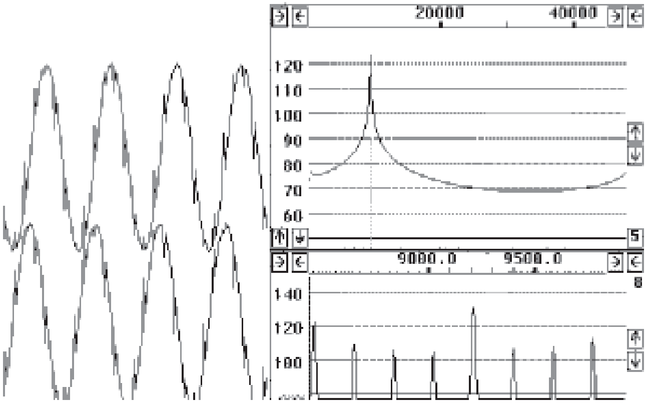


Fig 5—Unwindowed back transformation. Eight data points from the unwindowed transform are used to compute the baseband signal by back transformation. The time function comes in blocks of eight points, and there is a discontinuity each time two time functions are joined together. In this figure, the eight points are centered on the strong signal. The baseband spectrum computed from this time function shows these discontinuities as spurs, which are only about 20 dB below the signal. The origin of the transients is explained in the text.

back as an alias signal that appears to go down in frequency while the real signal is still tuned upward. The sampling speed should be set high enough to make the alias signal attenuated well enough when it has reached into the desired passband. The *Linrad* setup dialog is intended to be reasonably self-explanatory, and the *Linrad* homepage has links to detailed information.

Parameters That Depend on Receive Mode

The First Group: fft1

There are several distinctly different modes of operation for *Linrad*. As of this writing, only the weak-signal CW mode is partly implemented. It is possible to set weak-signal-CW parameters to get a system that is reasonably well adapted for SSB or normal CW. These modes can be selected from the main menu so you can switch quickly between modes, but the code actually executed remains that for weak-signal CW, until dedicated routines for the other modes are in place.

The total number of receive-mode-dependent parameters is quite long. The mode-dependent parameters are set from several screens; you are prompted to them when running a new mode for the first time, but you can also go there from a menu. Each screen controls its own part of the block diagram shown in Fig 1. By simply pressing ENTER at each of the parameter screens, you will select default parameters that will usually provide a reasonable starting point.

The first processing block has these parameters:

- First FFT bandwidth
- First FFT window (power of sine)
- First forward FFT version
- First FFT storage time (s)
- First FFT amplitude
- Enable second FFT

These parameters control block *fft1* in Fig 1. The bandwidth and the window control the size of the FFT. It is in powers of two, so you do not get exactly the bandwidth specified by the parameter. The window controls the shape of the filter associated with each FFT bin. A higher value yields a wider bandwidth in FFT bins but steeper skirts; a larger FFT may be required to get the desired bandwidth in hertz. A higher value also causes more CPU processing load because transforms must overlap more. The different FFT implementations may run at different relative speeds depending on processor and memory architecture. The storage time is important only if the second FFT is disabled. It tells *Linrad* how much memory to allocate for old transforms; they are used for AFC and spur rejection in the absence of the second FFT. The amplitude parameter can be used to fool *Linrad*. This, in case the signal level is too high, causing the noise floor to be treated as a strong signal—something likely to happen if *Linrad* is used to process an ordinary .WAV file.

When the second FFT is disabled, the output of *fft1* is routed directly to *timf3*, disabling the noise blanker. The input to *fft1* is the raw data from the sound card or raw data from a file. The input may be one or two channels in real or complex format as defined by the sound-card parameters or the raw data file. In the block diagram of Fig 1, T1 and T2 represent the two time functions from two antenna signals. The help key, F1, can be used to get some information about the effects of these parameters.

When choosing bandwidth and window, you build the basic filter used in the *fft1* block. Figs 2 and 3 show what happens when the window parameter is set to zero (no window). These figures were produced by running *Linrad* with a direct-conversion receiver at 2.5 MHz while sending a very pure sine wave at 2.509 MHz into it. In Fig 2, the frequency is adjusted for the signal to be located exactly on one frequency bin while Fig 3 shows what happens when the signal is located right between two bins. The figures are unaveraged power spectra of size 512.

The great difference between Figs 2 and 3 results because the Fourier transform is the spectrum of a signal that is obtained when the 512 samples of the input data are repeated continuously. When the frequency matches a bin exactly, the first point will fit exactly to the sine wave when placed as point 513 for the next repetition. When the frequency is exactly halfway, point 1 will not fit at all when placed as number 513. There will be a large discontinuity, and that discontinuity has signal energy over the entire frequency spectrum.

Unwindowed FFTs are very fast, but they do not allow a large dynamic range. When a group of bins is picked to make a rectangular filter, the stop-band attenuation will only be about 50 dB for a 512-point FFT. With larger transforms, the situation improves because the discontinuities repeat less often; but the phenomenon is still there.

The way to cure the problem is to use a window. Fig 4 shows exactly the same signal as Figs 2 and 3 but here a sine^4 window is applied. With such a high-order window, it does not matter whether the signal is centered on a bin or not. The points toward the ends of the time sequence are all extremely small, so the discontinuity is very well suppressed. Notice that the maximum is much wider. There are three points on the peak now, so a 2048-point FFT will be necessary for the same inherent bandwidth.

When *Linrad* is used as a radio receiver, a window power of two or three is sufficient. When using it as a high-performance spectrum analyzer with the first FFT generating broadband spectra, however, windows up to a power of nine may be used. That makes it possible to study extremely low levels of unintended sidebands on the test signals, if the hardware has the quality to allow it. The hardware used to produce Figs 2-4 represents the unit described at **antennspecialisten.se/~sm5bsz/*Linux*dsp/rxiq/opt2500.htm**. The unit can be purchased from **www.antennspecialisten.se**.

The choice of bandwidth and filter depends on what you are going to do with the output data. When the second FFT is disabled, the output is displayed *and* is used to produce a frequency-shifted signal in the time domain by *timf3*. When the decimation rate for *timf3* is low, the alias spurs from the *timf3* point-decimation process are far from the desired frequency. Then the basic filter skirts need not be very steep to suppress alias signals well. When the decimation rate is high, the filter requirements are higher. High decimation rates allow the computer to do advanced processing on many signals simultaneously. When the hardware is an ordinary SSB receiver, the dynamic range within the SSB bandwidth would likely be below 50 dB anyway, and then it does not matter much what window you choose.

Bandwidth selection depends on a compromise between the resolution you desire on the screen and the time delay you are willing to accept. If you want 1-Hz resolution with a sine^4 window, each transform spans about four seconds and a corresponding time delay is unavoidable. A high resolution is required to make the AFC follow extremely weak and unstable CW signals well enough to use coherent processing. AFC is not required for stable signals, and then there is no need for high resolution. A bandwidth of 20 Hz gives good visibility for CW signals on the waterfall graph with a modest processing delay. By setting a large bandwidth for *fft1*, it is possible to make very fast waterfall graphs from which you can read high-speed CW signals directly.

When the second FFT is enabled,

the output of *fft1* is used to produce notch filters for all strong signals in the passband. There is no reason to select a narrow bandwidth, but good skirt steepness is required. Typically, 200 Hz and sine^3 will fit high-performance broadband hardware. If you are going to select a very narrow bandwidth for the second FFT, it may be a good idea to select a smaller bandwidth for *fft1.* The ratio of *fft1* to *fft2* has some influence on how deep the notches for strong signals must be. If your computer is capable of making the *fft2* bandwidth 0.2 Hz, it would be unlikely to have problems making the *fft1* bandwidth 10 Hz or so, particularly since a less-demanding window would then be required.

Linrad has oscilloscope functions that allow you to see the time functions. With a signal generator, you may check the spurious responses that occur as a result of inadequate filters both in the time domain and in the frequency domain. With a poor filter, you will easily see the resampling spurs with the second FFT disabled.

The upper right part of Fig 5 shows the *fft1* spectrum in 256 points from 0 to 48 kHz of the same signal as used for Figs 2-4. No window is used.

The spectrum is frequency shifted to place the desired signal at zero frequency. The baseband signal is then filtered out by multiplying the transform with a nicely rounded filter function. For Figs 5-8, this filter is in eight points. The baseband signal is then obtained from an inverse FFT. The filter makes the 248 points outside the baseband equal to zero, so there is no reason to do the inverse transform in more than eight points. This is how decimation comes in, naturally. The two sine waves at the left are the resulting complex-time function. This is the baseband signal I and Q for the selected frequency. These sine waves are made up from segments of eight points, each spanning the same time as one frame of *fft1*. Since the number of points is reduced by a factor of 32, the bandwidth is reduced from 48 kHz to 1.5 kHz with a sampling speed of 1.5 kHz. The bandwidth is actually reduced a bit more because of the rounded shape of the filter used to pick the eight points. The lower-right part shows the baseband spectrum in 256 points. The signal itself is at 9200 Hz and the resampling spurs are separated by 187.5 Hz, the inverted value of the time for one *fft1* frame. The accuracy of this spectrum is poor towards the ends, the screen shows the spectrum divided by the filter function, now expanded from eight to 256 points and this division becomes division by zero at the ends.

The resampling spurs show up in the time function at the left side of Fig 5 as spikes that repeat at an interval of eight points. The explanation of why these spikes occur is that an unwindowed FFT has to use all of the frequencies to reproduce a sine wave in case it is not exactly at one of the bin frequencies. That is what we see in the *fft1* spectrum, the signal is at about 70 dB or more at all frequencies. When only eight points are selected, the inverse transformation will not give back the original signal; it gives something that differs at the beginning and end of the time interval. Successive transforms do not fit well to each other so spikes are generated. The way to avoid this problem is to use interleaved transforms and not use the end regions for the inverse transform. The associated spectra are then calculated with windowed transforms.

Fig 6 is the same as Fig 5 except that here, the 1.5-kHz segment used for the inverse transform is centered 3 kHz above the strong signal, which is now outside the range used for the inverse transform.

As can be seen in the main spectrum, the level of the unwindowed transform is at about 85 dB at 12 kHz and that signal, which is an artifact, shows up as spurious signals separated by 187.5 Hz.

What we see in Fig 5 and Fig 6 is what happens when resampling is done without an appropriate anti-alias filter. As was pointed out, with appropriate references, by Gerald Youngblood, AC5OG, (*QEX*, Nov/Dec 2002, p 31), filtering in the time domain and filtering in the frequency domain are equivalent. The problem is that the filter used to generate Figs 5 and 6 was not the nicely rounded filter in eight points. It is the sum of the output from eight filters (with frequency responses like the unwindowed FFT) summed together with weights according to the nicely rounded function.

Fig 6 clearly shows what is required to suppress the alias spurs. By using only the four center points of each transform and calculating twice as many interleaved transforms, we can construct a baseband time function from the inverse transform that does not include the end discontinuities. *Linrad* does it by applying a window because the power spectrum is needed for AFC, spur cancellation and to produce graphs. The transforms are interleaved for the 3-dB points of successive window functions to match the same points in the time function. To avoid problems related to the AGC (the gain may change between two transforms), *Linrad* makes soft

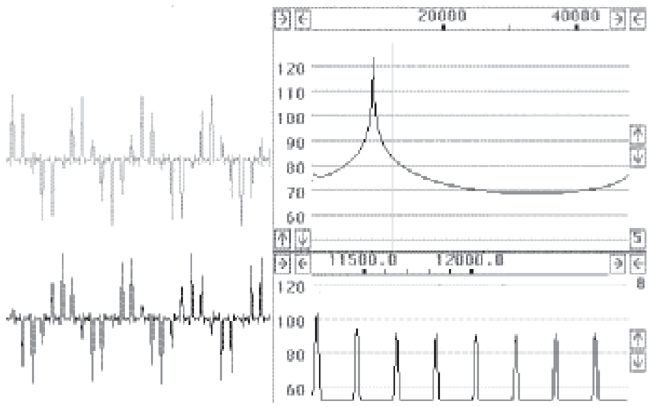


Fig 6—Unwindowed back transformation. Same as Fig 5 but this time the eight points used for the baseband are centered 3 kHz above the strong signal. The signal itself, the sine and cosine is now outside the baseband, but the discontinuities are very strong. As can be seen from the main spectrum, they are present at all frequencies.

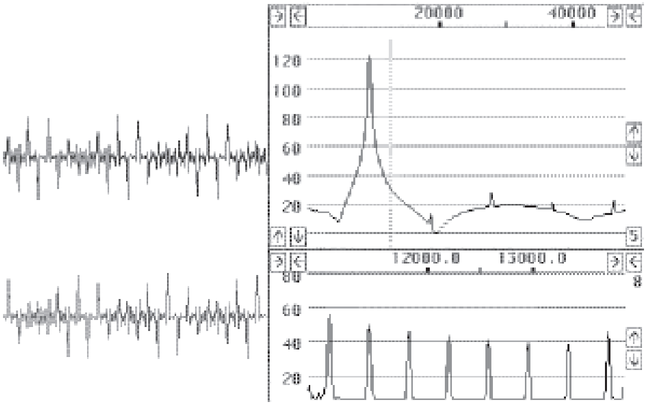


Fig 7—A sine^2 window makes the spectrum look much better. With the high dynamic range of the hardware used to produce these images, the sine^2 window is not good enough to bring the discontinuities below the noise floor with a small transform size like 256.

transitions between successive inverse transforms. A sine-squared window produces the filter function shown in Fig 7. When the baseband is filtered out by weighting together eight such filters, the alias signals are reasonably well suppressed as can be seen in Fig 7, both in the baseband time function and its spectrum.

By selecting a longer *fft1* transform, it is possible to suppress the alias spurs below any desired level. This is done by setting the "First FFT bandwidth" parameter narrower or by setting a narrower window function by making the "First FFT window (power of sine)" parameter larger. Fig 8 shows what happens if a sine^4 window is chosen for a size-256 *fft1*.

It is self-evident that one can avoid spurs completely by use of a transform size and window shape that reduces contributions from undesired signals to well below the noise floor for all data points used in the inverse transform. When a large baseband bandwidth is desired, the offending signal may be included among the data points used for the inverse transformation. This will be perfectly okay if all data points containing energy from the undesired signal are picked with exactly the same amplitude. As already mentioned above, the frequency range to be back-transformed is selected with a nicely rounded filter, which may lead to unequal amplitudes for the different data points on a strong signal. By selecting the bin bandwidth of *fft1* much narrower than the baseband bandwidth, one can make the difference in amplitude small, thereby placing these spurs below the noise floor.

When the second FFT is enabled, all the frequency bins of *fft1* are inverse transformed. Frequencies where strong signals are present are attenuated, however, and that creates a similar problem to the one just described. If the window or *fft1* size is inadequate, spurs will show up in the waterfall graph. By injecting a near-saturating signal into the antenna input, one can verify that such spurs are invisible and well below the noise floor of the hardware in use.

The Second Group: AGC, timf2 and fft2

These blocks are present only when the second FFT is enabled. They are controlled by the following parameters:

- First backward FFT version
- First backward FFT attenuation *N*
- Second FFT bandwidth factor in powers of two
- Second FFT window (power of sine)
- Second forward FFT version
- Second forward FFT attenuation *N*
- Second FFT storage time

When the second FFT is enabled, the main purpose of the first FFT is to find the frequencies on which strong local signals are present. Such frequencies are attenuated by the AGC to the extent that the strong signals will not overflow when further processing is done with 16-bit arithmetic. The AGC block splits the output of *fft1* into two groups. One contains the narrow-bandwidth signals that have been located in *fft1*, F1s and F2s in Fig 1. The other essentially contains the noise floor from the rest of the spectrum, F1w and F2w in Fig 1. The two sets of *fft1* transforms are back transformed in the *timf2* block to produce two sets of time functions; T1s and T2s contain strong signals, while T1w and T2w contain weak signals.

The noise blanker operates on T1w and T2w only. That means that it operates on a signal that has passed notch filters removing all strong signals that otherwise would have disturbed the operation of the blanker. The *Linrad* noise blanker is far more sophisticated than a conventional noise blanker and will be described separately.

Once noise has been removed from the weak signals, weak and strong signals are added together and sent to the *fft2* block. That block again converts the signals to the frequency domain with the output signals F1 and F2 as shown in Fig 1.

Since the AGC ensures that the data will fit into 16 bits, selecting the first inverse FFT implementation that uses MMX multimedia instructions saves a lot of CPU time because MMX is about three times faster than floating point. Likewise, the second FFT should be set to use MMX if the processor supports it.

Since some processing is designed to use 16-bit numbers, it is essential to make sure that the signal levels are set properly. The "First backward FFT attenuation N" and "Second forward FFT attenuation N" parameters set the signal levels by telling how many of the FFT butterfly loops should contain a right shift. Each right shift divides the signal level by two, which is attenuation by 6 dB. Gain set too high will cause saturation, while gain set too low will degrade the noise floor by the addition of quantization noise. For details, follow the link "set digital signal levels correctly" on the *Linrad* home page.

The parameter "Second FFT storage time" tells how much memory to allocate for old frequency-domain data. If the computer has enough memory, it is advantageous to allow a long time here. The AFC will be limited to the time interval specified here, and if the computer has plenty of memory, there is no reason to not use it.

The Third Group: AFC and Spur Cancellation

When this group is enabled, a group of parameters defining maximum AFC lock range, maximum number of spurs to cancel and so forth, must be selected. These parameters essentially allocate memory and set the upper limit for the operator's current preferences that he can select with the mouse. The AFC will lock to the strongest signal within the specified range, and to avoid locking to stronger undesired signals close to the desired one you may need to use a narrow lock range.

The functions in this group use the frequency-domain data from the entire spectrum. The input is the output of *fft2* if enabled, otherwise the input is the output of *fft1*. The third group will also contain the automatic Morse-code-to-ASCII translation, but this part is not implemented yet.

The Fourth Group: Baseband Processing

This is the last group of mode-

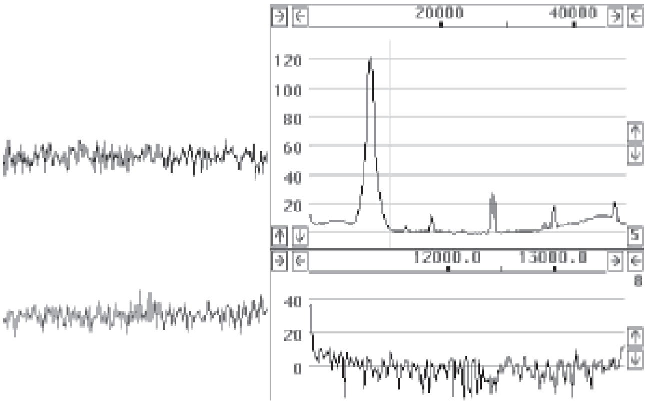


Fig 8—A sine^4 window is enough even on a small transform like 256 to make the near saturating signal narrow enough not to enter the baseband even though the baseband starts only 1.5 kHz above the strong signal.

dependent parameters. They are:

- First-mixer bandwidth reduction in powers of two
- First mixer number of channels
- Baseband storage time
- Output delay margin
- Output sampling speed
- Default output mode
- Audio expander exponent

The first-mixer bandwidth reduction is simply the decimation rate in the frequency-shifted inverse transformation that produces the baseband signal as discussed in conjunction with Figs 5-8. By selecting a large decimation rate, one saves CPU time and memory, but the baseband bandwidth is reduced at the same time.

The baseband noise blanker between *timf3* and *fft3* should operate at a bandwidth that is well above the bandwidth of the desired signal. As long as it is not implemented, there is no reason to use baseband bandwidths that are more than two times larger than the bandwidth of the desired signal. The factor of two is needed to accommodate a filter function.

The way the baseband signal is extracted from the broadband input signal is very efficient, and it is possible to process a large number of signals simultaneously. At present there is no reason to select more than one channel because only one channel can be sent to the loudspeaker/headphones. More channels will be useful when the Morse-code-to-ASCII routines are in place. For the other parameters, press F1 with the mouse cursor on one of them to get information.

The Receive Screen

Once the mode dependent parameters are set, *Linrad* enters receive mode and presents the receive screen to the user. The following windows are present:

- Main spectrum and waterfall
- High resolution spectrum
- Baseband spectrum
- AFC window
- Polarization control
- Coherence graph
- EME window
- Frequency control

Buttons in each window allow the user to change processing parameters. The spectra can be zoomed in and out and the user may move the windows around and set the sizes of the different spectra as desired.

The basic operation of *Linrad* is extremely simple. Move the mouse cursor onto a signal that is visible in the main spectrum and click the button. The corresponding signal, filtered through the baseband filter, will immediately be sent to the loudspeaker. The shape of the baseband filter can be modified with the mouse and the baseband frequency is shifted to the desired audio frequency by setting the BFO frequency.

All the processes are more or less automatic; but to take full advantage of them, one needs a basic understanding of how they work. Subsequent articles will describe some of these processes in detail. Some information can be obtained with the F1 help key, and there is a lot of information available at the *Linrad* home page.

Summary

This article has concerned itself with the coarse structure of *Linrad* and its block diagram. The significance of window functions for dynamic range has been illustrated; this problem is always encountered in SDRs when a signal is resampled at a lower speed. Subsequent articles will focus on how to use the special features of *Linrad* to improve receiver performance. □□

Rover Software for the PALM Organizer

Put your PDA to work figuring bearings, distances and logging contacts.

By Paul Wade, W1GHZ

A number of small handheld computer-like devices have appeared on the market. Surely there must be ham applications for them.

One of the most popular is a PDA (personal digital assistant) line from PALM Computing. It is not only useful for daily life, but also easy to program for new applications. Rather than just describe the ham programs I have developed, I will also discuss the evolution of the programs to best utilize the capabilities of the Palm platform.

Beginnings

In the summer of 1999, I acquired a Palm V PDA. The original version of the Palm was the Palm Pilot, a name that is often used for all versions. For those who haven't seen one, think of a handheld computer about the size of a chocolate bar. Some useful software is included: an address book, calendar, memo pad, "to do" list and calculator. A wide range of shareware is also available; for instance, I quickly replaced the original calculator with a scientific calculator using reverse-polish notation. Data entry is done with a stylus, by either using a stylized handwriting or tapping on a tiny keyboard with the stylus, as shown in the screen image in Fig 1.

After growing accustomed to the Palm V, I could see that this handheld device is an ideal tool for rover operation—but what about ham software? On the Internet, I didn't find anything specific, but did find a shareware *C* compiler, *PocketC*,[1] so I could develop my own. For starters, the *BD* (Bearing-Distance) program has always been essential for grid-square calculation used in VHF+ operation, and Matt, KB1VC, had already translated it into *C* code. I added some I/O (input-output) code for the Palm, and stripped the rest down to the essentials: enter two 6-digit Maidenhead locators, get back bearing and distance. The whole program requires only 4 kB of storage (total Palm V storage is 2 MB—newer models have more).

The other essential for a rover is logging. With the ARRL 10-GHz & Up Cumulative Contest approaching, I wrote a simple logging program. Again, the approach is minimalist: Record the essential information as simply as possible. Each time the program is started, you are prompted for current location, then for a call and grid for each contact. Each QSO is time-stamped and recorded, including distance in kilometers. Powering down the Palm computer does not terminate the program—it picks up instantly where you left off—so logging is quick and easy, and battery life is at least a week.

However, after the first weekend of contest use, we realized that switch-

161 Center Rd
Shirley, MA 01464
w1ghz@arrl.net

[1] Notes appear on page 51.

ing back and forth between *BD* and the logger was inconvenient. Matt suggested that a better approach would be to incorporate *BD* into the logger: Enter the grid, get the heading then enter the call upon successful completion of the contact. The next versions operated in this manner. All this sophistication bloated the program up to 7 kB of storage.

I originally made two versions of the logging program: *10G_LOG* for 10-GHz-only operation, and *LOG_CUM* for multiband operation. The logged fields are band, date and time, calls, grids and distance, as needed for the ARRL contest. Rather than invent a fancy database, the log is kept as a memo in the Palm built-in *MemoPad* application, writing one line per QSO. Operations like duping, formatting, QSO points and totals are best done after the contest on a desktop PC system. The Palm has a standard feature that transfers information to the desktop system via a serial or USB port and keeps both systems current with a "Hot-Sync" button.

Evolution

The simple logging program required text entry of all data, just as if the user had a keyboard. A few users helped by testing several versions through bug fixes and improvements. Finally, Simon, GM4PLM, suggested that it would be easier to select the band from a menu rather than enter it for each QSO.

Simon's suggestion sounded reasonable, so I looked for ways to implement it. Accompanying the *PocketC* compiler are some sample programs, including a tic-tac-toe game, played by tapping the desired box with a stylus. I modified the code to display a band in each box and detect which band was tapped, then included this function in the logging program. I'm not a professional programmer, but I do know that modifying working code for new uses improves programming productivity.

A short digression about *C* programming might be in order. *C* is a compiled language, which must be translated (compiled) into machine language before it can be executed, or run on the computer. *BASIC*, on the other hand, is often an interpreted language—the computer uses a *BASIC* interpreter, translating each time it runs a program. The compiled code typically runs much faster, which may not be noticeable on a fast Pentium PC, but the processor in a handheld device is much slower for reduced power consumption.

Good programming in any language uses a modular approach, using functions (in *C*) or subroutines (some in other languages) for specific operations. I wrote the logging program as a set of modular functions, so that operations might be enhanced by replacing the appropriate function. For instance, incorporating *BD* into the logger was a simple matter of adding a call to the `bd()` function. The compiler does the rest! Later, we will see that making several different versions of the logging program for different contests is done by creating different `main()` functions that call the appropriate functions for that version—the compiler uses those functions and ignores the rest.

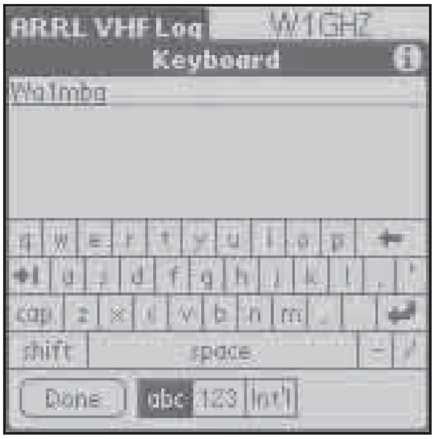
Table 1 shows the *C* code for the `main()` function of my general-purpose log. All the text on a line after "//" are comments, always a good programming practice to explain what is happening; even the original programmer forgets eventually. As you can probably see, this routine doesn't do much, just starts the program, finds the desired log in the memo pad, and then calls another function, `AddVHFUpLog_BD(mycall)` to do the actual logging.

Another useful programming practice is to generalize functions by using parameters. In the case of the tic-tac-toe board, we can generalize it from 3×3 boxes to *X*-by-*Y* boxes; *X* and *Y* are parameters. I included additional parameters for the size of the boxes and position on the screen, so that I could use one function to create any number of menus. Table 2 shows the function that draws the lines on the screen to create the boxes.

Finally, when I select a grid by tapping it with the stylus, a few Palm-specific instructions find the coordinates of the tap on the screen. Then the coordinates are converted to an offset—the number of grids in each direction away from my current grid, as shown in Table 3.

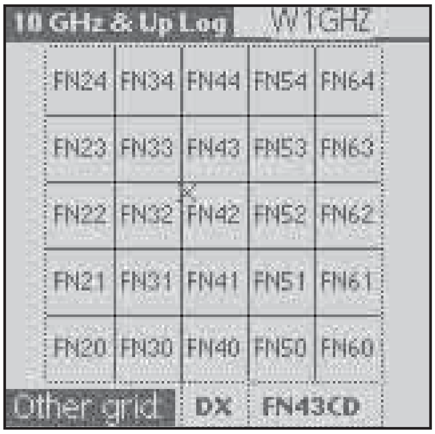
I hope these bits of code give you some of the flavor. If you like it, look at the rest of the source code in 0303LOGS.ZIP;[2] I've tried to provide enough comments to make it readable. Perhaps you will be inspired to improve on it and personalize your log.

More important than the implementation language and detail is the design philosophy. I bought and read



Wa1mba

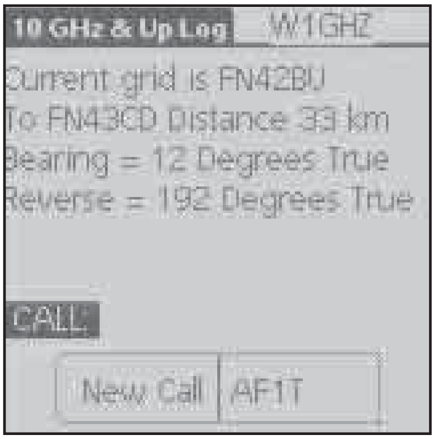
Fig 1—The keyboard pattern used by the Palm V for text input.



FN24	FN34	FN44	FN54	FN64
FN23	FN33	FN43	FN53	FN63
FN22	FN32	FN42	FN52	FN62
FN21	FN31	FN41	FN51	FN61
FN20	FN30	FN40	FN50	FN60

FN43CD

Fig 2—A screen capture of the 5×5 box matrix used for nearby Grid input.

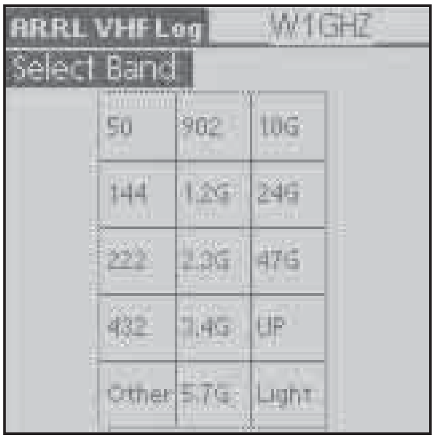


Current grid is FN42BU
To FN43CD Distance 33 km
Bearing = 12 Degrees True
Reverse = 192 Degrees True

CALL:

AF1T

Fig 3—Results of calculations performed by *BD,* along with a prompt for a new call sign.



50	902	10G
144	1.2G	24G
222	2.3G	47G
432	3.4G	UP
Other	5.7G	Light

Fig 4—A single stylus tap can enter bands up through 47 GHz. Higher bands require a text entry.

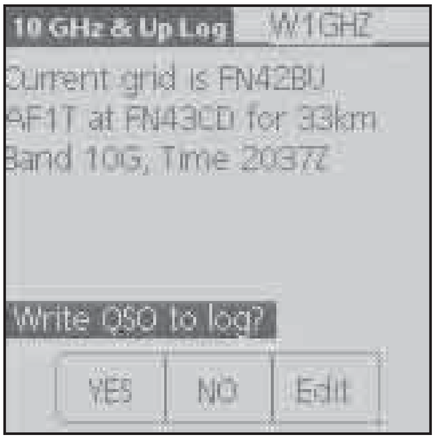
Table 1—*C* code for the main()function of the General-Purpose Log

```
// VHF & Up
// VHF & Up log
// by Paul Wade W1GHZ
// version 1.0 December 1999
// precalculate grid map and store
// version 0.9 December 1999
// HISTORY:
// from
// Log Cum
// simple log for the Microwave Cumulative Contest
// by Paul Wade W1GHZ
// version 0.6 November 1999
// select grid and band by tapping on map
// version 0.5 September 1999
// don’t write null calls
// version 0.4 August 1999
// ask for grid, return heading, then ask for call
// version 0.3 August 1999
// fixed GMT offset saving


// GLOBAL VARIABLES:
int Zoffset;  /* global GMT offset */
float Distkm, Baz;  /* ugly globals to avoid pointers */
string adj_grid[25];  /* precalculate adjacent grids for map */

// files with called functions
include “GridFunc.c”
include “BD_full.c”
include “Gridpeck.c”
include “BandPeck.c”
include “LogFunc.c”

// note: functions lack type in PocketC

StartVHFUpScreen()
{
// text(x,y,string) is a graphics mode function
// screen is 160x160, x,y=0,0 is upper left
  text(110,0,”v1.0");
   text(0,20,”VHF & Up Log Program”);
   text(0,35,” by W1GHZ 1999 “);
   text(0,50,”General VHF+ Logging”);
}

main()
{
  string mycall;
  clear();
   graph_on();           // use graphics mode
  clearg();

   title(“VHF & Up Log”);   // at top of screen

  textattr(2,1,0);

  StartVHFUpScreen();

   GMTadjust();          // offset for GMT

  StartVHFUpScreen();

   mycall = gets(“My Call:”);   // gets() creates a text input box

   mycall = strupr(mycall);

// look for a MemoPad record starting with the string ‘mycall’
   if (mmfind(mycall))           // found, use it
     {
     clearg();
     text(100,0,mycall);
     text(0,20,”Log file found”);
     AddVHFUpLog_BD(mycall);    // add entries to existing log
     }
   else if (mmnew())             // not found, start one
     {
     NewHeader(mycall);               // create the new log memo
     clearg();
     text(100,0,mycall);
     text(0,20,”New Log file started”);
     if (mmfind(mycall))   // make sure the log is there
       AddVHFUpLog_BD(mycall);        // add entries to the log
     }
   else                          // can’t find or create
     {
     clearg();
     textattr(2,2,0);
     text(0,35,”Couldn’t open log file “);
     textattr(2,1,0);
     beep(3);
     }

   mmclose();                    // finished: close the MemoPad
}
```

a book on Palm programming,[3] which helped me understand how to use the Palm effectively. One important concept is to consider the Palm as an extension of the desktop display, rather than as a standalone computer; by relying on the desktop to post-process the log, I was heading in the right direction. Another concept is to minimize the amount of data input required; my text-based logging had already proven slow and tedious.

I thought about VHF+ contest operating: How could I make logging faster and simpler? First, unless a station is in a remote location, many of the contacts are in nearby grid squares; a menu of adjacent grids would be much easier than entering each one. Next, we frequently "run the bands" with a station, working him or her successively on all bands we have in common; clearly, the grid and call don't change, so storing these from the previous contact is important. Since we use rather sharp beam antennas, another frequent occurrence is to work several stations in the same grid or portable location; again, storing the grid location is important. Finally, we occasionally hear or enter data wrongly initially, so easy editing of a QSO before confirming it for the log is important.

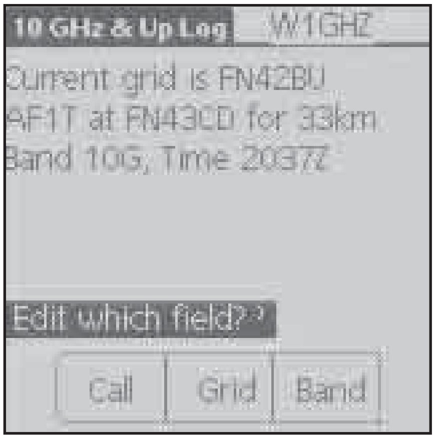
Now I had a design philosophy: Make these frequent entries easy, with a


10 GHz & Up Log W1GHZ

Current grid is FN42BU
AF1T at FN43CD for 33km
Band 10G, Time 2037Z

Write QSO to log?

Fig 5—Once the band is selected, the entry information is displayed for confirmation before it is stored.


10 GHz & Up Log W1GHZ

Current grid is FN42BU
AF1T at FN43CD for 33km
Band 10G, Time 2037Z

Edit which field?

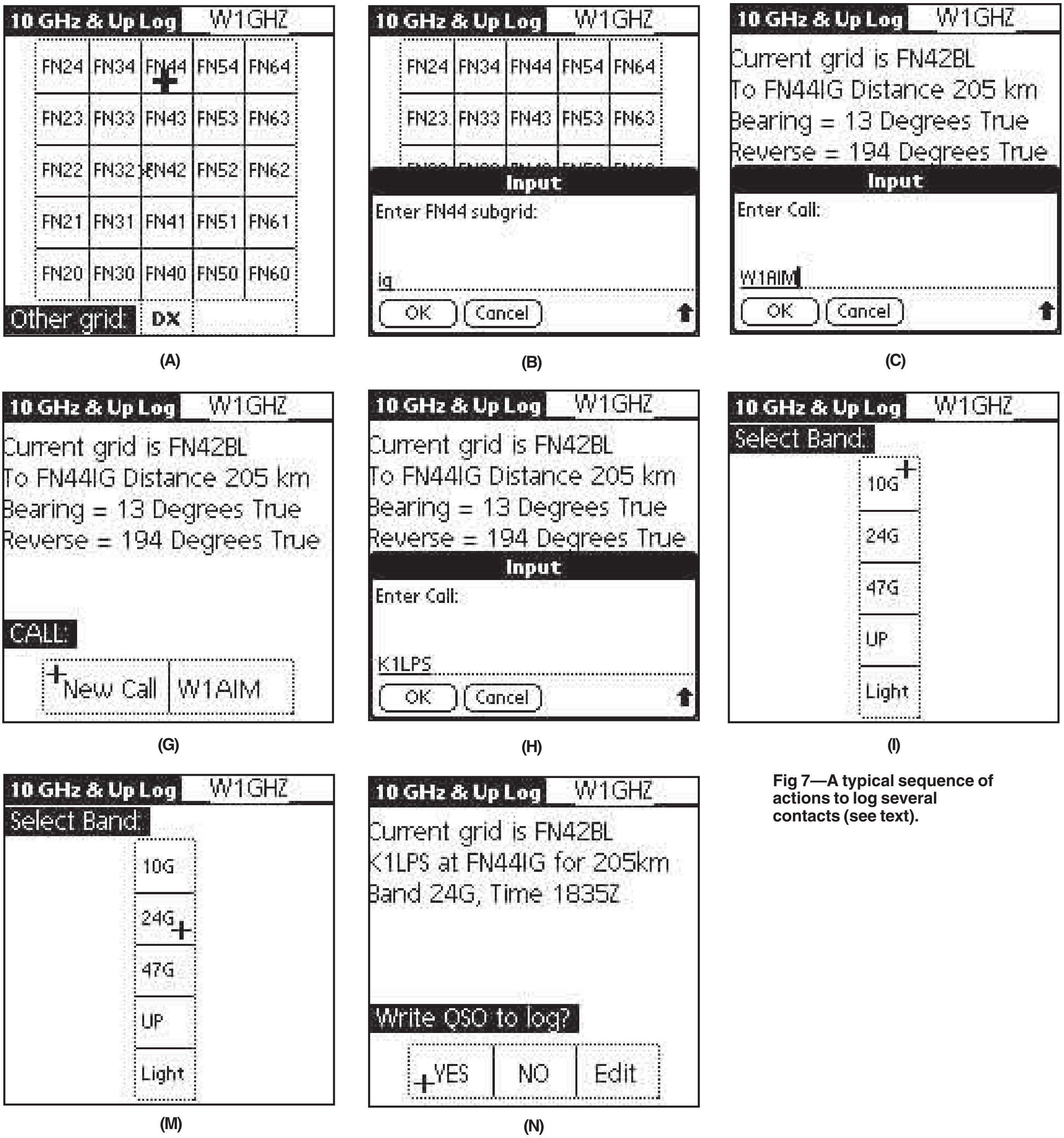
Fig 6—Tapping "Edit" in the screen of Fig 5 sends the user to this editing screen.

Table 2—A Function that Draws Lines on the Screen to Create Boxes

```
Outlineboxes(int xbox,int ybox,int xstart,int ystart,int xincrement,int yincrement)
// draws a set of boxes on the screen
// number of boxes is xbox by ybox
// xstart and ystart are lower left corner
// xincrement and yincrement are size of box

  {
  int m;

// vertical lines
   for (m = 1; m <= xbox; m++)
     line(1,(xstart + (m * xincrement)),ystart,
             (xstart + (m * xincrement)),(ystart + (ybox * yincrement)));
// horizontal lines
   for (m = 1; m <= ybox; m++)
     line(1,xstart,(ystart + (m * yincrement)),
           (xstart + (xbox * xincrement)),(ystart + (m * yincrement)));
// Draw the gray border
   frame(2, xstart,ystart,(xstart + (xbox * xincrement)),(ystart + (ybox * yincrement)),1);
  }
```

Table 3—Tap Coordinates are Converted to an Offset from the Current Grid

```
stylus_event_example()  // code snipet
// find the peck
// Wait for a stylus event
    {
      waitp();          // waits for pen event
      x = penx();       // returns pen x location
     y = peny();   // returns pen y location
   // which box was it in?
      x = ((x-xgstart)/xgincrement) - 2; //human readable direction
      y = ((y-ygstart)/xgincrement) - 2;
   // calculate grid using x and y
   // return grid
     }
```



(B)

Enter FN44 subgrid:

iq

(C), (G) and (H)

Current grid is FN42BL
To FN44IG Distance 205 km
Bearing = 13 Degrees True
Reverse = 194 Degrees True

(C)

Enter Call:

W1AIM

(H)

Enter Call:

K1LPS

(N)

Current grid is FN42BL
K1LPS at FN44IG for 205km
Band 24G, Time 1835Z

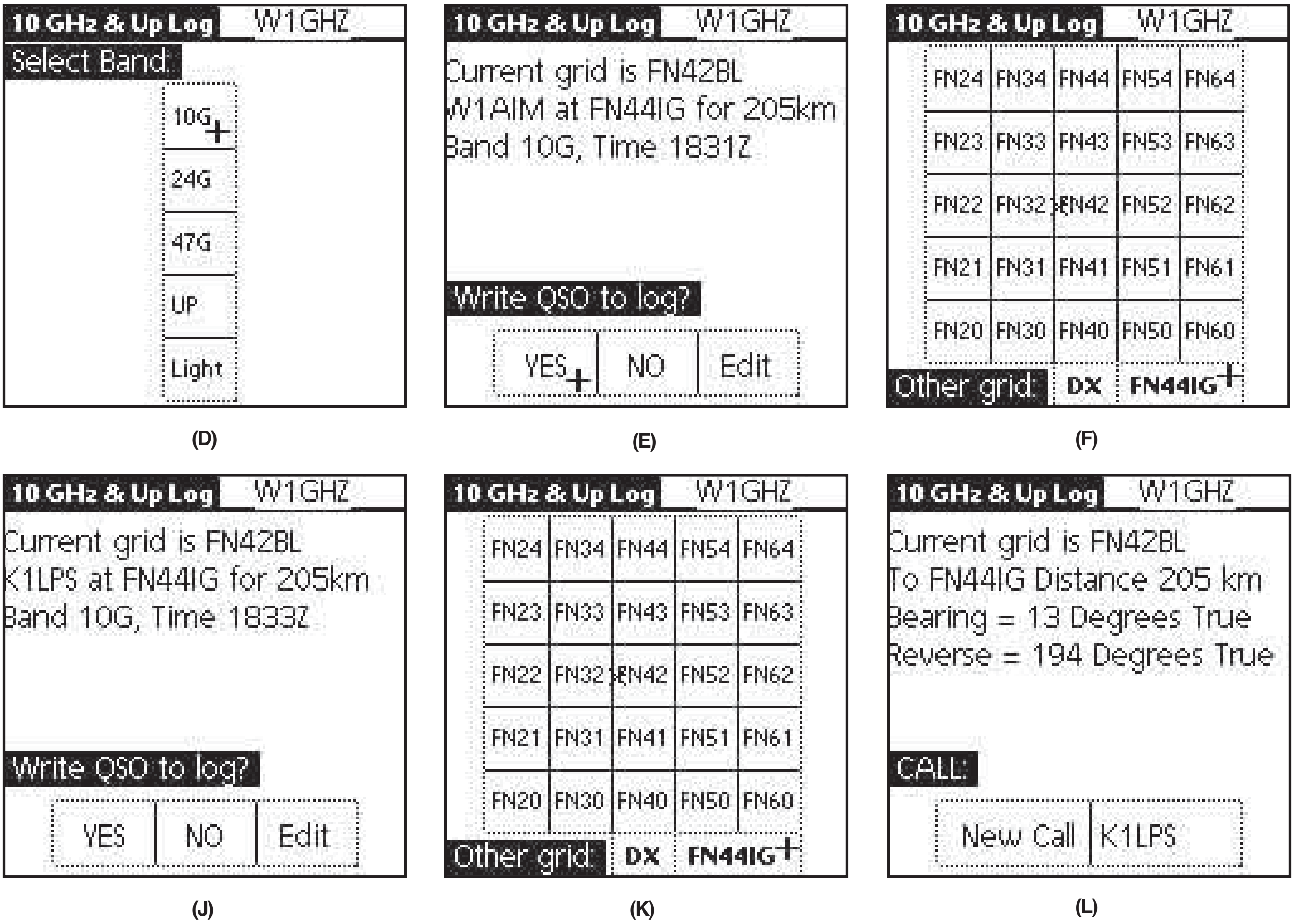
Fig 7—A typical sequence of actions to log several contacts (see text).

single tap of the stylus if possible, and use text entry only for unique data. The first entry of a call is certainly unique, but successive contacts with that station on other bands are not.

The obvious implementation for this design philosophy is the generalized tic-tac-toe menu. For grid selection, a 5×5 box menu allows selection of two concentric rings of grids in any direction, as shown in the screen capture of Fig 2. Note the small "x" at my actual 6-digit grid location in the central box, allowing quick visualization of approximate beam heading. The 4-digit grid (or the previous grid, in the lower-right box) is chosen with a stylus tap, then the final two digits are added as text. For grids outside the menu area, a "DX" choice prompts for text entry.

When the grid is complete, the bearing and distance are calculated and displayed, as in Fig 3, along with a prompt for the call. If the previous grid was chosen, a small menu gives you a choice: a new call or the previous one with a single tap.

After the call is entered, another menu, shown in Fig 4, allows choosing the band with a single tap.



(D)

10 GHz & Up Log W1GHZ

Select Band:

10G
24G
47G
UP
Light

(E)

10 GHz & Up Log W1GHZ

Current grid is FN42BL
W1AIM at FN44IG for 205km
Band 10G, Time 1831Z

Write QSO to log?

YES NO Edit

(F)

10 GHz & Up Log W1GHZ

FN24	FN34	FN44	FN54	FN64
FN23	FN33	FN43	FN53	FN63
FN22	FN32	FN42	FN52	FN62
FN21	FN31	FN41	FN51	FN61
FN20	FN30	FN40	FN50	FN60

Other grid: DX FN44IG

(J)

10 GHz & Up Log W1GHZ

Current grid is FN42BL
K1LPS at FN44IG for 205km
Band 10G, Time 1833Z

Write QSO to log?

YES NO Edit

(K)

10 GHz & Up Log W1GHZ

FN24	FN34	FN44	FN54	FN64
FN23	FN33	FN43	FN53	FN63
FN22	FN32	FN42	FN52	FN62
FN21	FN31	FN41	FN51	FN61
FN20	FN30	FN40	FN50	FN60

Other grid: DX FN44IG

(L)

10 GHz & Up Log W1GHZ

Current grid is FN42BL
To FN44IG Distance 205 km
Bearing = 13 Degrees True
Reverse = 194 Degrees True

CALL:

New Call K1LPS

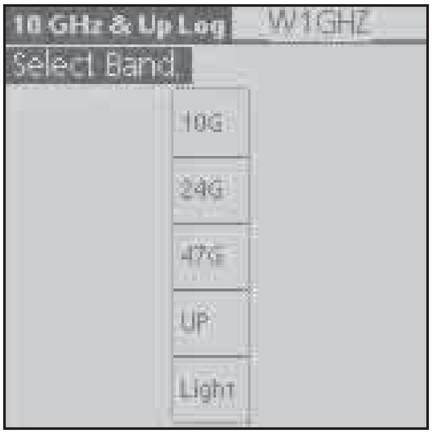
However, for bands above 47 GHz, text entry is required—stations on those bands are probably willing to tolerate the extra effort!

Once the band has been selected, we are ready to log the contact. The data is displayed for confirmation, as shown in Fig 5. Our final menu is to write the QSO to the log: "Yes", "No" and "Edit" are the choices. The first two are obvious, while "Edit" leads to still another menu, shown in Fig 6; each choice allows re-entry of the appropriate field, followed by another chance for confirmation. When the QSO is written, the current date and UTC are automatically added.

A random contact can thus be logged with three taps of the stylus plus text entry of the call and last two digits of the grid square. Contacts based on the previous one are even easier to enter; the best case, running the bands with a station, takes four taps per QSO.

Example

To see how easy logging has become, look at Fig 7; it is a cartoon sequence of log screens for several successive (hypothetical) contacts. Microwave contacts usually begin with some liaison on a lower band, usually 2 meters. W1AIM has informed me that he is portable in grid FN44ig; this is Mt Washington, New Hampshire, so he is probably cold and wet and will appreciate my logging the contact quickly! I look at the grid map on my Palm and tap on FN44 (I've added the large cross to the picture to show the location of a tap). Up pops a text-input window, shown in the next panel

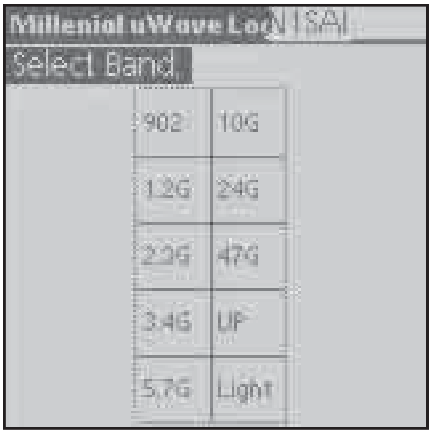


10 GHz & Up Log W1GHZ

Select Band.

10G
24G
47G
UP
Light

Fig 8—The Band Select screen for the ARRL 10-GHz and Up Cumulative Contest.



Millenial uWave Lo N1SAI

Select Band.

902	10G
1.2G	24G
2.3G	47G
3.4G	UP
5.7G	Light

Fig 9—The Band Select screen for the NEWS Millenial Cumulative Microwave Contest.

Fig 10—The Palm-V menu screen showing all ham applications on W1GHZ's Palm V.

Fig 11—W1GHZ operating the 2001 ARRL 10 GHz & Up Cumulative Contest from the top of Mt. Wachusett, Massachusetts, and logging on his Palm V. Photo by George Caswell, W1ME

Fig 12—W1GHZ operating the 2002 ARRL 10 GHz & Up Cumulative Contest from Block Island, Massachusetts, and logging on his PALM V.

(B). I add the "ig" sub-square to the grid using a stylus to write in the text-input box at the bottom of the screen (not shown in these cartoons). Then I tap on "OK," and the Palm calculates the beam headings and distance, shown in the next panel (C) and prompts for a call.

Once we have completed a successful exchange, I enter the call in the text-input box and tap "OK." A band menu pops up; I tap "10G" and the entry is complete. The information is displayed for a final check, shown in the fifth panel (E). If I tap "Yes," then the QSO is written to the log; if I made a mistake, a quick tap on "Edit" allows on-the-spot correction.

While I am logging this contact, another station is "tail ending" on 10 GHz. Since dish antennas are very sharp, he is probably in the same grid. After we complete a successful exchange, I can log this QSO quickly: The previous grid is in the lower right corner of the grid map, shown in panel F, so I can repeat it with a single tap. The next screen (panel G) again displays the bearing and distance, and asks if this is a new call—if it were the same call on a different band, a single tap would suffice. However, this is a new call, so I tap "New Call," then enter the call on the next screen (panel H). Then a tap on the band menu (panel I) and a "Yes" for confirmation (panel J) writes this QSO to the log.

Meanwhile, Larry has suggested moving to another band, 24 GHz, while we have the antennas aligned on each other. Assuming this QSO is also successful, all the information is the same except for the band, so it can be logged with one tap on each of four successive screens, each shown in a panel.

The cartoon sequence shows how easy logging can be: three QSOs logged with 14 taps of the stylus, one per screen, and only 12 characters actually entered. With a little practice, it becomes easy enough that logging doesn't require your full attention. This is important, since there are a few other things to do simultaneously: copy the contest exchanges, listen for talk-back on the liaison radio, keep stuff from blowing away and answer questions from the tourists. The last is important—we are also public relations ambassadors for Amateur Radio!

Other Versions

The previous screen examples are for a general-purpose VHF+ log, not for any specific contest. I have made four other versions for specific contest logging:

10G & Up: for the ARRL 10-GHz-and-Up Cumulative Contest. The major difference is the limited choice of bands, as shown in Fig 8.

Log 10GHz: for 10-GHz-only operation in the ARRL 10-GHz-and-Up Cumulative Contest, since most operators today operate on a single band. The band is logged automatically.

Millenial: for the Millennial Cumulative Microwave Contest, sponsored by the N.E.W.S. Group, a yearlong microwave contest. The band menu for the microwave bands is shown in Fig 9. Contest rules may be found at **www.newsvhf.com**

ARRL VHF: for the ARRL VHF Sweepstakes and QSO Parties, as well as the ARRL UHF Contest. Since only four-digit grid squares are required and VHF antennas aren't as sharp, a slightly different paradigm is used here: The first prompt is for the call, but with an option to run *BD*. A one-tap grid selection is enough for nearby grids except when running *BD*—the additional digits are needed for accurate calculation.

The Palm menu screen for my ham applications is shown in Fig 10, with icons for all the different versions of logging, *BD_Palm*, and two shareware applications, *RiseSet* and *COMPASS*, which are described below. A tap on an icon will start the program. Figs 11 and 12 show me enjoying the fruits of my labor.

Limitations

These logging programs are aimed at roving and contest operations where small size and convenience are more valuable than the power of a laptop computer and the speed of keyboard entry. One hidden limit is the maximum size of a Palm memo-pad record—roughly equivalent to 60 QSOs. In some parts of the country, this would be perfectly adequate, but not in high-activity regions.

There are at least two solutions:

Occasionally renaming the memo pad record and starting a new one.

The name is the first line of the record; the logging programs use your call as the record name. Since each QSO includes date and time, it is easy to reassemble later.

There are several shareware memo pad replacements, written by users who were unhappy with this limitation. I haven't tried any of them.

Desktop Processing

After the Palm has been "hot-synced" with a desktop system, the memo-pad record (or records) may be exported to a text file and processed into an ARRL-format log for submission. I process the logs using a *Perl* script, *palm2arl.pl*. *Perl* (*Practical Extraction and Report Language*) is a simple interpreted language particularly suited for processing and formatting text. Programs are called scripts. I learned it from *Perl for Dummies*;[4] if you look at the script, you can probably tell. A free *Perl* interpreter is available from **www.activestate.com**. You could do the same job in *BASIC*, or manually for small logs using just a text editor.

More recently, log submission for most contests is required to be in the Cabrillo format. I wrote another *Perl* script, *arrl2cab.pl*, to convert log files in the old ARRL format into Cabrillo format. The next time I make improvements to the logging program, I'll probably add a Cabrillo output to the *palm2arl.pl* script.

More Information

The software described above is free for amateur use and available from the W1GHZ 10-GHz home page at **www.w1ghz.org**. The Palm software requires the *PocketC* compiler, which has a shareware license with a modest fee. However, the freeware *PocketC* run-time only version, which allows you to run the programs but not compile them, is included in with the logging software in POCKETLOG.ZIP. For more information on Palm computing, start with **www.palm.com**.

For more shareware, there are links from both of the above URLs that lead to thousands of choices. One potentially useful bit of shareware is *COMPASS*, a "sun compass"—point the Palm at the sun and it indicates which direction is north. Unfortunately, the sun rarely appears during microwave activity! Another good one is *RiseSet*, which calculates the moon and sun positions for any longitude and latitude. I used *RiseSet* to make an EME contact with W5UN on 144 MHz.

The range of shareware is wide and eclectic. One program allows you to use the IR port as a TV remote control. For times when I am left waiting somewhere, there are a number of electronic books available for download. One that I enjoyed reading recently was Homer's *Iliad*.

There are also many Palm accessories available. For instance, both a GPS and an electronic compass are available which attach directly to the Palm serial interface; unfortunately, only one can be attached at a time. I may eventually interface the logging programs to a GPS for mobile roving.

Other Platforms

Why the Palm organizer and not one of the others on the market? Some, like the Handspring Visor, the Sony Clié, and some fancy cellphones, run the Palm OS (operating system) so they are interchangeable and will run these programs, as will any Palm model. Together, the Palm OS devices account for the majority of PDA devices sold to date.

Some of the other PDAs run *Windows CE* and have fancy color screens. The color screens eat batteries, and it would hardly be slanderous to equate "Windows" with "big and slow." Still, if you wish to have a go at one, sources for the programs are included, and there is a version of the *PocketC* compiler available for *Windows CE*.

There are also fancier compilers and development environments available. Rex, KK6MK, used the GNU *C* compiler (**www.gnu.org**) to develop *GL*, a more sophisticated Palm replacement for *BD*. See **users.rcn.com/rexa/Projects/Projects.html** for details.

Notes

[1] *PocketC* is available at **www.orbworks.com**.

[2] You can download this package from the ARRL Web **www.arrl.org/qexfiles/**. Look for 0303Logs.zip.

[3] N. Rhodes & J. McKeehan, *Palm Programming: The Developer's Guide* (Cambridge, Massachusetts: O'Reilly, 1998).

[4] P. Hoffman, Perl *for Dummies,* 2nd Edition (Foster City, California: IDG Books, 1998).

Remote Tuning of a Low-Frequency Loop Antenna

Here is a different method for remotely tuning a receiving loop antenna using variable-inductance toroidal coils.

By Robert Kavanagh, VE3OSZ

Loop antennas for reception of signals at frequencies in the broadcast band and below are usually tuned to the desired frequency by means of a variable capacitor located at the loop. Changing the tuning for a new frequency is inconvenient when the loop is located at some distance from the receiver. Some means of remotely tuning the loop from the receiver location is desirable.

Possible techniques to accomplish this remote tuning are to use varactor diodes or to employ a motor-driven capacitor. Each of these methods has its limitations. This article describes an alternative technique that may be preferable in some circumstances.

This work was stimulated by the *Tech Notes* article in *QEX* for May/June 2001.[1] That article described several applications for tunable toroids that had previously been presented in *RadComm* columns.

The basic principle involved in a tunable toroid is that the relative permeability of the core will change as the magnetic field intensity in the core is varied. This is a consequence of the saturation of the core as the field increases. This phenomenon is unwelcome to most users of toroids, who use them to make RF chokes and transformers. In this case, however, the nonlinearity of the saturation curve is highly desirable. If the magnetic field in the core can be made to change in some way, then the inductance of a coil wound on that toroid will also change. By placing that coil in series with the loop antenna and a fixed capacitor, the loop can be made to resonate over a range of frequencies.

Hysteresis Loops of Ferromagnetic Materials

Fig 1 shows the general shape of the hysteresis loop for ferromagnetic materials, such as ferrite and powdered iron, as used in toroidal cores. An unmagnetized core (point 0) has a relatively high initial permeability, which is the slope of the B-H curve beginning at point 0. Once the core is magnetized, the state of the core follows the path 0-1-2-3-4-5-6-1 as the applied magnetic field is increased to maximum at point 1, decreased to zero

849 Maryland Ave
Ottawa, ON K2C 0H9
Canada
ve3osz@rac.ca

[1] Notes appear on page 57.

at point 2, reversed and increased to a negative maximum at point 4, decreased back to zero at point 5, reversed again and increased back to a positive maximum at point 1. The state of the core will not return to point 0 unless the core is demagnetized in some manner.

When a coil on a ferromagnetic toroid carries both a direct current and a small-amplitude alternating current, a minor hysteresis loop will be traced out by the alternating current centered about a point determined by the direct current. The average slope of this minor loop determines the incremental permeability, and, hence, the inductance seen by the alternating current. This incremental permeability is less than the initial permeability. As the magnitude of the direct current increases, the average slope of the minor loop decreases resulting in a smaller incremental permeability.[2] Notice that the initial permeability, not the incremental permeability, is the parameter given in the specifications for toroids.

It follows that as the direct current in the coil is increased from zero to some maximum value, the inductance seen by an RF current in the coil will decrease from some maximum initial value to a minimum value. Decreasing the dc back to zero will increase the inductance to approximately its initial value. This variation in the coil inductance, caused by a change in the dc magnetic field in the core, may be used to tune resonant circuits.

The magnetic field in the core carrying a tuning coil can be varied in several ways including:

- Applying an external field created by an electromagnet,
- Passing a direct current through the tuning coil, and
- Passing a direct current through a second coil—a magnetizing coil—wound on the toroid.

Each of these methods can be used to tune a circuit containing the tuning coil. In this work, the second and third methods have been used.

Choice of Core Material

Toroidal cores are available in many different powdered-iron and ferrite materials. The selection of a particular material for use in this particular application involves several factors:

- The maximum and minimum values of inductance required,
- The maximum value of the direct current used to tune the circuit,
- The physical dimensions of the core to be used,
- The importance attached to temperature sensitivity, and
- The simplicity of the tuning technique and of the associated circuit.

Three different materials have been used in these experiments. The advantages and disadvantages of each will be described and the results will be useful, perhaps, to others who may wish to carry out further experiments of this nature.

Powdered-iron cores can be quickly eliminated as candidates for use in this application. The ability to vary the inductance of a coil is crucially dependent upon the magnetic material being *saturable*. Powdered-iron materials saturate at much higher flux densities than is the case with ferrite materials. This property makes them very suitable for certain applications, but means that high magnetic-field intensities would be needed to achieve a useful variation in the inductance of a coil wound on such a core. Only ferrite materials are given further consideration.

Ferrite materials that have been used are the Fair-Rite types 61, 43 and 85.[3] Reasons for the choice of these particular materials are given below.

Single-Coil Tuning Inductors

Initial experiments were carried out using a tuning inductor consisting of only one coil. In this case, the single coil performs two functions: it tunes the loop antenna as its inductance is varied and it carries a direct current whose value controls the permeability of the core. This technique

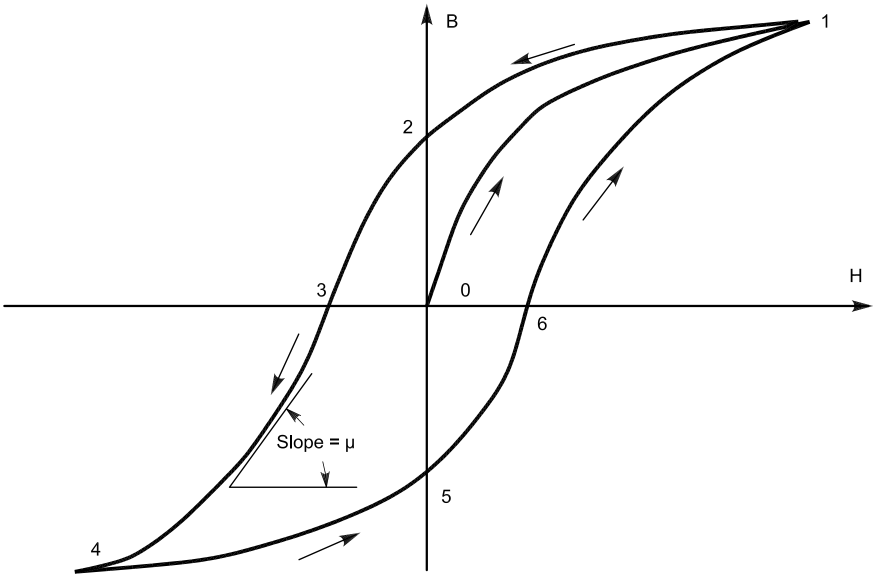


Fig 1—A typical hysteresis loop for ferromagnetic material.

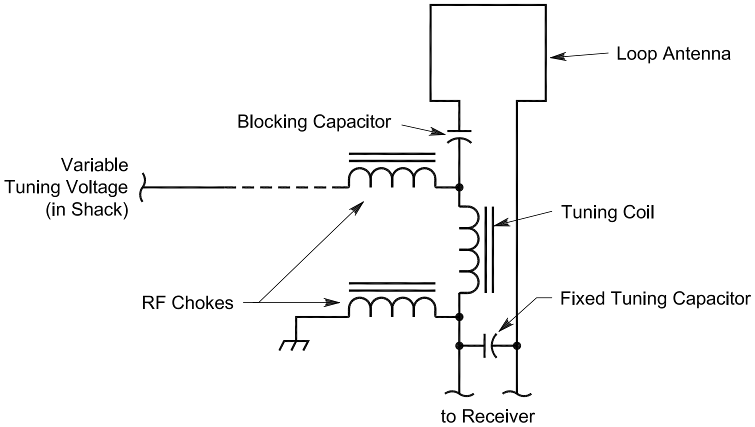


Fig 2—A schematic of the basic loop-tuning system.

has a certain simplicity and elegance but, as will be seen, it does have a disadvantage as compared to the dual-coil method. The basic tuning system using this approach is shown in Fig 2.

The frequency range of interest in this case is about 200-400 kHz. Nondirectional beacons are the target signals. The loop is about one-meter square with three turns. It has an inductance of about 60 µH. The RF chokes are needed to isolate the dc function of the coil from its RF function. In this particular case, 2.5-mH chokes were used.

The ideal material for the toroid core would have a high degree of curvature in the hysteresis loop over small ranges of dc magnetic-field intensity and would be insensitive to temperature changes. The curvature of the hysteresis loop determines the range of incremental permeabilities achievable, hence the possible range in inductance values. A small change in magnetic-field intensity implies need for a relatively small number of ampere-turns created by direct current in the coil. Clearly, the number of turns on the core affects both the inductance of the tuning coil and the magnetic field intensity obtainable for a given maximum value of direct current in the coil. Relative insensitivity to temperature change implies that the inductance of the tuning coil would not change appreciably as the ambient temperature was changed. In fact, ferrite materials are much more sensitive to temperature changes than is the case with powdered-iron materials. In general, high initial permeability results in high temperature sensitivity. As is to be expected, compromise is needed when selecting a material.

The first material tested was type-61 ferrite. This material was selected because the hysteresis loop (as shown in the Fair-Rite Soft Ferrites catalog)[4] shows a fair degree of curvature and this means that a substantial change in incremental permeability should be obtainable. The temperature sensitivity of the permeability is relatively low.

Determination of the core size and number of turns for the coil is not simple. For a given size core made of a particular material the inductance factor, A_L, (the inductance, in nanohenries, for a single turn coil), is known. Nevertheless, this is based upon the value of the initial permeability and not the incremental permeability. As previously mentioned, the incremental permeability will be less than the initial permeability. Therefore, the inductance seen by an RF current in the coil will be somewhat less than the value predicted by the factor A_L for a given number of turns.

In practice, the best approach seems to be an experimental one: Wind a coil on a test core with a reasonable number of turns and try it. In the present experiment, an FT50-61 was tried with several coils. Eventually, 60 turns was chosen. It resulted in the tuning characteristic shown in Fig 3. In this case, the fixed tuning capacitor was 1594 pF. The loop antenna could be tuned over a range from 215 to 390 kHz for a current range from 0 to 835 mA.

To extend the tuning range by increasing the maximum frequency, either a higher current or more turns could be used to increase the magnetic field in the core. A higher current might not be acceptable, though. Moreover, more turns would increase the inductance of the coil and so tend to counteract that desired increase in frequency.

Another approach to extending the tuning range is to use a core material having a greater change in the incremental permeability for a given range of current. To that end, a core made of

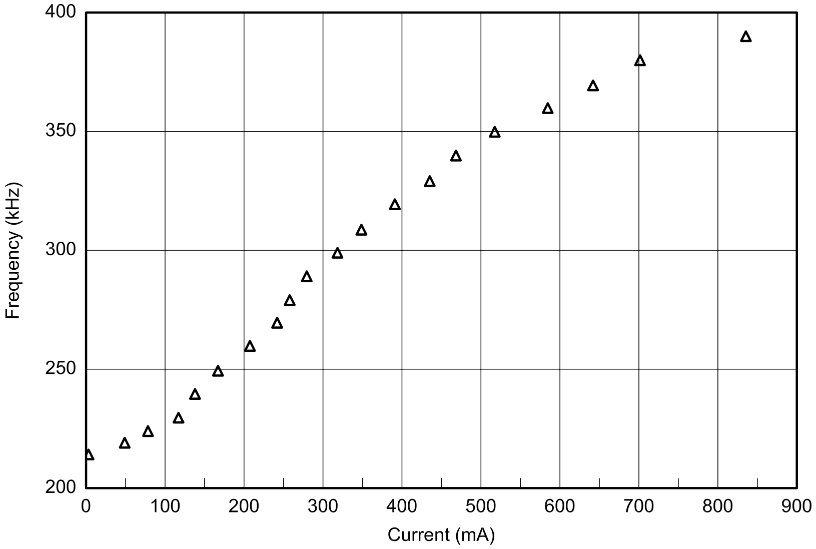


Fig 3—Loop tuning characteristic (resonant frequency versus current) for a FT50-61 core with a 60-turn coil.

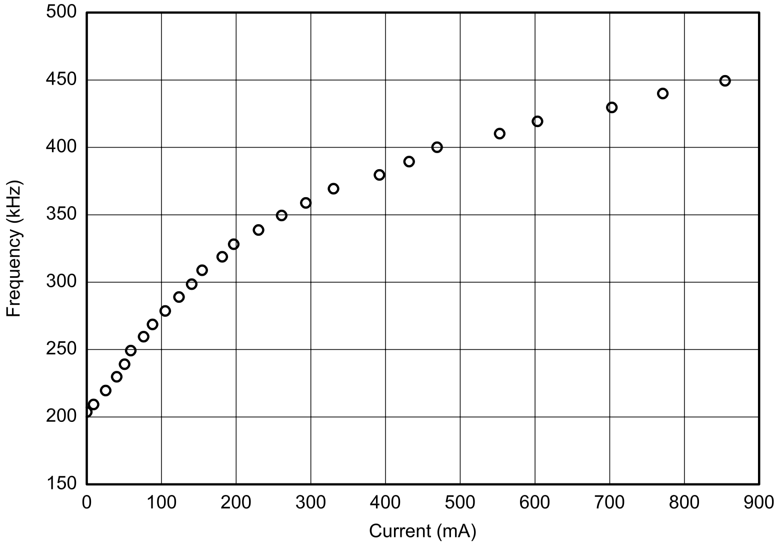


Fig 4—Loop tuning characteristic (resonant frequency versus current) for a FT50-43 core with a 43-turn coil.

type-43 material was next tried. This material has a higher initial permeability (μ = 850) than type-61 material (μ = 125). Inspection of the hysteresis loop for this material suggested that a wider range of incremental permeabilities might be obtainable for a smaller change in magnetic field than was the case for type-61 material. After several trials, an FT50-43 core with 43 turns was found to give useful results. The loop-tuning characteristic for this coil is shown in Fig 4. The fixed tuning capacitor was again 1594 pF. As may be seen, the tuning-frequency range was increased for about the same maximum current.

This tuning coil had two disadvantages. These were:

1. The maximum current was considered too large. A maximum current of the order of 100 mA would be preferable.
2. The material is very sensitive to temperature change. The initial permeability increases by about 50% as the temperature increase from 0 to 25°C. That means that the inductance would vary by about the same amount.

At this point, I decided to try a very different material, type 85. This is a "square-loop" material intended for switching and magnetic amplifier applications. The reason for considering this material was a big change in the slope of the hysteresis loop for a relatively small change in the magnetic field. I expected that the incremental permeability would vary in a similar manner as the direct current changed. The initial permeability is 900. The temperature sensitivity is about one third of that for type-43 material. The hysteresis loop is wide compared with those of type-61 and -43 materials, and I recognized that this property could lead to unusual tuning characteristics.

The cores used were Fair-Rite part number 5985001101.[5] The inductance of a 40-turn coil wound on one of these cores was measured at 400 kHz using an RF impedance bridge. The results obtained for a range of direct currents are shown in Fig 5. The numbered points correspond to those on Fig 1. The arrows show the manner in which the current was varied. Once the core has been magnetized (that is, after state 1 has been reached), about a 3:1 inductance ratio could be obtained for a current variation of ±45 mA. As would be expected, this kind of variation in inductance gave rise to a very interesting tuning characteristic. This is shown in Fig 6. The fixed tuning capacitor in this case was 732 pF. The numbered points correspond with those in Fig 5.

This "butterfly diagram" shows that it was possible to tune the loop antenna over a range from about 260 to 450 kHz for a relatively small current range of ± 90 mA. In this respect, this tuning coil is much superior to those previously used. The drawback to this coil was the hysteresis effect. In tuning the loop from the lowest to the highest frequency, one can move along the path from point 6 to point 1, varying the current from +18 to +90 mA. Alternatively, one could follow the path from point 3 to point 4 with a reversed current. To tune the loop in the opposite direction, from the highest to the lowest frequency, one could follow the path either from point 1 to point 3 or from point 4 to point 6. Nevertheless, it is clear that the current must be reversed at point 2 or 5, as the case may be. Another complication is that tuning must always be in one direction, either increasing or decreasing frequency. Any attempt to back up at some point results in the state of the coil moving on to a different hysteresis loop and, hence, a different tuning characteristic. It is possible to make "short-cuts" in traversing the tuning characteristic. Jumps from points 1 to 4 (and vice versa) and from 3 to 6 (and vice versa) are possible. However, jumps between points 2 and 5 are not possible.

Thus, while this coil provides a reasonable tuning range for a small range of currents, use of the coil in practice is somewhat complex. Nevertheless, one of these coils has been used with good results.

Two-Coil Tuning Inductors

Despite the simplicity of the single-coil tuning inductor, it does have a

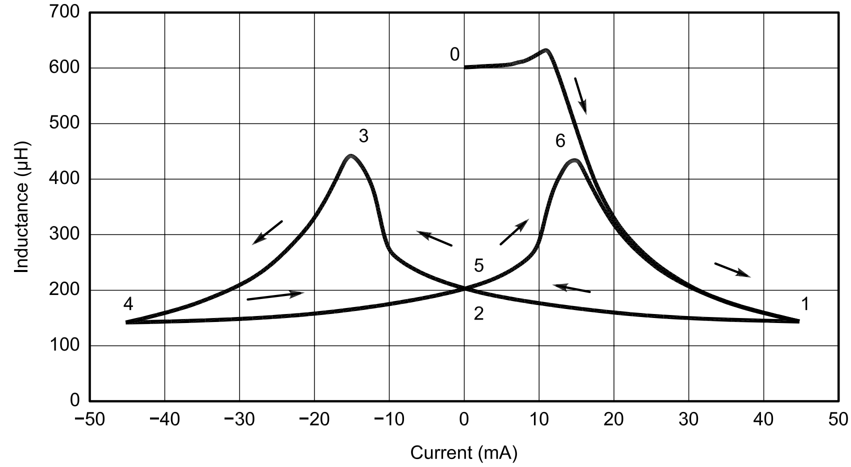


Fig 5—Measured inductance of a 40-turn coil on an 85-material core over a range of dc currents.

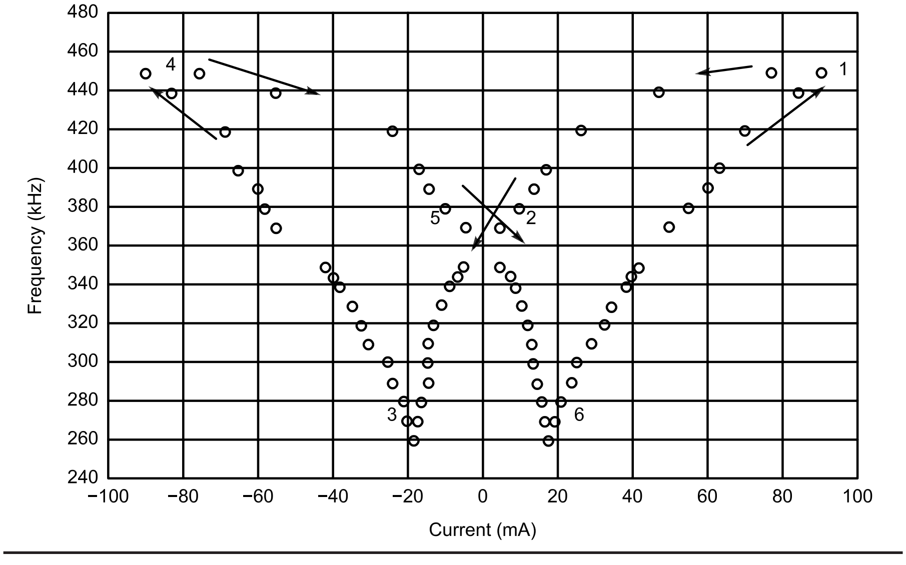


Fig 6—Loop tuning characteristic (resonant frequency versus current) for a type-85 core with a 40-turn coil.

disadvantage in that it is impossible to independently vary the RF and the dc properties of the coil. If the number of turns were increased with the objective of reducing the current needed to produce a given magnetic field, then the inductance of the coil would be greater and this would affect the tuning range.

A core with two windings—one for the tuning function and one for the magnetizing function—will largely avoid this problem. However, the loop circuit must be modified. Fig 7 shows the modified circuit, which now incorporates dc feed through the coaxial cable as well as a preamplifier.[6] At the loop end, the direct current passes through a T-filter consisting of two 2.5-mH RF chokes and a 0.2-µF capacitor. At the receiver end, it passes through a 2.5-mH choke. A 0.2-µF capacitor blocks the dc from reaching the receiver. It was found that the T-filter, rather than simply a single choke, was necessary to prevent RF feedback from the output of the preamplifier, through the magnetizing coil and tuning coil back to the input of the preamplifier. This feedback caused erratic tuning behavior.

The two-coil tuning element consists of a FT50-43 core with 50 turns of #30 AWG wire for the tuning inductor and 110 turns of #28 AWG wire wound on top for the magnetizing coil. The measured inductance of this coil at 400 kHz for a range of values of the magnetizing current is shown in Fig 8. To check for hysteresis effects, three series of measurements were made:

- Current increased from 0 to 100 mA;
- Current then decreased from 100 mA to 0; and finally
- Current increased again from 0 to 100 mA

The results, as seen in Fig 8, show only a small amount of hysteresis in the low-current range. The measured inductance varies over a 6:1 range, from about 600 µH down to about 100 µH. These measurements were made at a temperature of 19°C. Because the core is made of type-43 material, these values can be expected to change as the ambient temperature varies. The loop-tuning characteristic for this coil is shown in Fig 9 for two values of the fixed tuning capacitor.

This tuning coil provides (for this particular loop antenna) a resonant-frequency range of about 2:1 for a maximum direct current of from 130 to 170 mA depending upon the size of the fixed tuning capacitor. In this respect, it is a significant improvement over the single-coil inductor of Fig 4, though not as good as the type-85-core coil of Fig 6. It is, of course, significantly easier to use than the type-85 core coil; however, it is more temperature sensitive than the latter. This implies that the tuning characteristic will vary between winter and summer use.

Conclusion

Weighing the advantages against the disadvantages, I decided that the dual-coil inductor was the best choice, and it is now in regular use for tuning my loop antenna. Further experiments using other materials and core sizes could well result in further improvements. The results presented here were obtained in conjunction with use of a small loop antenna having a low inductance. For a larger loop having a greater inductance, the tuning coil

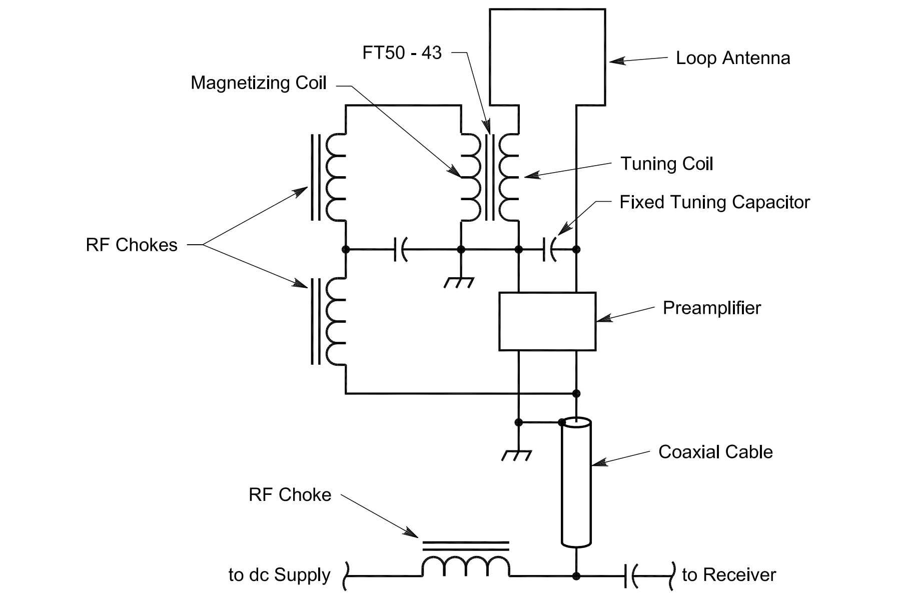


Fig 7—A schematic of a loop-antenna tuning system with a two-coil inductor, a dc feed through coaxial cable and a preamplifier.

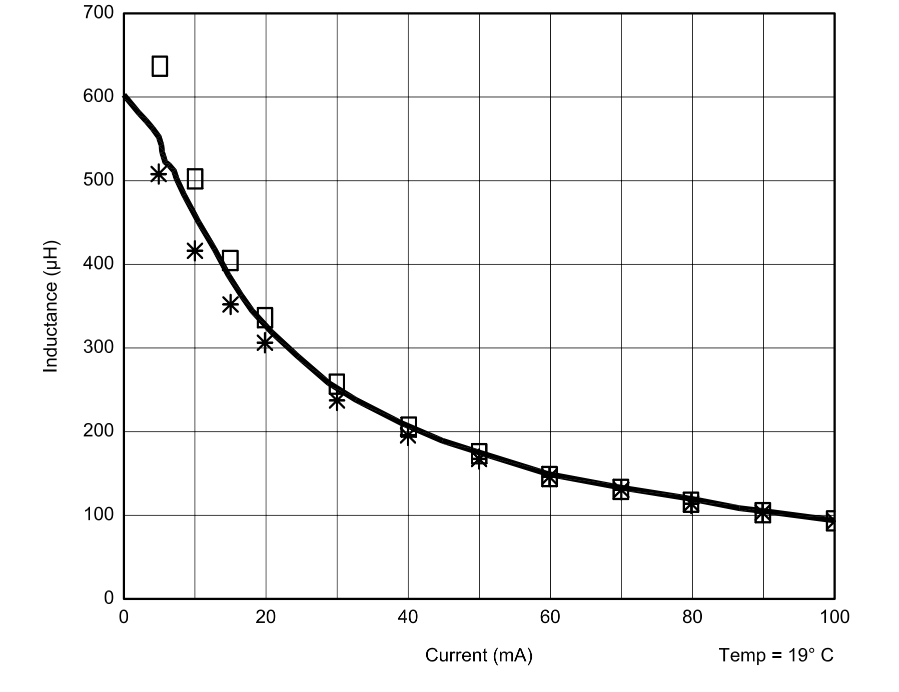


Fig 8—Measured inductance of one winding versus dc in the other winding on an FT50-43 core (at 19°C).

would need to have a larger inductance in order to achieve a similar range of tuning frequencies. This technique may have use in the tuning of receiving loops for reception of 160-meter signals. In any event, I hope that this account of my experiments will interest others in exploring this method for remote tuning of loop antennas.

Notes

[1] Tech Notes, *QEX,* May / June 2001, pp 55-57.

[2] See, for example, Martin A. Plonus, *Applied Electromagnetics* (New York: McGraw-Hill, 1978), p 427.

[3] Fair-Rite Products Corp, PO Box J, One Commercial Row, Wallkill, NY 12589-0288.

[4] *Fair-Rite Soft Ferrites*, 14th Edition.

[5] These cores are available from Elna Magnetics, PO Box 395, 234 Tinker St, Woodstock, NY 12498.

[6] The preamplifier used is essentially the same as that described by K0LR on his Web site **www.cpinternet.com/~lyle/**.

Bob Kavanagh taught and carried out research in electrical engineering for 25 years at the University of Toronto and the University of New Brunswick. He has BS, MS and PhD degrees in electrical engineering. His research field was automatic control and signal processing. For 10 years before retiring, Bob was a Director General at the Natural Sciences and Engineering Research Council of Canada.

Bob was first licensed as VE1YW in 1951 and has held the calls VE3AQO, VE1AXT and VE3OSZ since then. His Amateur Radio interests include DXing (especially on the top band), some contesting, experimenting with circuits and antennas, and propagation. A short article on dawn enhancement of signals was published with VE7DXR and NM7M in the January 2002 issue of The Low Band Monitor. *Bob is an authorized Amateur Radio examiner for Canada's licensing authority, Industry Canada.* □□

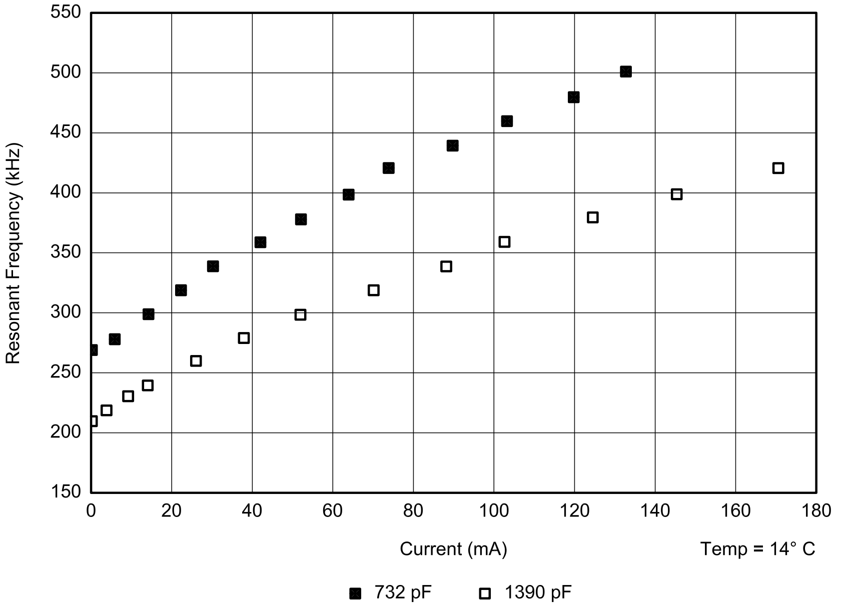


Fig 9—Loop tuning characteristics (resonant frequency versus current) for a FT50-43 core with two windings (at 14°C).

Next Issue in QEX/Communications Quarterly

Among other features in the next *QEX* will be something special for UHF-and-above fans. It is "Microwave Propagation Using the Upper Troposphere," by Bob Larkin, W7PUA; Larry Liljeqvist, W7SZ; and Ernest P. Manly, W7LHL. This absorbing (!) article discusses scattering that occurs between 5 and 16 km above the Earth's surface. The authors give complete details of contacts and measurements made by them between 1 and 10 GHz from locations in the Pacific Northwest. Their work involves rather sophisticated weak-signal techniques and Bob's DSP-10 transceivers.

The group tackled many challenging technical issues, including equipment frequency stability and Doppler shift. Don't miss this one—it adds significantly to our knowledge of atmospherics and microwave signal propagation. □□

RF

By Zack Lau, W1VT

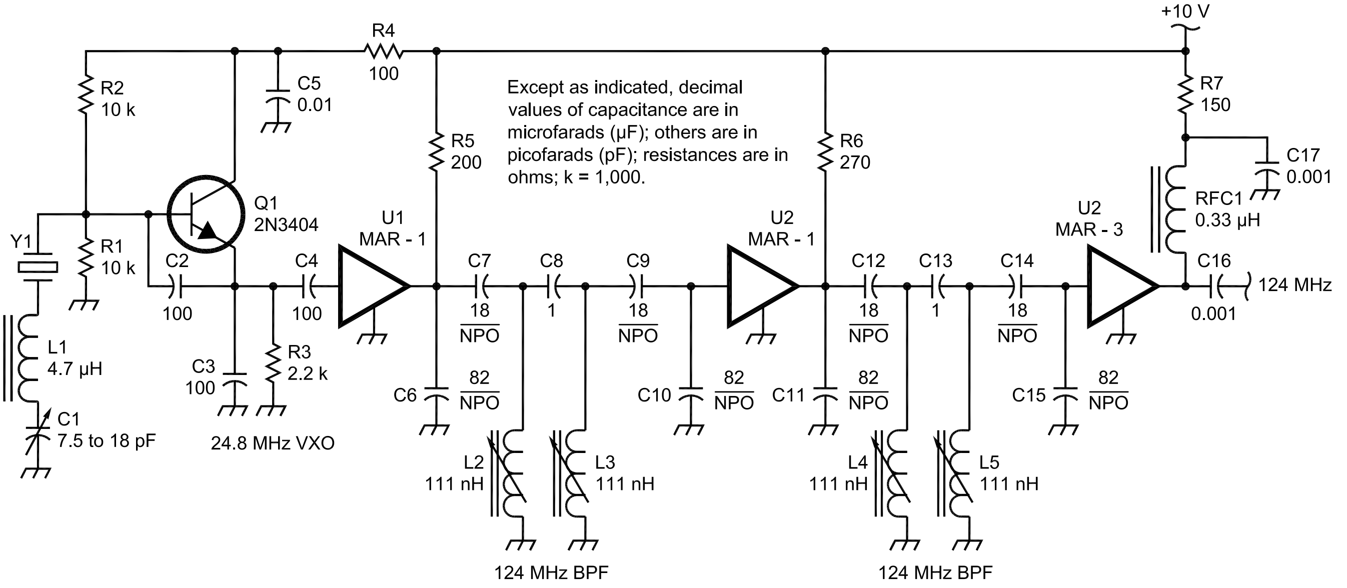
A 2-Meter Receiver

This 2-meter receiver can be used as an IF radio for satellite and microwave work. The circuitry is reasonably straightforward, using 50-Ω input/output-impedance building blocks throughout its RF circuitry. It has a sensitivity of –136 dBm with a 2.4 kHz –6-dB bandwidth. This is a 4-dB noise figure.[1] This is adequate for most receive converters. The two-tone third-order dynamic range is 82 dB—good for a simple design with a moderate noise figure. A receiver is quite useful for experimentation. There is no worry about accidentally transmitting into an expensive and fragile microwave mixer. You can learn a lot about the quality of a microwave signal just by listening to it with a good SSB/CW receiver.

The heart of any SSB/CW receiver is the local oscillator. The schematic is shown in Fig 1. It needs to be stable and reasonably free of spurious signals and noise. Minor imperfections that wouldn't be noticed in other applications become quite apparent when it's used in a radio. Cheap radios often omit SSB and CW modes because of low-quality oscillators, rather than the cost of adding a suitable product detector and beat-frequency oscillator. I used a 24.8-MHz Colpitts variable crystal oscillator or VXO multiplied by five to generate the 124-MHz local oscillator signal. A disadvantage of a VXO is the difficulty of adding a fixed offset for incremental tuning, an important feature in transceiver applications.

After some experimentation with different inductors, a swing of 47 kHz was obtained with a molded 4.7-µH series inductor and a 24.81-MHz crystal, for a total tuning range of 235 kHz. However, most of the tuning range was outside the 2-meter amateur band, from 143.786 to 144.024 MHz. I ended up using a 24.83-MHz crystal with a 2.2-µH inductor, which provided a tuning range from 144.083 to 144.180 MHz. Based on these measurements, a 24.85-MHz crystal may be a good choice for covering 144.0 to 144.200-MHz with a 4.7-µH-molded inductor. Some experimentation is usually needed to find the best inductor for a given crystal. C1 was the best available choice out of my junk box—it has a built-in Jackson Brothers reduction drive. You probably want to explore surplus sources and choose the best available option. You can get a lot of tuning range if the capacitor has a low minimum value, even without the series inductor. In this case, the oscillator will typically tune above the marked crystal frequency.

This can be an expensive approach if you need to buy custom crystals to cover a particular frequency range. You can cover 145 MHz inexpensively. An inexpensive 25-MHz computer clock crystal can be multiplied to 125 MHz. 145 MHz is easily covered with a 20-MHz IF and a 125-MHz local oscillator. A net series inductance can be used to lower the frequency of



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

Fig 1—124-MHz VXO schematic.
C1—7.5 to 18-pF tuning capacitor (see text).
L1—4.7-µH molded inductor (Mouser 43LS476).
L2-L5—7-mm tunable coil, 0.108-µH nominal inductance value (Digi-Key TK2804-ND; Toko E528NAS-100075).
RFC1—0.33-µH molded inductor (Mouser 43LS337).
U1, U2—Mini Circuits MAR-1.
U3—Mini Circuits MAR-3.
Y1—24.85-MHz (see text, International Crystal Manufacturing #435263).

[1] Notes appear on page 61.

225 Main St
Newington, CT 06111-1494
zlau@arrl.org

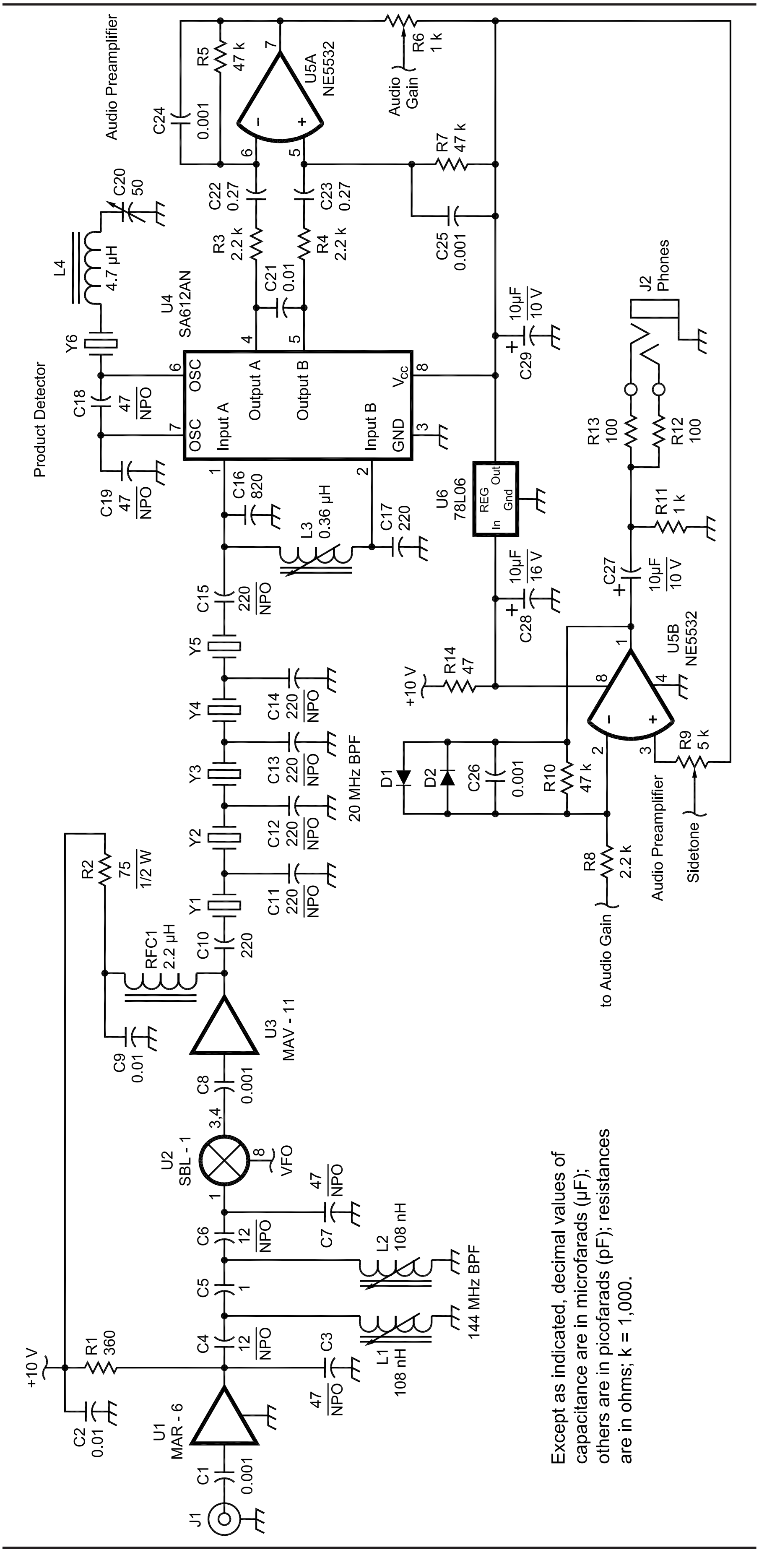
oscillation; you can raise the frequency with series capacitance.

Instead of sometimes scarce air/variable tuning capacitors, you may want to investigate a compression-trimmer design by Lew Smith, N7KSB.[2] He uses a 1/4-20 threaded rod to vary the spacing between two capacitor plates—pushing the plates together for more capacitance.

The 24.8-MHz oscillator needs to be multiplied by five to generate a 124-MHz signal that drives the receive mixer. I used a Mini Circuits silicon monolithic-microwave integrated circuit. The MAR-1 MMIC is an inexpensive gain block design to work in 50-Ω systems. The high harmonic content of the oscillator may help its effectiveness. A pair of two-pole band-pass filters is used to clean up the signal, eliminating the unwanted harmonics. The loss of each filter is 1.5 to 3 dB, depending on the quality of the components.

The 124-MHz band-pass filters use shielded tunable coils. It should be possible to use cheaper coils than the Toko coils specified, but some experimentation may be required. LC (inductor/capacitor) technology tends to be quite finicky at VHF, making substitutions more of a gamble. Toroids aren't as practical at VHF, due to the low inductances involved. The filters are designed for 50-Ω input and output impedances. This modular approach simplifies alignment if RF test equipment is available. You can hook the input to a 50-Ω signal generator and look at the output on a 50-Ω power meter or spectrum analyzer. The ideal test instrument is a network analyzer that will simultaneously display return loss and transmission loss. Such a display makes tuning remarkably easy.

The receiver schematic is shown in Fig 2. A Mini Circuits MAR-6 MMIC is used as an inexpensive preamplifier. Its noise figure is about 3 dB. It should be



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

Fig 2—144-MHz receiver schematic
C1—9 to 50-pF ceramic trimmer (Mouser 24AA084).
D1, D2—1N4148 diodes
L1, L2—7-mm tunable coil, 108-nH nominal inductance value.
L3—0.365 µH (originally a Digi-Key TK2729. TK3004 should be an acceptable substitute. Toko E540SNA-13003).
U1—Mini Circuits MAR-6 MMIC.
U2—Mini Circuits SBL-1 doubly balanced diode mixer.
U3—Mini Circuits MAV-11 MMIC.
U5—NE5532 dual op amp.
U4—Phillips SA612AN mixer/oscillator chip. The NE602 is the popular version in ham literature, but is not in current production.
Y1-Y5—20-MHz clock crystals (ECS 200-20-1 20.0MHz Digi-Key X036). The five crystals for the filter should be within a 200 Hz range.

omitted if you are using a receive converter with excessive gain. RF selectivity is obtained with a two-pole bandpass filter, which has about 2 dB of loss. This filter is essential for good performance. It filters out the unwanted image response of the mixer. This image will degrade sensitivity, raising the noise figure by 3 dB. In some cases, it may be desirable to put another filter ahead of the MAR-6 preamplifier, which is easily overloaded by strong signals, such as off the air cell-phone signals or transverter LO leakage. In this case, you may gain a small improvement in sensitivity by minimizing the loss of the input filter. Actually, this is true for most receivers—practical tradeoffs typically dictate the use of compromise filters. A large helical filter, perhaps 1.5×2.5×3.5 inches, can provide less than 0.5 dB of insertion loss, if carefully adjusted. However, the proper adjustment of bandwidth and coupling is difficult without proper test equipment. A 145-MHz CBT-type Toko helical filter is not recommended; 8 dB of loss is excessive for an RF input filter.

It may be possible to detect overloading by monitoring the voltage across R1—a big signal will cause the MMIC to draw more current, lowering the voltage at the output pin. Be a little careful when measuring the voltage: Connecting a big test probe to the output of the MMIC can create an unwanted feedback path. This could cause the amplifier to oscillate. It's more elegant to measure the voltage by connecting a moderately high value resistor to the MMIC output and decoupling it with a bypass capacitor to form an output port. A big test probe can be connected to the decoupled output port. This technique may also be useful when looking for spurious oscillations; it isn't unusual for preamplifiers to oscillate when terminated at both ends with selective filters, which significantly reduce the system sensitivity.

It may be worthwhile to use high-quality porcelain-chip capacitors instead of the 47-pF and 12-pF NP0s. One of their first intended applications is in solid-state VHF amplifiers, where they must handle very high currents at VHF. Thus, they are superior to silver-mica capacitors, which often have wimpy leads more appropriate to high-voltage, low-current circuits. A bonus of using chip capacitors is the availability of fractional-picofarad values, which is quite useful for small coupling capacitors.

The mixer is a standard Mini Circuits SBL-1 doubly balanced diode mixer, which provides good performance with proper port terminations. (It can easily be degraded if you aren't careful.) The IF port is the most sensitive to reflections—terminating it with a sharp crystal filter is a bad idea. The Mini Circuits MAV-11 MMIC is a simple way to obtain a reasonably broadband termination for the IF port of the mixer. The five-pole 20-MHz crystal filter sets the –3 dB receive bandwidth at 2 kHz. The –6-dB bandwidth—a specification more typical for crystal filters—measures 3 kHz. The actual receive bandwidth is narrower, due to audio filtering. The five-element Cohn crystal filter is made out of inexpensive 20-MHz clock crystals. Custom crystals continue to become ever more costly, while cheap clock crystals seem to be getting even less expensive. Some surplus dealers sell them by the box or bag. The crystal frequencies should be matched with a test oscillator; they should all fall within a 200-Hz frequency range.[3]

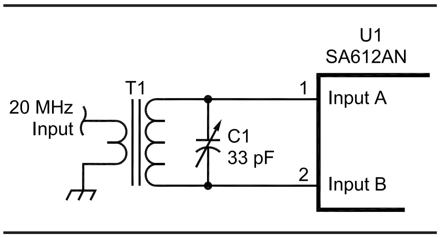


Fig 3—A tuned product-detector input circuit.
C1—33-pF nominal, 9 to 50-pF ceramic trimmer capacitor, surface mount (Mouser 24AA074).
T1—Narrow band transformer wound on T-30-6 iron-powder core. Primary is 21 turns of #26 enameled wire. Secondary is 3 turns of #26 enameled wire over the primary winding. The inductance of the primary is 1.75 µH; the *Q* at 25 MHz is 217.

The detector and carrier oscillator use the popular Phillips SA612AN mixer/oscillator chip. Originally, the SA612AN input circuit was a tuned toroidal transformer similar to that used in my QRP Three-Bander, as shown in Fig 3.[4] The transformer has just three turns on the secondary, compared to two on the original, which was a compromise to cover 18 to 25 MHz. Accounting for the SA612's input capacitance of 3 pF, about 33 pF of capacitance is needed to tune the input circuit. This was replaced with a now-obsolete variable Toko coil and a fixed capacitor, to simplify construction. The Digi-Key TK3004 looks like an acceptable substitute for "toroid-phobics."

A disadvantage of the superheterodyne receiver, compared with the simpler direct-conversion receiver, is the difficulty of properly setting the beat-frequency oscillator. The best solution I've seen was developed to aid K2 builders: use an audio spectrograph and a white noise source to align the BFO. Software is available to convert a PC and sound card into an audio spectrograph. For details, visit the "K2 RX Filter Adjustment" section of the Elecraft Web site: **www.elecraft.com**.

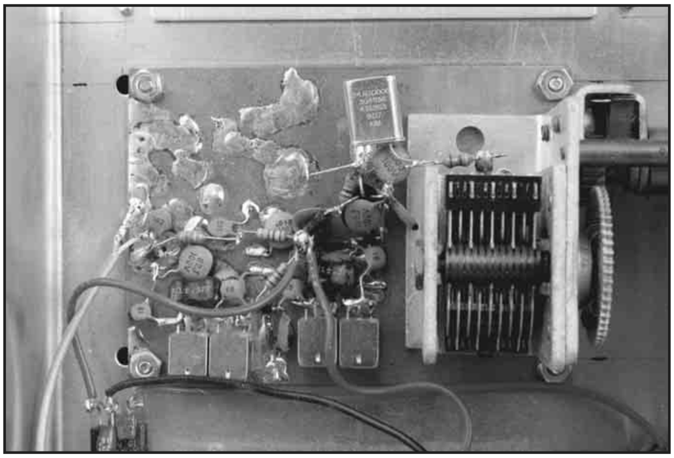


Fig 4—124-MHz VXO used in the receiver. It uses a MMIC multiplier.

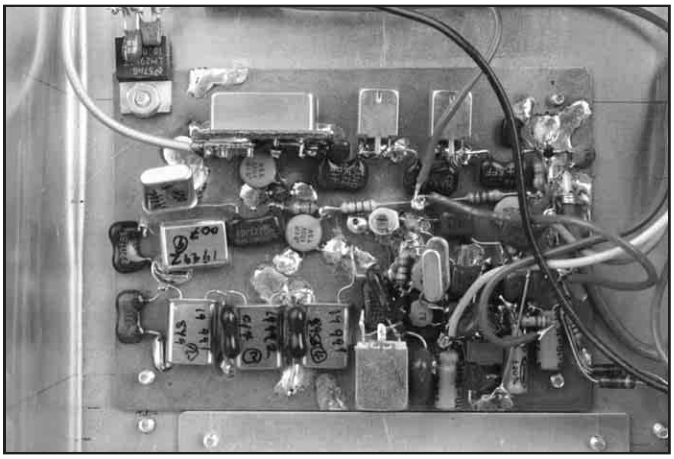


Fig 5—The MMIC preamplifier, 144-MHz band-pass filter, SBL-1 mixer, crystal filter, NE602 product detector and audio amplifiers. The 10-V regulator is also visible.

A solid-state HF noise source will work just fine, if fed directly into the input of the crystal filter.

The audio preamplifier after the detector is balanced. This reduces AM detection and hum. The other half of the audio chip provides headphone-level output. An easy way to obtain speaker-level output is to connect an amplified computer speaker. It may be difficult to add another audio stage if you aren't careful, although this design does have more common-mode rejection than most. High-current ground paths can easily induce unwanted feedback into audio circuits. Many beginners have difficulty grasping exactly how poor grounds affect circuit performance. Some designers suggest using separate regulated supplies for the audio-amplifier stages. The Phillips NE5532 isn't just another op amp in this application—it has the low-output impedance needed for driving headphones without distortion. D1 and D2 provide audio clipping—reducing excessively strong signals. They were added after the photographs (Figs 4 and 5) were taken.

Resistor R11 prevents a dc voltage from building up on the phone jack, to prevent an audible pop or click from occurring when headphones are plugged into the jack. R12 and R13 not only attenuate the signal to appropriate headphone levels, but allow either stereo or mono headphones to be used with a single headphone jack.

Fig 6 is a simple 10-V low-dropout regulator using an LM2940T-10.0. C1 should be nonpolarized to take advantage of the reverse-polarity feature of this IC. Reverse polarity protection is always a good idea for amateur equipment, which is often run from batteries. You never know when something will be re-wired to a different power connector, perhaps incorrectly. It is designed to handle an input voltage between 10.5 and 20 V. This is adequate for most applications. It can handle even more voltage with a good heat sink.

If more receive sensitivity or dynamic range is desired, the place to start is a better preamplifier than the MAR-6 MMIC, as it was chosen for simplicity over performance. Next, it may be worthwhile to investigate high-level mixers sold by Mini-Circuits. The FET ring HJK-3H looks very good, but measurements indicate a need for high-side injection (164 MHz). Be prepared to experiment—it often takes much work to actually achieve the advertised intercept point. Very subtle circuit changes can degrade the intercept point of the mixer, reducing the dynamic range of the receiver.

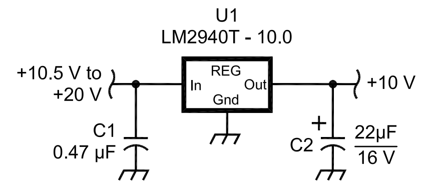


Fig 6—10-V low-dropout regulator.
C1—22 µF, 16 V tantalum (minimum capacitance value, may be increased in value).
U1—National Semiconductor LM2940T-10.

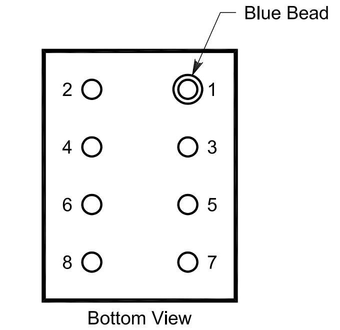


Fig 7—The SBL-1 has an unusual pin pattern. Study it carefully before wiring the part into the circuit.

Construction

The prototype was constructed on unetched circuit board—an excellent copper ground plane. A single-sided board is actually preferred; you can mount it against an aluminum chassis without worrying about electrolytic corrosion.

Assembly is best done in stages: Build a circuit and test it for proper operation before going onto the next stage. I'd build the oscillator first, then the multiplier chain. Many amateur receivers can be used to hear the 24.8-MHz signal—it is close to the 12-meter amateur band. If you have lots of test equipment, it may be a good idea to build the preamplifier first and work your way toward the audio amplifier. This makes it easier to optimize the receiver for dynamic range. However, those with limited test gear will find it easier to start with the audio amplifier, and work toward the preamplifier.

The Mini Circuits SBL-1 can be tricky to wire up—the pinout is very different from the convention associated with typical integrated circuits—carefully study Fig 7. To keep the leads short, I made a little daughter board with eight holes for the mixer pins. After soldering this board perpendicular to the ground plane, I mounted the mixer in place.

The heat sink on the LM2940T-10.0 regulator is optional—the receiver doesn't really draw enough current to make it necessary. However, it is very easy to heat sink, as the metal tab can be grounded. This may allow it to operate at even higher voltages, perhaps up to 26 V, based on datasheet specifications.

Notes

[1] A purist would actually calculate the noise equivalent bandwidth for a five-pole Cohn filter having –3 dB and –6 dB bandwidths of 2 and 3 kHz, respectively.

[2] Lew Smith, N7KSB, "A Compression Capacitor for QRP Transmitters," Hints and Kinks, *QST*, Feb 2002, pp 67-68.

[3] Wes Hayward, "Designing and Building Simple Crystal Filters," *QST*, Jul 1987, pp 27-29.

[4] Zackary Lau, W1VT, "The QRP Three Bander," *QST*, Oct 1989, pp 25-30.

[5] Mini-Circuits World Headquarters, PO Box 350166, Brooklyn, NY 11235; tel 718-934-4500, fax 718-332-4661; **www.minicircuits.com**. □□

Letters to the Editor

An Introduction to Intellectual Property Law (Mar/Apr 2003)

That was an excellent article regarding patent process, but I really believe much has changed in just the last 18 months. I am finding the USPTO Web site to be very helpful and—unlike many for-profit patent-assistance sites—to be receptive to the private inventor. Much obfuscation in the old process is slowly, but surely, being made more humane.

My advice to any potential patent seeker is to know your field of effort—don't try to bluff your way. Research your subject and be able to write intelligently about it. Yes, it probably can be "rocket science," but it is not beyond the reach of any person who has common sense and knows their field of effort.—*Bob Woods, W7IUT, 3043 E Clarendon Ave, Phoenix, AZ 85016-7014;* **bwoodspgs@cox.net**

Dan and Doug,

I read your article on intellectual property in the Mar/Apr issue of *QEX* with much interest. The restrictions created by intellectual property law on use of novel works should be of concern to all amateurs.

The role of intellectual property law is fundamentally to make our works property like any other. One of the most basic principles of property is the right to exclude others from use. I believe, from the perspective of the Amateur Radio community, that this is very regrettable. The modification of our inventiveness and creativity is well suited to the world of commercial exchange. It is not so well suited to the world of amateur experimentation. How many of us would subscribe to *QEX* if we believed we were restricted by intellectual property law from building or modifying the designs we read about in its pages?

Fortunately, there is another major piece of intellectual property law that deals with the conditions under which intellectual property may be used: the law of license. A license is just permission to use a piece of intellectual property. Typically, a license will require a user to pay for use of the property and place restrictions on how the property may be used. I would like to discuss another kind of license that encourages dissemination and improvement of new works.

Although there are many competing articulations of these principles, free licenses allow users access to the underlying intellectual property (not just its commercially usable form), allow modification and allow further dissemination. The Debian Free Software Guidelines (**www.debian.org/social_contract#guidelines**) are one articulation of these principles. The Debian GNU/*Linux* Project, the largest project of its kind, uses these principles to determine whether a program is sufficiently free to be included in the Debian distribution. The guidelines require that the source code for the program must be included and distributed without restriction. They require that users be allowed to redistribute the program. They require that users be allowed to modify and redistribute their modifications. The guidelines also require that the program's license not discriminate against persons groups or fields of endeavor. In other words, the license a program carries may not restrict, for example, commercial or military uses. These principles provide for the widest possible access for the work. Other provisions relate to the relationship between the program and other programs with which it may be distributed and to other matters.

There are now literally thousands of programs available that meet these requirements, including the kernel of the *Linux* operating system. Why would anyone choose such a license for the distribution of their work? Many see it as a form of voluntary repayment for the free software from which they themselves have benefited. Others do it for political reasons. Still others distribute their work freely so that it might reach a broader audience more quickly.

Isaac Newton once said, "If I have seen further, it is by standing on the shoulders of giants." It is critical for those of us who wish to learn at our own pace and from each other that these shoulders remain available for us all to stand on. Many of us see in the most aggressive pursuit of intellectual property the tragic loss of a common intellectual community, a loss just as real as the fencing off of the ancient commons. So, certainly, copyright your work; but I urge all readers to license their work freely.—*Streph Treadway, KB1DJI, 2 Laurel St, Chelsea, MA 02150-1006;* **sbt@ginkwunk.net**

Sir,

Unfortunately, the article "An Introduction to Intellectual Property Law" in the Mar/Apr 2003 *QEX* is full of errors. Concerning patents, the most serious one is that mailing a letter to yourself is totally ineffective to prove priority of invention since the envelope can be steamed open. Also, while it is true that a document describing your invention should be written and witnessed, it must be witnessed by someone who understands the invention. The average notary public is unlikely to satisfy this requirement unless the invention is very simple.

If you are unwilling to disclose the invention to anybody, there is a disclosure document program by the USPTO which costs $10; but be aware that it is no substitute for filing a patent application (see **www.uspto.gov**, in particular MPEP 1706 for details).

On copyrights, in general, it is not true that by giving credit you avoid copyright infringement. However, this may be true when copying from scientific publications, for example, *QST* or *QEX,* if that is the policy of the publication.

On trademarks, the expiration of a patent has nothing to do with a mark becoming generic, but is determined by use. For example, if when people say "Aspirin" they mean only a generic product (the acid in this case), made by Bayer, the mark would still be good; but if when they say "aspirin" they mean that product as made by anybody, the mark has become generic. The latter is obviously the situation today.

The above is somewhat simplified and there are more errors, but hopefully this will prevent serious misinterpretation on the part of the readership.—*Henry Steckler, K2OT, 8D Seabreeze Ave, Milford, CT, 06460;* **HSteckler@pgpatent.com**

Dear Henry,

Thanks for your comments. I think you are right about mailing a letter to yourself. However, the witnessing of documents by a notary public should stand as evidence in any priority dispute, whether or not the witness understood the material. In addition, we do not believe we wrote that by giving credit, one avoids copyright infringement.

We did not write that expiration of a patent directly affects trademark validity. We simply pointed out that since the patent on aspirin had long since expired, many companies were making it and the common use of the term in the popular vernacular is what caused the loss of secondary meaning for Bayer. The term "aspirin" is now completely generic and may be used without fear of trademark violation. We realize the same cannot quite be said about "Xerox" or "Kleenex." We did point out that the confusion of consum-

ers is part of the determination when it comes to brand names.—Doug Smith, KF6DX, QEX *Editor*

A Software-Defined Radio for the Masses, Part 4 (Mar/Apr 2003)

Dear SDR aficionados,

As many of you know, my ISP shut down my free site last week due to traffic overload. Therefore, I have set up a permanent address at **www.flex-radio.com**. Please bookmark this address and pass it on to all your friends.

In addition, there were two inaccurate statements in the sidebar titled, "About Intel Performance Primitives." First, the correct product name is the Intel Integrated Performance Primitives (Intel IPP). Second, one can continue to use the evaluation copy after the 30 days, but all support terminates after the initial period. If a reader wishes to receive support and upgrades, he or she must purchase a license that includes full support and access to upgrades for a one-year service period.—*Gerald Youngblood, AC5OG, 8900 Marybank Dr, Austing, TX 78750;* **ac50g@arrl.net**

A High-Performance Digital-Transceiver Design, Part 2 (Mar/Apr 2003)

I just received my *QEX* for Mar/Apr 2003 and, in reading over the article "A High-Performance Digital-Transceiver Design, Part 2" by Jim Scarlett, I am a bit confused by Fig 2.

This diagram shows no dc connection to the collectors of the two amplifier transistors. T1 and T2 are marked on the diagram as being ADT1-1 and ADT16-6, respectively. However, the parts list calls for these to be six-turn primary, three-turn secondary windings on BN202-61 ferrite cores!

On the diagram, T3 is marked as BN202 (a core) whereas the parts list calls for an ADT1-1 transformer. There may be other similar confusing items I haven't noticed yet; but at first glance, it appears as if some proofreading is missing here. Any comments?—*Ernie Moore, VE3CZZ, 37 Ashgrove Cres, Ottawa, ON K2G 0S1, CANADA;* **ejmoore@trytel.com**

Hi Ernie,

You are correct with regard to the errors in Fig 2. I had sent an update for the schematic and caption, most of which transcribed correctly. However, the part you mention did not.

I hadn't noticed that the transformers had been renumbered in my schematic capture program, and did my updated caption from that schematic. The result is that T1 and T2 in Fig 2 correspond to T3 and T4 in the caption, and vice versa. The transformer descriptions in the caption are correct.

The transformer shown as T2 (T4 in the caption) has the 12-V bias supplied via the center tap (see caption). Capacitors C8 and C10 should also be removed. This way, the amplifier will work as described in the text.—*Jim Scarlett, KD7O, 10625 Highstream Dr, Raleigh, NC 27614;* **jim.scarlett@analog.com**

Hi Doug:

The Mar/Apr 2003 issue of *QEX* is nothing short of excellent! Keep up the good work.

A quick note: Jim Scarlett's article "A High-Performance Digital-Transceiver Design, Part 2" has an error in the RF amplifier schematic. Fig 2 omits the V_{cc} feed, which I would assume would be through the center-tap of the balanced output transformer. Also, the reference designations for the various transformers in Fig 2 need correction; they don't match those reference designations called out in the caption for the same figure.—*Alex Mendelsohn, AI2Q, 164 Sea Rd, Kennebunk, ME 04043-2404;* **ai2q@adelphia.net**

Hi Doug,

I finally got my copy of the March issue! The mail is apparently very slow to Raleigh this time of year. I found a couple of errors in the schematic changes that I missed and therefore didn't correct. These are:

1. In Figs 1 and 2, the resistors on the relay control lines should be 10 Ω, not 1 kΩ. I missed this one. As is, the relays won't switch with 5 V; and if you raise the voltage to switch them, the resistors get very hot.

2. As Alex stated, the output transformer in Fig 2 is wrong. C8 and C10 do not exist, and the output transformer should be center-tapped with +12 V applied. He is also correct on the designators, where T1/T2 and T3/T4 are transposed.

3. In Fig 4, there should be a 0.01-ìF cap between Q3 and R25. This can be omitted if you want a heater instead of a buffer, but it's not recommended. I didn't catch this one in time either.

4. In Fig 5, R3 should be 10 kΩ. I found problems with the bandwidth of the output when using 100 kΩ.

73, Jim Scarlett, KD7O

A High-Level Accessory Front End for the HF Amateur Bands (Mar/Apr 2003)

We have received our airmail *QEX* copies. My father and I both thank you very much. What a kind surprise to see the nice photo on the cover! That is very much appreciated.

We have found one mistake on p 52 in the Fig 6 schematic of the PIN-diode attenuator circuit. The bypass relay must be as follows (our PC-board template is okay): RF In goes to one common terminal of K11 and RF Out goes to the other.

The 100-nF input capacitor goes to the K11A terminal; the output 100-nF capacitor goes to the K11B terminal. Of course the relay position indicated is in the normally closed, OFF, position, so RF is bypassed by the short and the attenuator circuit is outside the RF path.

We now have additional information about the preamplifier performance:

IN3LNC, Claudio has performed an accurate set of tests of our preamplifier alone, and here are the results that fully confirm our preliminary data:

Gain: 12 dB (12.7 dB at 7 MHz, 11.3 dB at 30 MHz, flat within 2 dB over all 30-MHz range).

OIP3: + 43 dBm (with 7L14 by Tektronix and 2× HP8640B, –6 dB hybrid coupler, 10 dB attenuator each leg, the two tones at 20, 50 and 100 kHz spacing and for all three offsets that *OIP3* data has been confirmed).

NF = 3.9 dB (measured with a calibrated noise source at 30 MHz).

1-dB compression point: +24.5 dBm (input +11.5 dBm).

The output impedance was lower than 50 Ω; but in this application, this is not a problem. Performance is not degraded and may be useful; that is, when paralleling two 50-Ω receivers without any 2:1, three-way combiner.

One more thing: I have discovered a very powerful tool to play with changes in receiver gain, noise, IP3, MDS per stage and cumulative data to simulate a complete receiver system without long calculations. Try this professional, nice and freely available software by AGILENT: *AppCAD* Version 2.5.1, download setup251.exe (12.2 MB) from **www.agilent.com/view/appcad**. Once it's installed, choose the menu function "Noise Calc" and enjoy it.

There is more information (including a dBm to dB-μV conversion table) on my Web site **www.qsl.net/ik4auy** (look under Measures & Simulations, an article in Italian for future publication in *ARI* magazine; but the relations are easy to read).—*Sergio Cartoceti, IK4AUY;* **sergio.cartoceti@tin.it**

Name: ham radio, born January, 1968.

Why **ham radio** *(magazine)? The electronics and communications industry is moving forward at a tremendous clip, and so is amateur radio. Single sideband has largely replaced a-m, transistors are taking the place of vacuum tubes, and integrated circuits are finding their way into the ham workshop. The problem today, as it has always been, is to keep the amateur well informed.*—Editor Jim Fisk, W1DTY (SK), from the preview issue of *ham radio* magazine, February, 1968 (last issue published in June, 1990).

hr
focus
on
communications
technology . . .

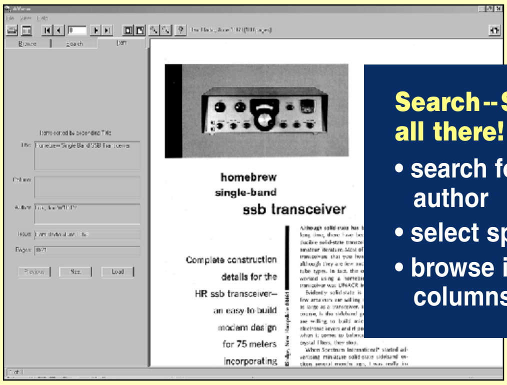
Featuring Ham Radio CD-ROMs!

System Requirements: Pentium or equivalent IBM-compatible PC, and Microsoft Windows™ 95, 98, NT 4.0, Me, or 2000.

Enjoy quick and easy access to back issues of this popular magazine! These CD-ROM sets include high quality black-and-white scanned pages, easily read on your computer screen or printed. All the articles, ads, columns and covers are included.

Readers will enjoy a wealth of material that spanned the gamut of Amateur Radio technical interests: construction projects, theory, antennas, transmitters, receivers, amplifiers, HF through microwaves, test equipment, accessories, FM, SSB, CW, visual and digital modes.

The complete set covers more than 30,000 pages!



homebrew
single-band
ssb transceiver

Complete construction details for the HR ssb transceiver— an easy to build modern design for 75 meters incorporating

Search--Select--Browse—it's all there!

- search for articles by title and author
- select specific year and issue
- browse individual articles and columns

Only $59.95 per set:* **Each set includes four CDs!**

Ham Radio CD-ROM 1968-1976 ARRL Order No. 8381
Ham Radio CD-ROM 1977-1983 ARRL Order No. 8403
Ham Radio CD-ROM 1984-1990 ARRL Order No. 8411

SAVE $30! when you order the complete set:*

All 3 Ham Radio CD-ROM Sets (1968-1990)
ARRL Order No. HRCD **$149.85**

***Shipping/handling fee:** US orders add $5 for one set, plus $1 for each additional set ($10 max, via UPS). International orders add $2.00 to these rates ($12.00 max, via surface delivery). Sales tax is required for orders shipped to CA, CT, VA, and Canada.

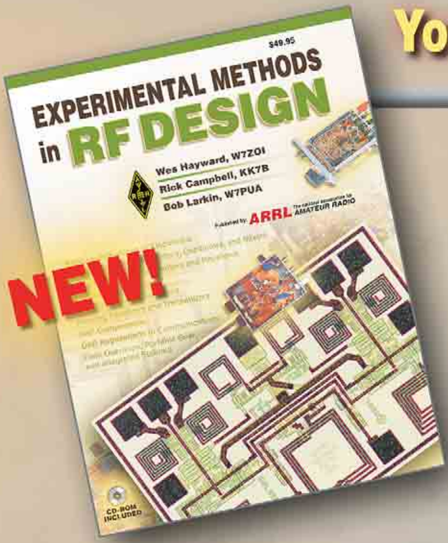
ARRL *The national association for AMATEUR RADIO*

225 Main Street, Newington, CT 06111-1494
tel: 860-594-0355 fax: 860-594-0303

In the US call our toll-free number **1-888-277-5289** 8 AM-8 PM Eastern time Mon.-Fri.
Quick order **www.arrl.org/shop**

ARRL Resources for RF, DSP, and Design

Your Communications Journey Begins Here!


$49.95
EXPERIMENTAL METHODS in RF DESIGN
Wes Hayward, W7ZOI
Rick Campbell, KK7B
Bob Larkin, W7PUA
Published by ARRL The national association for AMATEUR RADIO
NEW!
CD-ROM INCLUDED

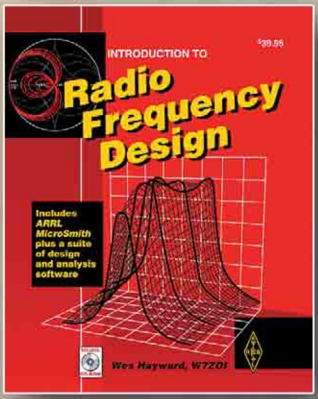
Experimental Methods in RF Design
ARRL Order No. 8799 .... **$49.95***

***Successor to the widely popular* Solid-State Design for the Radio Amateur.**

Immerse yourself in the communications experience by building equipment that contributes to understanding basic concepts and circuits. Explore wide dynamic range, low distortion radio equipment, the use of direct conversion and phasing methods, and digital signal processing. Use the models and discussion to design, build and measure equipment at both the circuit and the system level. Laced with new unpublished projects and illustrated with CW and SSB gear.
CD-ROM included with design software, listings for DSP firmware, and supplementary articles.

Contents:

Basic Investigations in Electronics
Chapters on Amplifiers, Filters, Oscillators, and Mixers
Superheterodyne Transmitters and Receivers
Measurement Equipment
Direct Conversion Receivers
Phasing Receivers and Transmitters
DSP Components
DSP Applications in Communications
Field Operation, Portable Gear
and Integrated Stations


INTRODUCTION TO
$39.95
Radio Frequency Design
Includes ARRL MicroSmith plus a suite of design and analysis software
Wes Hayward, W7ZOI

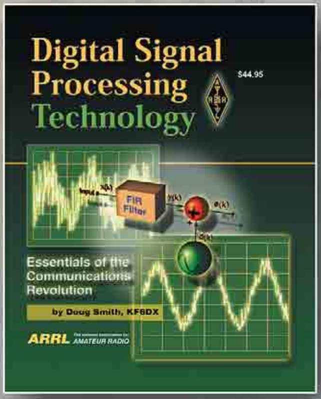
Introduction to Radio Frequency Design,
includes software. ARRL Order No. 4920 .... **$39.95***

The fundamental methods of radio frequency design using mathematics as needed to develop intuition for RF circuits and systems. Simple circuit models are used to prepare you to actually design HF, VHF and UHF equipment. Book includes a CD-ROM with ARRL MicroSmith Smith® Chart simulation software, and a suite of design and analysis software (for IBM PCs and compatibles).

First ARRL Edition, third printing, © 1994-2000.

Contents:

Low Frequency Transistor Models
Filter Basics
Coupled Resonator Filters
Transmission Lines
Two-Port Networks
Practical Amplifiers and Mixers
Oscillators and Frequency Synthesizers
The Receiver: an RF System


Digital Signal Processing Technology
$44.95
FIR Filter
Essentials of the Communications Revolution
by Doug Smith, KF6DX
ARRL The national association for AMATEUR RADIO

Digital Signal Processing Technology—Essentials of the Communications Revolution,
ARRL Order No. 8195 .... **$44.95***

A comprehensive, readable work for anyone interested in Digital Signal Processing (DSP). The book begins with basic concepts, details digital sampling including fundamental and harmonic sampling, aliasing and mechanisms at play in real data converters, digital filter design, mathematics of modulation and demodulation, digital coding methods for speech and noise-reduction techniques, digital transceiver design, and other current topics. Sufficiently analytical for the advanced engineer or experimenter (with a working knowledge of algebra), while simultaneously affording an understandable picture of this exciting technology.

Contents:

Introduction to DSP
Digital Sampling
Computer Representations of Data
Digital Filtering
Analytic Signals and Modulation
Digital Coding Systems for Speech
Direct Digital Synthesis
Interference Reduction
Digital Transceiver Architectures
Hardware for Embedded DSP Systems
DSP System Software
Advanced Topics in DSP...... *and more*

***Shipping and Handling charges apply. Sales Tax is required for orders shipped to CA, CT, VA, and Canada.**

Prices and product availability are subject to change without notice.

ARRL *The national association for* *AMATEUR RADIO*

SHOP DIRECT ***or call for a dealer near you.***
ONLINE WWW.ARRL.ORG/SHOP
ORDER TOLL-FREE 888/277-5289 (US)