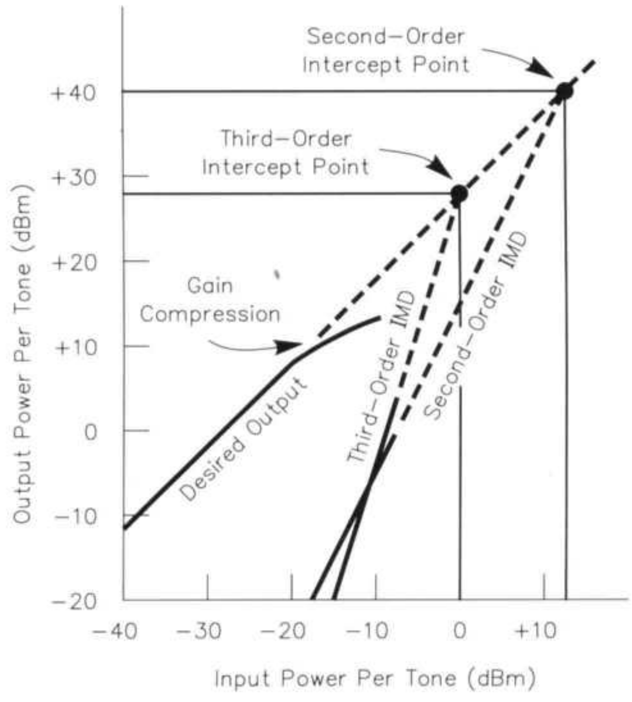
QEX

$1.75

ARRL Experimenter's Exchange

July 1994



Receiver IMD Testing

QEX: The ARRL
Experimenter's Exchange
American Radio Relay League
225 Main Street
Newington, CT USA 06111

QEX (ISSN: 0886-8093) is published monthly by the American Radio Relay League, Newington, CT USA.

Application to mail at second-class postage rates is pending at Hartford, CT.

David Sumner, K1ZZ
Publisher

Jon Bloom, KE3Z
Editor

Lori Weinberg
Assistant Editor

Harold Price, NK6K
Zack Lau, KH6CP
Contributing Editors

Production Department

Mark J. Wilson, AA2Z
Publications Manager

Michelle Bloom, WB1ENT
Production Supervisor

Sue Fagan
Graphic Design Supervisor

Dianna Roy
Senior Production Assistant

Joe Shea
Production Assistant

Advertising Information Contact:

Brad Thomas, KC1EX, *Advertising Manager*
American Radio Relay League
203-667-2494 direct
203-666-1541 ARRL
203-665-7531 fax

Circulation Department

Debra Jahnke, *Manager*
Kathy Fay, N1GZO, *Deputy Manager*
Cathy Stepina, *QEX Circulation*

Offices

225 Main St, Newington, CT 06111-1494 USA
Telephone: 203-666-1541
Telex: 650215-5052 MCI
FAX: 203-665-7531 (24 hour direct line)
Electronic Mail: MCIMAILID: 215-5052
Internet:qex@arrl.org

Subscription rate for 12 issues:

In the US: ARRL Member $12, nonmember $24;

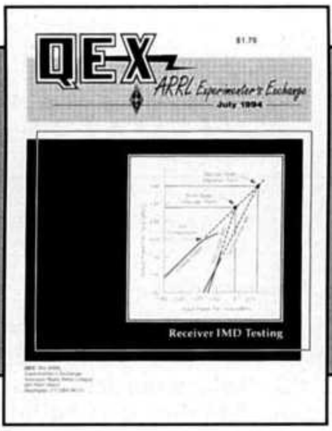
US, Canada and Mexico by First Class Mail:
ARRL Member $25, nonmember $37;

Elsewhere by Airmail:
ARRL Member $48, nonmember $60.

QEX subscription orders, changes of address, and reports of missing or damaged copies may be marked: *QEX* Circulation. Postmaster: Form 3579 requested. Send change of address to: American Radio Relay League, 225 Main St, Newington, CT 06111-1494.

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

Copyright © 1994 by the American Radio Relay League Inc. Material may be excerpted from *QEX* without prior permission provided that the original contributor is credited, and *QEX* is identified as the source.



About the Cover:
Receiver IMD is characterized by the receiver's intercept points.

ISSUE NO. 149

Features

3 **Testing and Calculating Intermodulation Distortion in Receivers**
By Dr. Ulrich L. Rohde, KA2WEU

5 **DSP Voice Frequency Compandor for use in RF Communications**
By John Ash, KB7ONG, Fred Christensen, KA6PNW and Rob Frohne, KL7NA

11 **RF Counter Data Collection and Visualization**
By Brian J. Mork, KA9SNF

Columns

25 **RF**
By Zack Lau, KH6CP/1

July 1994 QEX Advertising Index

AMSAT: 31
Brian Beezley, K6STI: Cov III
ByteMark Corp: 32
Communications Specialists Inc: 32
Down East Microwave: 31
L.L Grace: Cov II
LUCAS Radio/Kangaroo Tabor Software: 32
P.C. Electronics: 31
TwinsLAN ARC: 31
Yaesu: Cov IV
Z Domain Technologies, Inc: 32

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: 203-666-1541 Telex: 650215-5052 MCI.
MCIMAIL (electronic mail system) ID: 215-5052
FAX: 203-665-7531 (24-hour direct line)

Officers

President: GEORGE S. WILSON III, W4OYI
1649 Griffith Ave, Owensboro, KY 42301

Executive Vice President: DAVID SUMNER, K1ZZ

Purpose of *QEX*:

1) provide a medium for the exchange of ideas and information between Amateur Radio experimenters
2) document advanced technical work in the Amateur Radio field
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 correspondence for publication in *QEX* should be marked: Editor, *QEX*.

Both theoretical and practical technical articles are welcomed. Manuscripts should be typed and doubled spaced. Please use the standard ARRL abbreviations found in recent editions of *The ARRL Handbook*. Photos should be glossy, black and white positive prints of good definition and contrast, and should be the same size or larger than the size that is to appear in *QEX*.

Any opinions expressed in *QEX* are those of the authors, not necessarily those of the editor or the League. While we attempt to ensure that all articles are technically valid, authors are expected to defend their own material. Products mentioned in the text are included for your information; no endorsement is implied. The information is believed to be correct, but readers are cautioned to verify availability of the product before sending money to the vendor.

Empirically Speaking

The Amateur's Toolbox

There's a long history within Amateur Radio of adapting the available tools to the activities of the amateur. Whether it is building an antenna support using 2-by-4s and a hand saw, or making PC boards with a copier, amateurs seem always willing to try a low-tech (read: low-cost) way of accomplishing something in preference to buying the right tool for the job. And often, the end result is indistinguishable from the "real thing."

This tradition is being carried forward in the world of software. Amateurs are adapting existing software to the need at hand. One example of this is in this month's article by Brian Mork, KA9SNF. Brian is using a communications program, originally designed to control a modem, and a spreadsheet program to implement a data-collection and analysis system. The result is a simple, easy-to-use system whereby the amateur can collect and analyze frequency-counter data. Sure, you can go out and buy specialized data-collection software to do the same job. But why would you if you have software sitting on the hard disk that will do the job, once a bit of ingenuity is applied?

The reason we bring this up is that we suspect that a lot of amateurs have done similar things with existing software. For example, some software—such as *Mathcad*—is ideal for developing or analyzing circuits. We have published a couple of articles over the past year or so in which *Mathcad* analyses were used to illuminate or discover aspects of circuit operation. What is interesting about such uses, aside from the end-result circuits themselves, is the way software was applied to the problem—without the necessity of buying dedicated (and expensive) circuit analysis software.

Other possibilites may be less well-known. For example, one amateur we know is working on a digital voice system that uses *Linux*, a free unix clone, and a standard PC sound card. Much of the detail work of arranging to get the data into and out of his program is handled by this free operating system software.

There is a lot of software out there—much of it free or low-cost shareware. If you have come up with ways of using low-cost software to solve problems that might not be obvious, *QEX* readers would like to hear about it.

This Month in *QEX*

If you're interested in developing or measuring high-performance receivers, you need to know how to characterize intermodulation distortion performance. "Testing and Calculating Intermodulation Distortion in Receivers," by Ulrich L. Rohde, KA2WEU, supplies the techniques and mathematics you need to know.

Narrow-band voice modulation seems like an anachronism from the 1970s. It never caught on because the implementing technology was too complex and difficult. But with the advent of low-cost DSP hardware, like the Texas Instruments DSP Starter Kit (DSK), that has changed. A "DSP Voice Frequency Compandor for use in RF Communications" turns out to be nothing more than a program that runs on the DSK, as described by John Ash, KB7ONG, Fred Christensen, KA6PNW, and Rob Frohne, KL7NA.

Brian Mork, KA9SNF, shows us how "RF Counter Data Collection and Visualization" can be accomplished easily, using software tools you probably already have.

Wondering how to get on AO-13 Mode S? In this month's "RF" column, Zack Lau, KH6CP/1, provides a complete design for a Mode-S receive converter with a 2-m IF.—*KE3Z, email: jbloom@arrl.org.*

Testing and Calculating Intermodulation Distortion in Receivers

Determining receiver IMD performance isn't difficult, although it does take good test equipment.

by Dr. Ulrich L. Rohde, KA2WEU

In my recent series of articles in *QST*, I've tried to show that, while the receiver third-order IMD performance amateurs have been measuring for years is important, second-order IMD products are just as important—perhaps even more so in this era of wide-bandwidth receiver front ends.[1]

In this article, I'll discuss briefly how to measure IMD in receivers and how to calculate intercept points from those measurements, allowing us to compare the IMD performance of different receivers objectively.

Making the Measurements

Second and third-order IMD can be measured using the setup of Fig 1. The outputs of two signal generators are combined in a 3-dB hybrid coupler. Such couplers are available from various companies, including Synergy Microwave Corporation. The 3-dB coupler should have low insertion loss and should itself produce negligible IMD. The signal generators are adjusted to provide a known signal level at the output of the 3-dB coupler, say –20 dBm for each of the two signals. This combined signal is then fed through a calibrated variable attenuator to the device under test. The shielding of the cables used in this system is important: at least 90 dB of isolation should exist between the high-level signal at the input of the attenuator and the low-level signal delivered to the receiver.

The measurement procedure is simple: adjust the variable attenuator to produce a signal of known level at

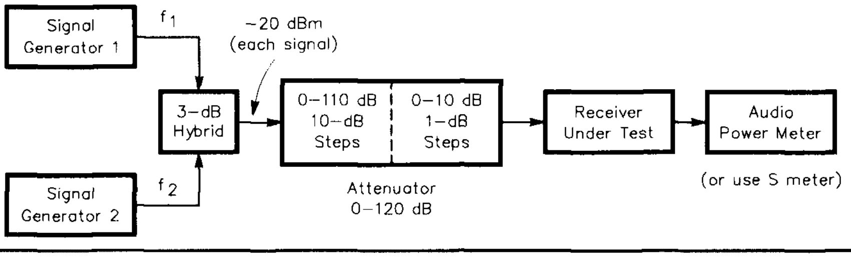


Fig 1—Test setup for measurement of IMD performance. Both signal generators should be types such as HP8640, HP8662, or Rohde & Schwarz SMGU, with phase-noise performance of –140 dBc/Hz or better at 20 kHz from the signal frequency.

[1]Rohde, U. L., "Key Components of Modern Receiver Design—*Part 1*," *QST*, May 1994, *Part 2*, June 1994, *Part 3*, July 1994.

52 Hillcrest Dr
Upper Saddle River, NJ 07458

the frequency of the expected IMD product ($f_1 \pm f_2$ for second-order, $2f_1 - f_2$ or $2f_2 - f_1$ for third-order IMD).

To do this, of course, you have to figure out what equivalent input signal level at the receiver's operating frequency corresponds to the level of the IMD product you are seeing. There are several ways of doing this. One way—the way used by the ARRL Lab in their receiver tests—uses the minimum discernible signal. This is defined as the signal level that produces a 3-dB increase in the receiver audio output power. That is, you measure the receiver output level with no input signal, then insert a signal at the operating frequency and adjust the level of this input signal until the output power is 3 dB greater than the no-signal power. Then, when doing the IMD measurement, you adjust the attenuator of Fig 1 to cause a 3-dB increase in receiver output. The level of the IMD product is then the same as the MDS level you measured.

There are several things I dislike about doing the measurement this way. The problem is that you have to measure noise power. This can be difficult. First, you need an RMS voltmeter or audio power meter to do it at all. Second, the measurement varies with time (it's noise!), making it difficult to nail down a number. And third, there is the question of the audio response of the receiver; its noise output may not be flat across the output spectrum. So I prefer to measure, instead of MDS, a higher reference level. I use the receiver's S meter as a reference. I first determine the input signal level it takes to get an S1 reading. Then, in the IMD measurement, I adjust the attenuator to again give an S1 reading. The level of the IMD product signal is now equal to the level I measured at S1. Note that this technique gives a different IMD level value than the MDS technique. That's OK, though. What we are trying to determine is the *difference* between the level of the signals applied to the receiver input and the level of the IMD product. Our calculations will give the same result whether we measure the IMD product at the MDS level, the S1 level or some other level.

An easy way to make the reference measurement is with the setup of Fig 1. You'll have to switch in a lot of attenuation (make sure you have an attenuator with enough range), but doing it this way keeps all of the possible variations in the measurement fairly constant. And this way, the difference between the reference level and the input level needed to produce the desired IMD product signal level is simply the difference in attenuator settings between the reference and IMD measurements.

Calculating Intercept Points

Once we know the levels of the signals applied to the receiver input and the level of the IMD product, we can easily calculate the intercept point using the following equation:

$$IP_n = \frac{n \bullet P_A - P_{IM_n}}{n-1} \qquad \text{Eq 1}$$

Here, n is the order, P_A is the receiver input power (of one of the input signals), P_{IM_n} is the power of the IMD product signal, and IP_n is the nth-order intercept point. All powers should be in dBm. For second and third-order IMD, Eq 1 results in equations:

$$IP_2 = \frac{2 \bullet P_A - P_{IM_2}}{2-1} \qquad \text{Eq 2}$$

$$IP_3 = \frac{3 \bullet P_A - P_{IM_3}}{3-1} \qquad \text{Eq 3}$$

You can measure higher-order intercept points, too.

Example Measurements

To get a feel for this process it's useful to consider some actual measured values.

The first example is a Rohde & Schwarz model EK085 receiver with digital preselection. For measuring second-order IMD, signals at 6.00 and 8.01 MHz, at –20 dBm each, were applied at the input of the attenuator. The difference in attenuator settings between the reference measurement and the level needed to produce the desired IMD product signal level was found to be 125 dB. The calculation of the second-order IP is then:

$$IP2 = \frac{2(-20\text{ dBm}) - (-20\text{ dBm} - 125\text{ dB})}{2-1}$$
$$= -40\text{ dBm} + 20\text{ dBm} + 125\text{ dB}$$
$$= +105\text{ dBm}$$

For IP_3, we set the signal generators for 0 dBm at the attenuator input, using frequencies of 14.00 and 14.01 MHz. The difference in attenuator settings between the reference and IMD measurements was 80 dB, so:

$$IP3 = \frac{3(0\text{ dBm}) - (0\text{ dBm} - 80\text{ dB})}{3-1}$$
$$= \frac{0\text{ dBm} + 80\text{ dB}}{2} = +40\text{ dBm}$$

We also measured the IP_3 of a Yaesu FT-1000D at the same frequencies, using attenuator-input levels of -10 dBm. A difference in attenuator readings of 80 dB resulted in the calculation:

$$IP3 = \frac{3(-10\text{ dBm}) - (-10\text{ dBm} - 80\text{ dB})}{3-1}$$
$$= \frac{-30\text{ dBm} + 10\text{ dBm} + 80\text{ dB}}{2}$$
$$= \frac{-20\text{ dBm} + 80\text{ dB}}{2} = +30\text{ dBm}$$

Synthesizer Requirements

To be able to make use of high third-order intercept points at these close-in spacings requires a low-noise LO synthesizer. You can estimate the required noise performance of the synthesizer for a given IP_3 value. First, calculate the value of receiver input power that would cause the IMD product to just come out of the noise floor, by solving Eq 1 for P_A, then take the difference between the calculated value of P_A and the noise floor to find the dynamic range. Doing so gives the equation:

$$ID_3 = \frac{2}{3}\left(IP_3 + P_{\min}\right) \qquad \text{Eq 4}$$

Where ID_3 is the third-order IMD dynamic range in dB, and $P_{\min}$ is the noise floor in dBm. Knowing the receiver bandwidth, BW (2400 Hz in this case) and noise figure, NF (8 dB) allows us to calculate the noise floor, $P_{\min}$:

$$P_{\min} = -174\text{ dBm} + 10\log(BW) + NF$$
$$= -174\text{ dBm} + 10\log(2400) + 8$$
$$= -132\text{ dBm}$$

The synthesizer noise should not exceed the noise floor when an input signal is present that just causes an IMD product signal at the noise floor level. This will be accomplished if the synthesizer noise is less than:

$$P_n = ID_3 + 10\log(BW)$$
$$= 114.7\text{ dB} + 10\log(2400)$$
$$= 148.5\text{ dBc / Hz}$$

in the passband of the receiver. Such synthesizers hardly exist. □□

DSP Voice Frequency Compandor for use in RF Communications

Implementing NBVM gets a lot easier using low-cost DSP hardware.

John Ash, KB7ONG, Fred Christensen, KA6PNW and Rob Frohne, KL7NA

Introduction

With the radio frequency (RF) spectrum getting more and more crowded all the time there is a profound need to make more efficient use of these bands. In 1978, Harris and Cleveland developed a unique method of voice frequency and amplitude compandoring called narrow-band voice modulation (NBVM).[1] This technique offers significant bandwidth conservation, which in turn improves signal-to-noise ratio (SNR) and decreases cochannel interference. However, their technique required cumbersome analog hardware. With new DSP hardware it is now possible to develop these techniques in a flexible and fairly inexpensive way. This project involves implementing the voice frequency compandoring part of Harris and Cleveland's NBVM on Texas Instrument's new DSP Starter Kit (DSK).

Simply put, frequency compandoring is a method of removing the most unused frequency components from a given source and then shifting the remaining components together. The resulting signal may be sent over nearly any analog or digital transmission system. On the receiving end, the frequency components are expanded back out before the signal is used. With the technique described in this article, the bandwidth necessary for intelligible speech can be reduced from about 3 kHz down to about 1.7 kHz, thus conserving over 40% in RF bandwidth usage. In addition, the SNR is improved by about 2.4 dB and cochannel interference is reduced.

In normal speech, Harris and Cleveland found that intelligibility can be maintained with certain bands of frequencies removed. A spectrogram of a voice sample (Fig 1) shows frequency components at different times throughout a sampling period. The frequency components are calculated over small time intervals using a Fast Fourier Transform (FFT). Darker areas indicate higher energy. It is obvious that frequencies below 100 Hz and above 2500 Hz are used very little in human speech. Additionally, notice the frequencies from about 600 Hz to 1200 Hz. Harris and Cleveland found that after removing these frequency components, normal speech was still intelligible! Part of their compandoring technique was to simply remove this band of frequency components as well as

Notes appear on page 10.

John Ash, KB7ONG
1605 NE Valley Rd #4
Pullman, WA 99163
email: jash@eecs.wsu.edu (Internet)

Fred Christensen, KA6PNW
220 6th St
Lakeport, CA 95453
email: chrifr@wwc.edu (Internet)

Rob Frohne, KL7NA
204 South College Ave
College Place, WA 99324
email: frohro@wwc.edu (Internet)

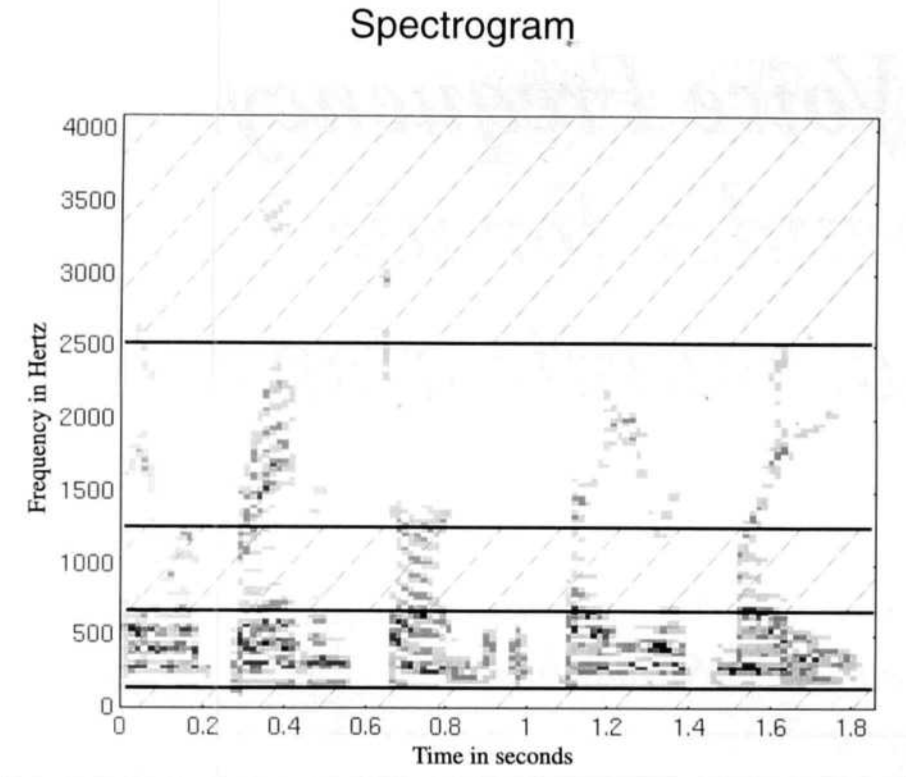


Fig 1—Spectrogram of a male voice saying: "They were brave and strong, but the tiger was mighty." In a spectrogram the presence of frequency components over a small time interval are plotted in frequency as a function of time.

those below 100 Hz and above 2500 Hz. All the remaining frequency components were then shifted together, ready for transmission.

This can be seen in another way by looking at the audio frequency response plots shown in Fig 2. A normal audio signal usually contains all the frequencies up to 3000 Hz, except for some of the very low frequencies (Fig 2a). In this compandoring technique we filter out all the frequencies above 2500 Hz and those between 625 and 1250 Hz. The filtered frequency response is shown in Fig 2b. Notice that the resulting gaps are basically wasted space because they would be inconvenient to use. Therefore, we shift the upper frequencies (1250 to 2500 Hz) down next to the lower frequencies so we have a compressed signal looking like Fig 2c. This filtered and compressed signal can then be transmitted over any audio channel. It would be difficult to understand this signal by simply listening to it since the frequency components are not in their proper places. Therefore, on the receiving end we will just shift the upper band of frequencies back up to their original places, so that the resulting signal will again look like Fig 2b. This final signal is still quite understandable.

System Overview

A complete compandoring system consists of two parts, a transmit encoder (Fig 3) and a receive decoder (Fig 4). First, the encoder takes an audio signal from a microphone, digitizes it with an A/D converter, and presents it to two specially designed digital filters (filters #1 and #2T) which eliminate all but the most important frequency components. Through Harris and Cleveland's research and some experimentation of our own, we decided the most important frequency components to keep were those from 176 to 626 Hz and from 1251 to 2500 Hz. In the next step, the upper passband (output from filter #2T) is frequency-shifted down and added to the lower one (output from filter #1) to make a single baseband signal with no spectral gaps, which is then transmitted.

The decoder on the receiving end uses similar filters for reseparating the two bands. In fact, the same filter (filter #1) can be used for the lower band, but a different band-pass filter (filter #2R) from 626 to 1875 Hz must be used for the upper band because the frequency components were previously shifted down by the encoder. The final step on the receiver end is to shift the upper passband (output from filter #2R) back up to where it originally was, add it to the lower band (output from filter #1), and send it to an audio amplifier and speaker. Though the concepts sound simple, and they are, we need to develop some theory in order to implement them.

Theory

There are only two basic functions that are needed: filtering and frequency shifting. For the filtering, we need two different types of band-pass filters. The first one, used both for transmitting and receiving, extracts the lower frequency components, which do not need to be shifted (Fig 5, block 1). The second filter type (Fig 5, blocks 2T and 2R) extracts only the positive frequencies from the higher band, allowing them to be shifted. This is explained in more detail below. For our digital filters, we chose to use finite impulse response (FIR) filters (see "Description of FIR Filters" sidebar). These can be easily implemented with a buffer of prior inputs and a series of multiply and add operations for which the DSP is optimized.[2,3] Additionally, we applied Kaiser windows with β=3 to each of our FIR filters to reduce the stop-band ripple.

FIR Filter #1

For the lower passband between f_0 and f_1, a standard band-pass FIR filter can be used. Since the frequency response of an FIR filter is periodic with period of f_s, the unwindowed coefficients may then be calculated by the following formula:

$$h_1[k] = \frac{1}{\pi k} \bullet \left[\sin\left(2\pi f_1 k / f_s\right) - \sin\left(2\pi f_0 k / f_s\right)\right] \qquad \text{Eq 1}$$

where f_S is the sampling frequency, and k is the tap number.

FIR Filter #2 and the Analytic Signal

Before we derive the tap coefficients for the second filter, we need to understand a mathematical concept called the analytic signal.[4] This tool is needed for proper frequency shifting (described in the next section).

An analytic signal is a mathematical abstraction—a complex signal which contains only positive frequency components. These components can be shifted to the desired location and the result is still an analytic signal provided they weren't

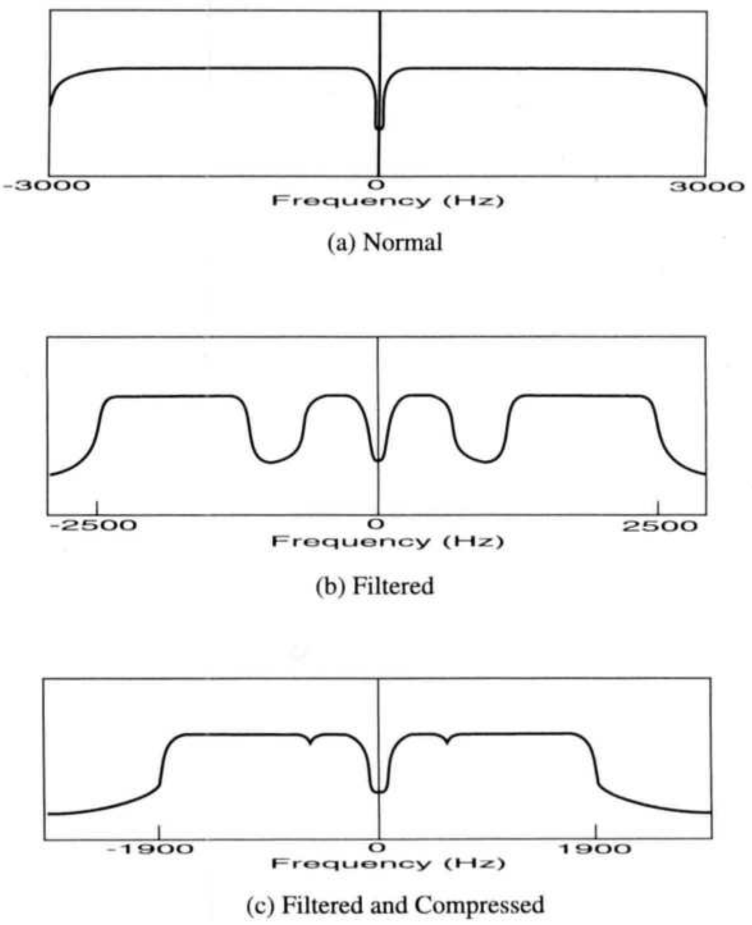


Fig 2—Audio frequency response plots for various stages in the compandoring process. (a) Typical voice audio signal. (b) Audio signal after removing unwanted frequencies. (c) Audio signal after remaining frequency components have been shifted together.

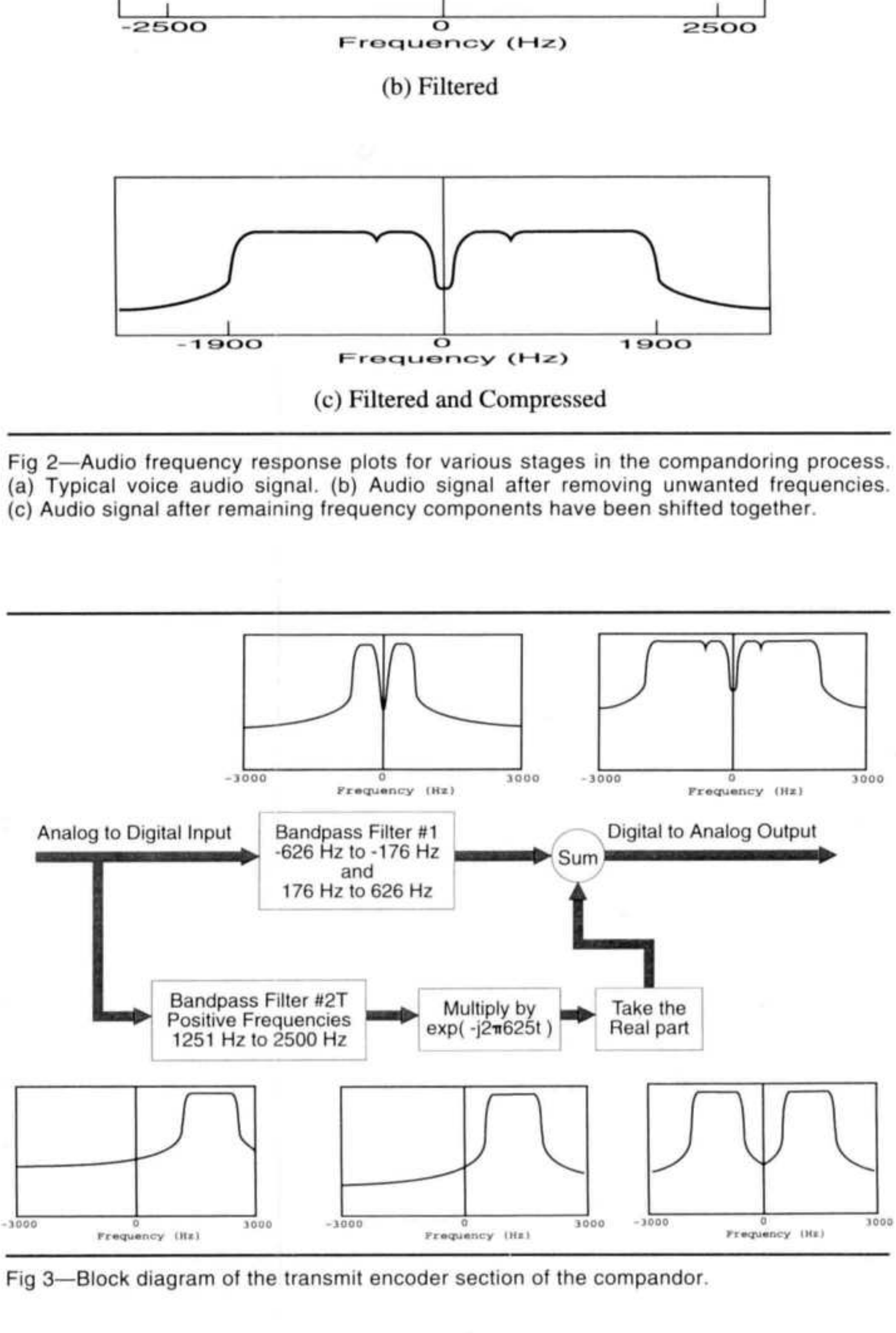


Fig 3—Block diagram of the transmit encoder section of the compandor.

shifted into the negative frequency range. Since all the required information is contained in the analytic signal, the negative frequencies can be easily restored. An analytic signal, $c[t]$, is defined to be of the form

$$c[t] = a[t] - jb[t] \qquad \text{Eq 2}$$

where t is time, $a[t]$ is the original signal including both the positive and negative frequency components, and $b[t]$ is its Hilbert transform. To restore the original signal from an analytic signal, we simply take the real part (ie, drop the imaginary part, $b[t]$).

So, how is this analytic signal created? Well, it is created by simply filtering out the negative frequency components. Since we also need to filter the passband (the one that will be shifted) we combined both of these operations into a single filter. The unwindowed FIR filter coefficients for this filter (passband of f_2 to f_3, see Fig 5, block 2T) are given by:

$$h_2[k] = \frac{1}{f_s}\int_{f_2}^{f_3}(2)e^{j2\pi fk/f_s}\,df \qquad \text{Eq 3}$$

The factor of 2 is inserted to preserve the signal power. Once evaluated this becomes:

$$h_2[k] = \frac{1}{\pi k}\left[\sin(2\pi f_3 k/f_s) - \sin(2\pi f_2 k/f_s)\right]$$
$$-j\frac{1}{\pi k}\left[\cos(2\pi f_3 k/f_s) - \cos(2\pi f_2 k/f_s)\right]$$

Eq 4

Notice that these filter tap coefficients have the form of an analytic signal (Eq 2, with $t = k/f_s$), and that the real part of Eq 4 is of the same form as the filter tap coefficients derived for filter #1, Eq 1.

In summary, the output of filter #2 contains only positive frequency components (an analytic signal) which may be shifted in frequency. After frequency shifting, the negative frequencies may be restored by keeping only the real part of this analytic signal.

The only differences between filter #2T and filter #2R are the frequencies used to develop the filter coefficients. Filter #2T is used in transmit mode and its passband output is shifted down in the spectrum. Filter #2R is used in receive mode and its passband output is shifted up in the spectrum.

Frequency Shifting

Frequency shifting can be done quite simply by multiplying a signal in the time domain by a complex exponential. As evidence of this, if the Fourier transform is taken of this product it can be seen that there will be a shift of frequencies in the frequency

Description of FIR Filters

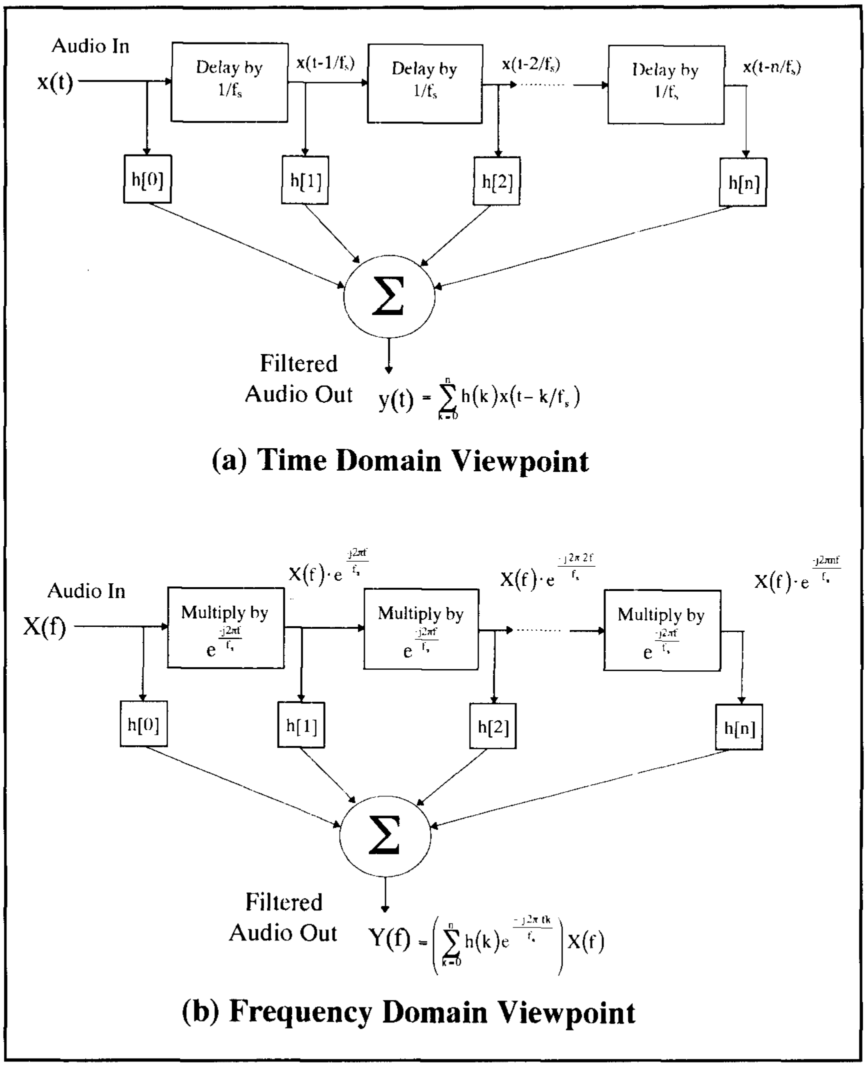

Audio In
x(t)
Delay by 1/f_s
x(t-1/f_s)
x(t-2/f_s)
x(t-n/f_s)
h[0]
h[1]
h[2]
h[n]
Σ
Filtered Audio Out
$y(t)=\sum_{k=0}^{n} h(k)x(t-k/f_s)$
(a) Time Domain Viewpoint
Audio In
X(f)
Multiply by $e^{-j2\pi f/f_s}$
$X(f)\cdot e^{-j2\pi f/f_s}$
$X(f)\cdot e^{-j2\pi 2f/f_s}$
$X(f)\cdot e^{-j2\pi nf/f_s}$
Filtered Audio Out
$Y(f)=\left(\sum_{k=0}^{n} h(k)e^{-j2\pi fk/f_s}\right)X(f)$
(b) Frequency Domain Viewpoint

Fig A—(a) FIR filter operation in the time domain. Given the *h*[*k*]'s, we are implementing an FIR filter as shown in the diagram. (b) FIR filter operation in the frequency domain. This diagram shows the frequency response of the filter in part (a). Notice that the tap coefficients (*h*[*k*]'s) are simply the Fourier series coefficients of the frequency response.

FIR filters are often used because they are simple and easy to understand (see Note 6). Diagrams of the operation of an FIR filter are shown in Fig A. An FIR filter works by multiplying an array of the most recent n data samples by an array of constants (called the tap coefficients), and summing the elements of the resulting array. (This operation is commonly called a dot product.) The filter then inputs another sample of data (which causes the oldest piece of data to be thrown away) and repeats the process.

The interesting part of designing FIR filters is translating the desired frequency response into filter tap coefficients. As can be seen from Fig A, the equation for the output of an FIR filter in the time domain is:

$$y(t)=\sum_{k=0}^{n} h[k]x(t-k/f_s) \qquad \text{Eq A}$$

where f_s is the sampling frequency and k is the filter tap number.

A complex exponential multiplication in the frequency domain corresponds to the $1/f_s$ delay in the time domain. Thus the frequency domain equivalent of Eq A (from Fig A) is:

$$Y(f)=H(f)X(f) \qquad \text{Eq B1}$$

where

$$H(f)=\sum_{k=0}^{n} h[k]e^{-j2\pi fk/f_s} \qquad \text{Eq B2}$$

is the frequency response of the FIR filter. It can be seen that *h*[*k*], the filter tap coefficients, are precisely the Fourier series coefficients of *H*(*f*), which is periodic with period f_s. Therefore the tap coefficients may be calculated from the following equation:

$$h[k]=\frac{1}{f_s}\int_0^{f_s} H(f)e^{j2\pi kf/f_s}df \qquad \text{Eq C}$$

An additional option for improving the performance of the filters, is to apply a window function to the filter tap coefficients (see Note 2). Coefficients affecting the higher frequencies are scaled down to reduce the ripple in the stop band. This can be done as follows:

$$h_w[k]=h[k]\cdot w[k] \qquad \text{Eq D}$$

where *w*[*k*] are the window scale factors, *h*[*k*] are the original tap coefficients, and $h_w[k]$ are the windowed tap coefficients. The resulting coefficients may now be used in place of the original ones. An unwanted side effect of windowing, however, is the sacrifice of a sharp roll-off at the cutoff frequency.

So, the design process is to pick the desired frequency response, *H*(*f*), and then calculate the tap coefficients (Eq C). The actual frequency response will only approximate the desired response because the number of filter taps is finite. Optionally, a window function may then be applied to the filter tap coefficients. The final step is to plot the actual frequency response, *H*(*f*), using Eq B2 to make sure the resulting response is acceptable. Lousy responses can be tweaked using different windowing schemes.

domain. In mathematical terms:

$$\Im\left[x(t)\cdot e^{j2\pi f_0 t}\right]=\int_{-\infty}^{\infty} x(t)e^{j2\pi f_0 t}e^{-j2\pi f t}dt$$
$$=\int_{-\infty}^{\infty} x(t)e^{-j2\pi(f-f_0)t}dt=X(f-f_0)$$

Eq 5

Note that t is time and f_0 is the number of Hertz shifted.

At first glance, it would appear that simply filtering the passband as in filter #1 and then multiplying by this exponential would shift the frequencies to the proper place. But this will not work because the filtered signal contains both positive and negative frequency components. If a shift is attempted in this case, all frequency components will be shifted in a given direction and the resulting positive and negative frequency components will not be symmetrical around the origin. The frequency spectrum must be symmetrical around the origin, otherwise you have a complex number in the time domain (which you'll find rather difficult to send out the D/A converter).

The way around this problem is to create an analytic signal (as described in the previous section). This analytic signal can be frequency shifted to the desired location by multiplying by the complex exponential. Then, since it is still an analytic signal, the negative frequency components can be restored by taking its real part. This is how we shift the frequency spectrum while keeping it symmetrical around the origin.

The two different implementations of filter #2 will be frequency shifted in different directions. In the transmitting section the upper band of frequencies (Fig 5, block 2T) will be shifted down, adjacent to the lower band. As pointed out earlier this can be easily accomplished by multiplying the analytic signal output from filter #2T by $e^{-j2\pi(f_2-f_1)t}$, where f_2-f_1 is how many Hertz you want it shifted down. Taking only the real part of this product effectively restores all of the negative frequency components of the shifted band, thus completing the shift operation. Adding this result to the output from filter #1 yields the final output, which can then be sent to the D/A converter and on to the transmitter.

In the receiver section the upper band of frequencies (Fig 5, block 2R) will be shifted back up to their original place (Fig 5, block 2T). This is accomplished by multiplying the output of filter #2R by $e^{+j2\pi(f_2-f_1)t}$. Again the real part of this product is taken and added to the output of filter #1. This result is then sent to the D/A converter and on to the speaker. Notice that the only difference between transmitting and receiving is a sign change in the exponential shift operator.

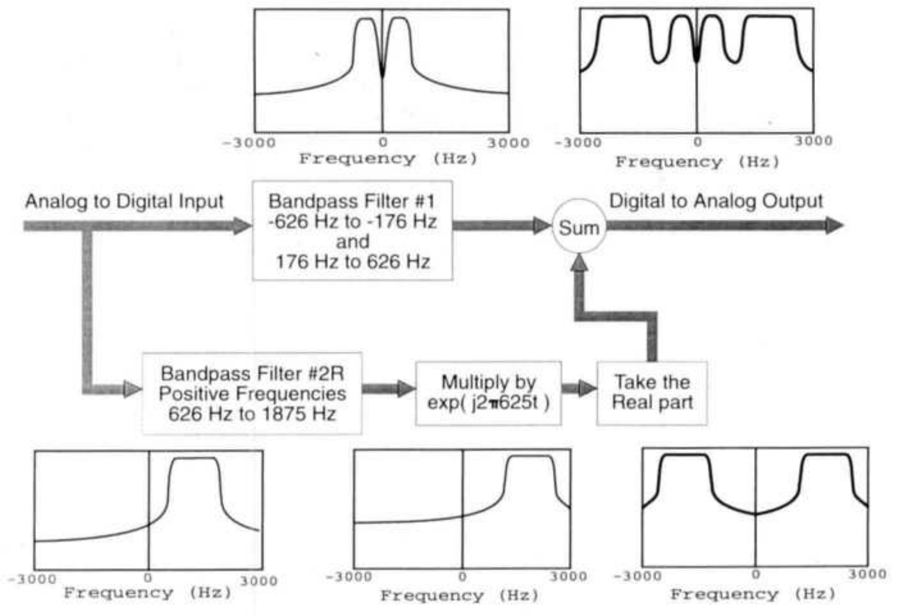


Fig 4—Block diagram of the receiver decoder section of the compandor.

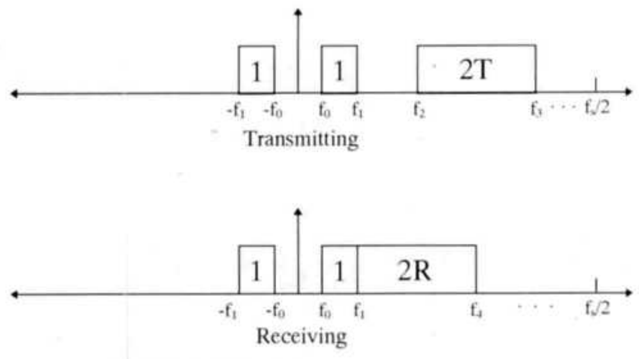


Fig 5—This shows the desired FIR filter bands. Block 1 is the output of filter #1. Block 2T is the output of filter #2T (T = Transmit). Block 2R is the output of filter #2R (R = Receive). Note that filters #2T and #2R only pass positive frequencies for shifting purposes.

Implementation

Implementing the baseband compandor system on a DSP brings many additional considerations and limitations into play. In our implementation we used the Texas Instruments DSP Starter Kit (DSK). The DSK is limited to integer math, 1568 words of memory, which may be split between data and program memory in a variety of ways, and a 100-nanosecond cycle time.[5] Our sampling rate was 10 kHz and we were able to implement 100-tap FIR filters in real-time.

When coding, it was important to minimize the number of instructions in order to leave as much memory as possible for the filter-tap coefficient tables and the complex exponential table. Wherever possible, we attempted to minimize the complex math and the number of arrays that had to be stored. We also conserved code by using much of the same code for transmitting as well as receiving. Filter #1 is the same for transmitting and receiving, and filter #2 is the same also except that different filter coefficients are used for transmitting and receiving. These facts can be used to shorten the code.

The filter coefficients were generated with the help of *Matlab* Version 4.0. After the numbers were obtained and windowed (with Kaiser windowing), they were converted into integers by multiplying by large scale factors so

Pseudo-code for the Baseband Frequency Compandor

```
INITIALIZE
  Set TR_FLAG = RECEIVE (default)
  Initialize pointer EXPPTR to the EXP array

INPUT
  Obtain a new data point from the A/D converter
  Store it in the first position of the BUFFER array

FILTER #1
  Multiply all BUFFER elements with COEF1 elements
  Sum the results into ANSR1

FILTER #2
  If TR_FLAG = TRANSMIT
    Complex multiply all BUFFER elements with COEF2T elements
    Sum the results into ANSR2
    Shift all elements of BUFFER down
  Else (TR_FLAG = RECEIVE)
    Complex multiply all BUFFER elements with COEF2R elements
    Sum the results into ANSR2
    Shift all elements of BUFFER down
  Endif

SHIFT
  If TR_FLAG = TRANSMIT
    Complex multiply ANSR2 and an element from EXP (referenced by EXPPTR)
  Else (TR_FLAG = RECEIVE)
    Complex divide ANSR2 by an element in EXP (referenced by EXPPTR)
  Endif
  Store the result in ANSR2
  Increment EXPPTR pointer
  If EXPPTR is past Last_Element then
    Reset EXPPTR to First_Element

COMBINE
  Add the real part of ANSR2 to ANSR1
  Store the result in ANSR1

OUTPUT
  Send ANSR1 to the D/A converter
  Update the TR_FLAG (Transmit/Receive)

Loop back to INPUT
```

Variables

N = *NUMTAPS*+1 = Number of filter taps
L = *LUPSIZE* = Number of elements in EXP array

Name	*Dimension*	*Type*	*Description*
COEF1	1 x N	Real	Filter #1 coefficients—same for Trans/Rec
COEF2T	2 x N	Real & Imag	Filter #2 coefficients for Transmitting
COEF2R	2 x N	Real & Imag	Filter #2 coefficients for Receiving
BUFFER	1 x N	Real	Buffer of previous inputs
EXP	2 x L	Real & Imag	Look-up table of $\exp(-j2\pi f_0 t)$
ANSR1	1 x 1	Real	Output of Filter #1, temporary variable
ANSR2	2 x 1	Real & Imag	Output of Filter #2, temporary variable
EXPPTR	—	Pointer	Pointer to elements in EXP array
TR_FLAG	boolean	(Trans/Rec)	Transmit/Receive flag

Fig 6—Pseudo-code for the baseband frequency compandor.

that integer math could be used. They were also scaled so that the amplitude of both filters in the passband was equal. Finally, the numbers were written to files and included in the program memory and data memory.

We utilized the periodicity of the sine and cosine functions to develop a finite length lookup table for the complex exponential used in frequency shifting. Strictly speaking, this limits the sampling frequency, f_s, to an integer multiple of the shift frequency, f_2-f_1. By sampling at f_s=10 kHz, and shifting the upper frequency band by f_2-f_1=625 Hz, we came up with an exponential lookup table containing 32 words. Actually, two separate lookup tables of 16 words were used, one for the sine function (imaginary part) and one for the cosine function (real part). For higher sampling rates the size of the tables will increase accordingly.

Fig 6 shows the pseudo-code for the DSP program we developed. Actual assembly code and filter coefficients that we used for the DSK can be obtained over the Internet by anonymous ftp from the site "ftp.wwc.edu" in the subdirectory "/pub/compandor."

Conclusions

The prototype system we developed works surprisingly well with quite understandable speech even after all the processing that has been done to it. We tested transmissions with two compandors (DSKs) on 40-meter SSB using three radios to compare cochannel interference and signal quality. The third radio was used for creating an interference signal. The compandor definitely helped to lessen the cochannel interference from this interfering signal transmitting at the same signal strength. We were unable to test the full impact of the compandor, though, since we have only 2 DSKs. In other words the interfering radio couldn't transmit a compandored signal. We have not used it extensively yet on the air since our present prototype does not have automatic transmit/receive mode switching, or stand-alone capability.

Notes

[1]Harris, Richard W. and Cleveland, J. F., "A Baseband Communications System," *QST*, Nov 1978.
[2]Oppenheim, Alan V. and Schafer, Ronald W., *Discrete-Time Signal Processing*, Prentice Hall, 1989.
[3]Karl, John H., *An Introduction to Digital Signal Processing*, Academic Press, Inc, 1989.
[4]Bracewell, Ronald N., *The Fourier Transform and Its Applications*, 2nd ed, 1986.
[5]*TMS 320C2X User's Guide—Rev C*, Texas Instruments Inc, 1993.
[6]Morrison, Frank P., "The Magic of Digital Filtering," *QEX*, Feb 1993.

RF Counter Data Collection and Visualization

Collect the data, reduce the data, interpret the data—easy enough if you have the tools!

By Brian J. Mork, KA9SNF

Introduction

Many of us own or have at least used radio frequency counters. Perhaps you have been thinking about getting one. A recent article introduced the necessary vocabulary, with several references for further reading.[1] This article expands on one of the capabilities becoming available on hand-held frequency counters. It will expose you to what can be done with a digital data port and provide details on how to do it.

Optoelectronics, one of many manufacturers, recently made additions to their product line that instigated this article. Their two newest frequency counters offer a digital data port as a standard feature. In this article, I'll describe how to build an interface and develop the software necessary to use this capability. The post-processing I'll show you will tremendously increase the value of what could otherwise become an underused feature. I think you'll find the software portion refreshing. I'm not going to force you to learn a new program. Rather, I'll show you how to build a new capability using programs you already have.

Who's Interested?

Who might be interested in using a frequency counter with data logging and analysis capability? One group would be the bench-top builder, tinkerer or contractor who wants to document progress. As you'll see later, in the process of documenting frequency readings from the counter, you can

Notes appear on page 23.

6006-B Eaker St
Fairchild, WA 99011

Photo 1—The Optoelectronics M1 frequency counter on the left and my Zenith Minisport laptop computer on the right. The surplus equipment vendor sold me this computer for $50 cash and carry. Notice all identifying marks on the computer were removed, representing the requirements of the surplus equipment market.

improve the quality of the measurements. Another more exciting use is to go sniff the air waves and see what's out there. How about determining the frequencies being transmitted from some mysterious tower on the back of the local municipal building? How about searching out intermittent hidden transmitters? *Nuts & Volts* recently published a sequence of articles on electronic privacy and surveillance.[2] The equipment and methods I describe are a natural for this area of interest.

I am basing the specifics of this article on my Optelectronics M1 frequency counter and a surplus Zenith Minisport laptop computer that is IBM-PC/MS-DOS compatible. Photo 1 shows the complete system. Optoelectronic's M1 and 3000A counters have identical data ports. Either one could be used with the methods described here. The Minisport is particularly amenable because of its TTL-level serial port. I purchased mine for only $50 from the source listed in the "Points of Contact" sidebar. If your computer doesn't have a TTL-level port, don't worry. It's simple to convert between TTL and RS-232 voltage levels. I'll describe several methods to do this. Alternately, you could follow the examples of the Poor Man's Packet (PMP) or Baycom packet modems and use parallel port bits (which are TTL) to transmit and receive serial data. If this capability catches your fancy, read up on the Poor Man's Packet TNC design, and my MLHacker series.[3,4]

Technical Background

Optoelectronics sells a commercial interface and logging program (the CX12 RS-232C converter and *OptoLog* software). Their equipment offers a degree of potential compatibility with the rest of the world because of its commercial availability. I expect it comes with the same one-year warranty as the counters. It's compatible with any device conforming to RS-232 voltage levels and 3-line RS-232C logic. Minimum effort here. On the other hand, it's rather expensive and it's inflexible. Their system is just a logger. I'm going to show you how to log and graphically visualize the data. If you follow me through this article, you'll have the understanding and details to create your own superior data processing system at a fraction of their cost.

The Optoelectronics 3000A and M1 meters, both recent designs, are reviewed elsewhere.[5] The 3000A offers some additional features, but these data are not available from the data port. Three items mentioned in the reviews are relevant to this article.

The data port is a stereo ⅛-inch phone jack. The M1 applies voltage levels on the ring of the plug/jack (transmitted serial data). If I accidentally plug in a monaural jack, I just shorted out the counter's transmitter, which, judging from the chip arrangement inside, is not buffered. In other words, a simple user-error shorts out the microprocessor. How about reversing the send and receive lines?

Secondly, the counter has limited ability to associate the frequency data with real-world events. Computer initiation of the data transfer gets back a frequency measurement, but there's no way to know if it's a new frequency or not. It may be a duplicate of the previous reading, or it may be the previous reading. With the power of a microprocessor on board, it would seem simple to send an extra byte to flag it as "old" or "new" data. With ingenuity, an extra byte isn't even needed. ASCII characters are used, and the 8th bit of each is wasted. All those wasted bits could transfer the status of the on-board memory, switch positions, age of the data, signal strength, etc. Even passing all this information, there would be unused bits! If you didn't care to interpret this extra information, the 8th bit would cause minimum grief to simple logging programs (a lot of software just strips the 8th bit anyhow).

Lastly, I have never understood why a counter with 1-part-per-million accuracy needs more than six digits of resolution. A lot of manufacturers are building them this way, but realize only the six most significant figures are meaningful no matter how many show on the display or are sent over the dataport.

Hardware

As mentioned above, Opotoelectronics offers a converter box that will work between their counters and your standard IBM/PC compatible serial port. In the spirit of this article, I have a few alternatives. Remember the M1 has two signal connections (one each way) at TTL-compatible levels.

Native TTL

The Minisport laptop computer I use has a normal male 9-pin serial port on the back. Additionally, it has a 16-pin double-row header docking port for a custom modem module. The docking port offers the TTL equivalent of a serial port at address 2F8(hex) (MSDOS COM2 in this and most other computers), power, and a power-down option. Photo 2 shows the interface card, and Table 1 shows the pinout connections between the Minisport and the M1 used for this project.

Even without the convenience of a header like this, most PC-compatible computers have TTL level-compatibility in the form of the parallel port. On newer parallel port cards (post PS/2),

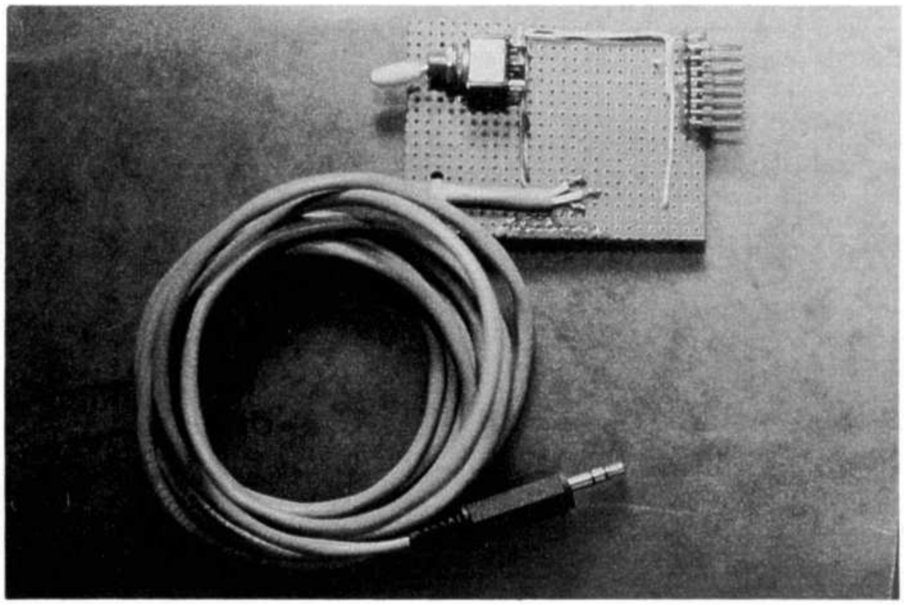


Photo 2—The interface between the M1 and the Minisport requires no active components. The stereo phone jack plugs into the M1. The 16-pin header plugs into the Minisport. The 1⅞ x 3-inch perf board card slides into a recess until the top of the toggle switch sticks out about ⅛ inch.

this is a bidirectional port and would work fine with drivers to toggle the parallel bits so they look like serial data. The Poor Man's Packet or Baycom software does this and is widely available. Alternately, you could build an add-on board turning your parallel port into a bidirectional 16-bit port.[6]

Packaged Converters

If you have to (or prefer to) use an RS-232 serial port, you can convert the voltage levels with your own single-chip converters. This route is still about 1/10th the price of the CX12 interface.

You could build a level converter with the classic 1489/1488 chip pair, and even more options are now available.[7] Maxim (see the "Points of Contact" sidebar) makes an entire series of chips for this type of interfacing. They even offer a low-power chip, the MAX220, that has a quiescent supply requirement of only 500 µA while generating its own ±10 V. It can be powered from the serial port lines themselves (no external power supply required). If your chosen interface uses too much power, consider tapping an IBM-PC compatible game port. Ground is available on game port pins 4, 5, and 12. Power (+5 V) is available from pins 1, 8, 9, and 15.

Table 1

Minisport modem header and M1 serial data port connect lines. For other computer hardware, build the cable so that the computer applies voltages on the ⅛-inch phone plug TIP, and the M1 applies voltages on the plug's ring. All references ground, on the plug's shield.

Minisport modem header looking in the slot, right side up:

1	3	5	7	9	11	13	15
2	4	6	8	10	12	14	16

1	+5-V output
2	+10-V output
3	−10-V output
4	GND—connect to phone jack SHIELD
5	TTL "modem on." High when CMOS setup turns COM2 on. Alternately, the DOS command MACHINE MODEM ON lets you set it to high; MACHINE MODEM OFF sets it to low.
6	Carrier Detect (CD), TTL input
7	Received Data (RxD), TTL input—connect to phone jack RING
8	GND
9	Transmit Data (TxD), TTL output—connect to phone jack TIP
10	Speaker. Touching +5 V through a 1-kΩ resistor makes the speaker squawk.
11	Ground ?
12	Ring Indicator (RI), TTL input
13	Data Terminal Ready (DTR), TTL output
14	Clear to Send (CTS), TTL input
15	Request to Send (RTS), TTL output
16	Data Set Ready (DSR), TTL input

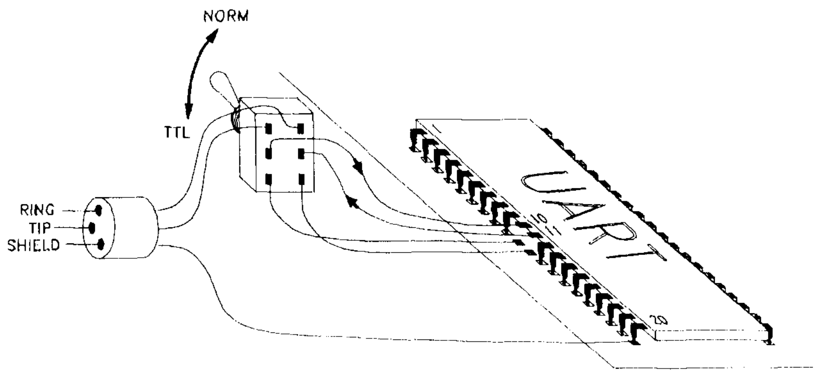


Fig 1—Perspective view of one hardware option: a UART tap-in. Mount the switch and a new connector (choose a noqPC style) on the back of your computer. Pins shown are good for 8250, 16450, and 16550 chips. If you don't have one of these UARTs, instructions for tapping in by the RS-232 buffer chips are given in Table 2 and the text.

Hardware Tap

If you're a hardware hacker type, a fourth alternative may be for you. Extract TTL levels from your computer by intercepting the data lines on your computer's I/O card. Grab them between the UART and the RS-232 level shifters.

Chances are good that you have an 8250, 16450, or 16550 UART on your computer. If you look on your serial card and see one of these chips, Fig 1 is for you. You could cut a circuit trace, but I find it much cleaner and more reversible to gently bend up two legs of the UART. As shown in the figure, this, in effect, "cuts the trace" and lets you get in between. Route the UART side of the cut to the common pole of the new switch. Connect one pole of the switch to the data line coming from the M1 counter. The other pole of the switch then picks up on the data line coming from the RS-232 voltage level shifters. In Fig 1, I showed a generic round connector to emphasize that I would not use a standard DB-9 or DB-25 or any other connector typically found on the back of a PC. You don't want to accidentally plug in your standard peripherals into this jack. The voltages you plug in here go directly to the UART without buffering. Keep them within TTL or +5-V-powered CMOS limits.

Nonstandard or highly integrated architectures incorporate the function of UARTs into other chips. For instance, I have a GSI multi-I/O card. It provides two 16550 ports using an XR16C552 chip. Voltage-level conversion to RS-232 standards, however, is still done with the 1488/1489 pair.

Under these circumstances, it's probably easier to tap the lines at or near the level shifters, on the UART side. You'll have to use an ohmmeter and find where pins 2 and 3 of the back-panel serial connector connect to the 1488 or 1489. Use Table 2 to determine which IC pins you should tap into. When wiring the DPDT switch under this setup, there's a subtle change. In Fig 1 (tap by the UARTs) the IC pins go to the switch common (center) terminal. If you tap into the level shifters, the circuit board trace or IC socket gets the common terminal and the IC pins get connected to one pole of the switch.

Cutting circuit board traces or bending up IC pins is required because you can't just splice the counter's data line into the existing TTL line; the 1489 and the M1 would both be trying to assert voltage levels on the line going to the UART. You don't know which device will successfully communicate with the computer—or whether they will exceed each other's current limits as the two devices try to assert different voltage levels.

For the line headed out of the computer, the UART's ability to source and sink current (to drive the preexisting level converter and your counter) are your primary concern. Because a DPDT switch is insignificantly more expensive than an SPDT switch, I recommend you switch both lines as shown in Fig 1.

While You're Building

If you own a Minisport and you already have a soldering iron out to build a board like that shown in Photo 2, realize that TTL-level data lines are made available on the computer modem header *after* coming through a UART. RS-232 levels are not normally available on COM2 of a Minisport. Owners of a Minisport wishing to *add* RS-232 levels to COM2, see Minisport Laptop Hacker #15. Notice the blank central area on my interface card? This is my intent. It won't be used for this project, but it will double the standard COM port count!

Under these circumstances there are also options available to allow automatic selection of data sources when using the Minisport. These schemes eliminate the need for a switch. If the M1 gets plugged in, it talks correctly with the computer. If an RS-232C device gets plugged in, it talks correctly with the computer. This way RS-232C or TTL devices can be interchangeably used. See Fig 2 for design details.

Software

Low-Level Data Format

The data from the counter consists of a serial stream at 4800 bps, 1 start bit, 8 data bits, 1 stop bit, and no parity bit (8N1). From memory, I believe the *3000A Owner's Manual* describes a bit rate of 2400 bit/s. Voltage and current limitations are shown in Table 3. These are typical low-power, TTL-compatible signals. Polarity matches that normally used to feed 1488 or Maxim level converters.

Optoelectronics says data comes from the M1 in 11 ASCII character bursts. Actually, there are 12 if you count the terminating carriage-return (CR) that is included after every reading from the counter.

You request a burst of 12 ASCII characters by sending the counter an ASCII CR character. Fig 3 shows what the CR looks like, reduced to simple voltage levels. The data line to the counter (the phono plug tip) must have a quiescent voltage level of greater than 2 V and pulse down to below 0.8 V to indicate a CR.

Although the specifications are for 8N1 data format, 7S1 (7 data bits, space parity and 1 stop bit) also work. The counter can't distinguish which you're sending. Other formats also get the CR to the counter properly, but then show parity errors as the ASCII data arrives back to the computer. Why even mention these alternate formats? Because I want to emphasize that

Table 2

Tapping data lines between the UART and RS-232 level shifters, near the level shifters. Find which pin of a 148x the back panel connector pin goes to/from. Read across to the "Tap Into" column. That's the pin on the IC you need to bend up for access of that line. In the case of the 1488 driver, three of the potential outputs have two inputs. One input is likely to be unconnected or tied to a high voltage independent of data being sent. The other input (representing the data) is the one you want.

DB-9	*DB-25*	*Function*	*If routed to...*	*Tap Into*
5	7	Ground	1488-7 & 1489-7	n/a
2	3	Data from M1	1489-1	1489-3
			1489-4	1489-6
			1489-13	1489-11
			1489-10	1489-8
3	2	Data to M1	1488-3	1488-2
			1488-6	1488-4 or -5
			1488-11	1488-12 or -13
			1488-8	1488-9 or -10

Table 3
Voltage Levels and Current Capability of the M1 and 3000A Counters.

Data out, TxD, (ring)		
	Logic 0 0.0-0.45 VDC	(1.60 mA max sink current)
	Logic 1 2.4-5.00 VDC	(0.06 mA max source current)
Data in, RxD, (tip)		
	Logic 0 0.0-0.70 VDC	(0.05 mA max load current)
	Logic 1 2.0-5.00 VDC	(0.05 mA max load current)

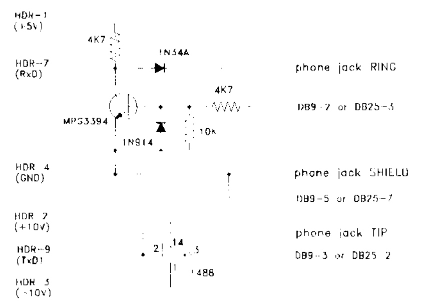


Fig 2—Automatic selection of RS-232 or TTL voltage levels. The HDR-x connections are numbered for the Minisport modem header. I showed a discrete transistor and a 1488 gate. However, a 1489 gate can be used to replace all the discrete components of the RxD circuitry except the germanium diode. A third option is to use a MAX233A—replacing everything except the germanium diode.

ASCII data is nothing more than timed voltage excursions. In fact, the simplicity of this whole data interface begs for the design of a custom "carriage return generator" and the associated receive electronics to make a remote reading display for the frequency meter. This would be simple using Microchip's PIC series of microcontrollers. Maybe the next project. Let me know if you're interested.

Most of the time, you'll just use a PC-compatible serial port to generate the waveform of Fig 3. But don't shy away from other options. Computers from other manufacturers can be made to do the same voltage transitions. As mentioned before, you could send and receive serial data through the parallel port by oscillating and monitoring certain bits. If you elect this route, you'll have to acquire software that controls these bits properly. One good source is local landline BBSs that have CD-ROMs available on-line. Look around for an amateur radio BBS and look for CD-ROM shareware for the Baycom or PMP modems. My point is this: simple timed voltage excursions are serial communication. Sometimes we forget that.

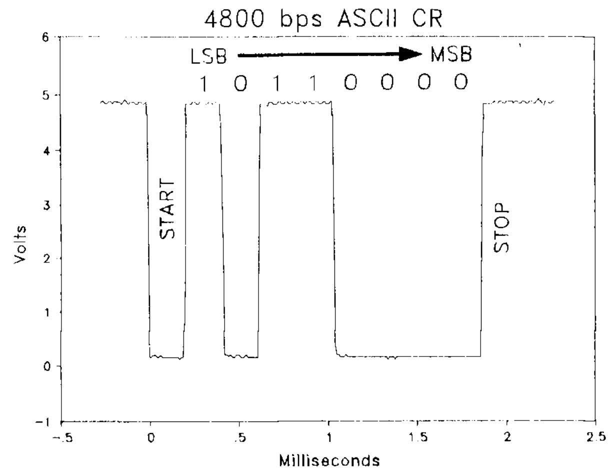


Fig 3—Voltage waveform used to poll the M1, acquired with a Fluke 97 Scopemeter. There is no difference between 8 bits, no parity, 1 stop and 7 bits, space parity, 1 stop bit. Each generates this waveform, which, when applied to the phone jack tip, causes the M1 to send back a frequency reading on the phone jack ring.

```
  set port com2                    ;Minisport's modem header
  set baudrate 4800                ;3000A docs say 2400 baud. Typo?
  set parity none
  set stopbits 1
  set cr_in cr_lf                  ;map each inbound CR to a CR/LF pair
  message "Enter filename for frequency data: "
  kflush                           ;flush the keybord
  get S0
  message "^MReceived filename:"
  message S0
  isfile S0
  if success
    message "Requested file already exists, appending data."
  endif
  message "ESC to terminate command script & not save data."
  message "Turn off meter or disconnect cable to terminate & save data."
  rflush                           ;flush the receive buffer
  log open S0
  if failure
    message "Can't open the requested file; exiting command file"
    goto exit
  endif
request:
  transmit "^M"                    ;A "^M" generates a carriage return
  waitfor "^M" 3                   ;Wait for the counter's CR
  if waitfor
    goto request
  else
    goto nodata                    ;to stop data collection, turn off meter!
  endif
data:
  message "Incoming data timeout: saving previous data."
exit:
  log close
```

Fig 4—*Procomm* script for logging frequency data. Type this in your favorite editor and save as file GF.CMD.

Control/Logging Program

I use *Procomm* as my primary communication program, and its script language is powerful enough to automatically collect data from the M1. It sends the repetitive requests to the counter for data and logs incoming values to a file. Subsequently, I use another general purpose program (a spreadsheet) to process and plot the data. This is the environment I'll cover in the rest of this article because I think almost everybody has a spreadsheet and a communication program capable of logging incoming text. I'm interested in using software I have to the greatest extent possible rather than learning new stuff!

Some of you may wish to use dedicated custom software that logs the data and processes it. There are several reasons why this is tempting. Foremost is the opportunity to selectively save data, reducing the reams and reams of data generated by the infinite loop script shown in Fig 4. Second, you could integrate your computer-controlled scanner and a tape recorder switch into the scenario. Imagine listening for transmissions across the entire spectrum, sending the scanner to any active frequency and engaging the tape recorder. Writing a custom program to glue together equipment you already own is attractive! If you're interested in doing this, send me information on your scanner and I'll work with you to get the project going. Perhaps we can coauthor the successor to this article.

Other, lesser reasons may prompt you. Perhaps you need fast turnaround of the raw data interpretation. You could have a real-time histogram "grow" in front of your eyes if you integrate the appropriate graphic routines into your program. You could generate a speaker tone when data meeting

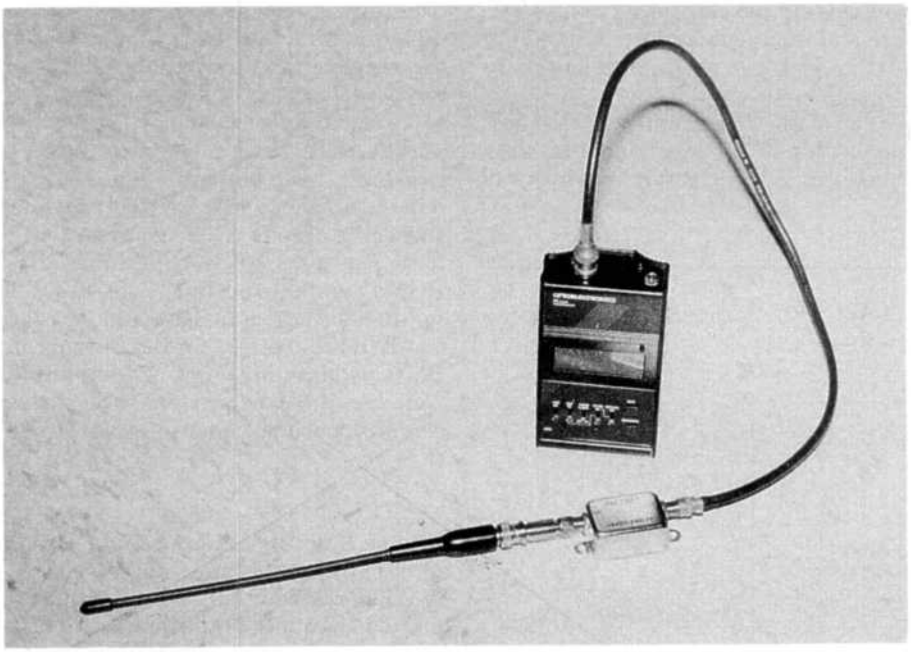


Photo 3—The M1 counter, with the "sniffer cable" and FM notch filter installed.

some criterion is acquired.

Many times, dedicated software needs to be written for fast data acquisition. In the case of this meter, it's not necessary. Because these meters handshake only as polled equipment (you get data only when you request it), a slow *Procomm* script isn't overrun. Even if you do PMP or Baycom-type bit processing (no UART to handle the task), you can knowingly set up to handle the burst of bits ahead of time. Once the burst of bits starts coming from the M1, you'll have 208 µsec to process each bit. Once all the bits are saved, you can process to your heart's content because nothing else is going to be coming until you request it.

In the end, it's largely a matter of choice. In this article, I'm trying to keep the process of data collection educational. These methods are portable. Visible. Not hidden inside some EXE file. They're flexible to your changes. I want to fuel your imagination and encourage you to carry this introduction further. Let me know what you do.

Data Collection

Connections

Connect the computer end of the cable to the serial port TTL-level lines. For Minisport owners, this involves sliding the perfboard card into the left side of the computer. Press it in to seat into the connect header. Connect the 3-conductor ⅛-inch plug on the other end of the wire into the counter's serial jack. Table 1 shows the wiring of this cable for the Minisport. If you've built another type of interface, just remember, the computer must provide TTL signals on the tip; the M1 will provide TTL-compatible voltages on the ring.

Operator Procedure

While running *Procomm*, use Alt-P to choose 4800,N,8,1,COM2 from the *Procomm* Line Settings menu. Turn on the meter and press a CR on the computer keyboard. The most recent frequency reading will appear on the computer screen, preceded by blank spaces if anything but the highest resolution is selected on the counter. If this doesn't work, there's probably a problem with your wiring or your communication settings. Check again.

If this works, the next step is to check proper entry and exit of the command script. Turn the meter back off and use Alt-F5 to run the command file GF.CMD by specifying "GF". The script will ask for a data filename. If it is original, this file will be created. If the file already exists, you will be informed that data will be appended to the end of the existing file. Enter a filename and press return. The filename is echoed back to you for confirmation and then you are reminded of the two ways to exit the data collection loop.

The two ways of exiting are (1) press ESC to abort the command script loop, or (2) timeout waiting for response from the meter. Timeout is the normal way to exit, causing all data logged by *Procomm* to be saved into the previously specified file when the log closes. ESC will cause the data collection loop to be stopped without closing the log file, so the last block of data will not yet be written to the disk. Because, in this example, we've run the script with the meter turned off, the script should timeout prior to getting any data (takes about 3 seconds). Use the Alt-F4 option of *Procomm* to shell out to DOS and TYPE the filename you specified. It will contain the lone message "Incoming data timeout: saving previous data." This will be the last entry in a set of data (unless you used ESC to abort data collection) and can be used to separate data sets that are subsequently appended to the same file.

Type EXIT at the DOS prompt to return to *Procomm*. Turn the meter on. Press CR a couple times to be sure everything is working. Use Alt-F5 to run the GF script again. Enter a filename. Sit back and enjoy! You're collecting reams of frequency readings. Press ESC or turn off the meter (or just unplug the serial cable) to stop data collection. If you used ESC, you may wish to manually close the data file (Alt-F1) to write the last block of data.

Meter Options

Notice that selecting long gate times on the M1 virtually guarantees multiple readings of the same data. If you run the meter with the filter engaged, you'll be assured of stable readings on the meter display, but even if no new reading is obtained, the script will continue logging the most recent frequency over and over again. Running with the meter's capture option engaged is undesirable. You *don't* want to stop the meter from getting new values.

The best data is obtained by running with the longest gate time that does not force duplicate data. I prefer to run with the filter and capture off. The post-processing I'll show you with the data actually works better with "full throttle" data collection, getting a copy of anything the meter is hearing.

Along this vein, it is important to limit what frequencies your meter hears. Don't use a scrap piece of wire for an antenna. Select an antenna tuned to resonate near your frequency band of interest. Listening to ambient RF noise with no antenna on my meter gives random counting at about 1 to 2 MHz, with no segments of the signal strength graph illuminated. With an AR1000XLT scanner stock antenna installed, the display hovers around 95 to 100 MHz. With a Radio Shack FM band notch filter in line, about half the bar graph is illuminated and frequencies in the 75 to 85-MHz range are recorded. See Photo 3 for this arrangement. It is my favorite. The FM trap and extra 24 inches of coax make a nice hand-held "sniffer" probe for poking around radio phones, RF-leaky computers, out the car window, etc.

Performance Figures

The speed at which you collect data can be determined by several factors. Using the methods I've already presented, the rate is determined by how fast your computer executes the command script. It turns out to be about 134 samples each minute—about 2 per second.

Alternately, you can select the acquisition rate by making a file of CR characters and using the communication program to send these to the meter at a predetermined rate with the paced ASCII upload option. Tables 6 and 7 give the necessary syntax for doing this type of paced acquisition using *Telix*, another communication program with a wide user base. Either scripted or paced data acquisition can be done with either *Procomm* or *Telix*. The documentation provided should be enough to get you going on the other method or the other program, depending on what you have and want. You can let the system run indefinitely, or sample for a known time at a known rate.

Having long gate times selected on the counter will affect how often new data is obtained, but the command script will chug away at the predetermined rate, recording duplicate versions of the last reading if no new data has been recorded by the meter. Gate time and meter options may be changed on the fly during a data collection run with no adverse effects. Whatever gate time you have selected on the meter will be reflected in the number of digits after the decimal point for each frequency reading.

The data file will be a plain ASCII text file, consisting of one frequency per line. The meter sends only a CR to terminate each line. Be sure you have the "CR translation in" option of your communication program set to translate inbound CRs to CR/LF pairs, for compatibility with MSDOS files. While you're at it, be sure the "CR translation out" option is set to send only CRs.

Post Processing

Prior Preparation

I realize there are many different spreadsheet programs out there. I want to convey the procedures to everybody and, yet, offer exact commands to those who can use them. If your program uses slightly different syntax or naming conventions, pay attention to the overall intent of each of the following sections. Specific commands for *SuperCalc* and *AsEasyAs* (a shareware spreadsheet) are tabulated in Table 4. These give the specific syntax for the operations I'll refer to in the next sections. Before committing to the spreadsheet you currently have, read ahead and make sure it can do the required operations.

Two excerpts from spreadsheets I use are shown in Figs 5 and 6. These figures are important for two reasons. First, they give all the cell formulas you'll need to make your own spreadsheet template. You should make two master templates and save them under names like EXTRACT and BIN. The implications of these names will become clear later. Each time you process a set of data, copy these templates over to a new filename and then load the data into that new file. Alternately, when you start the spread-

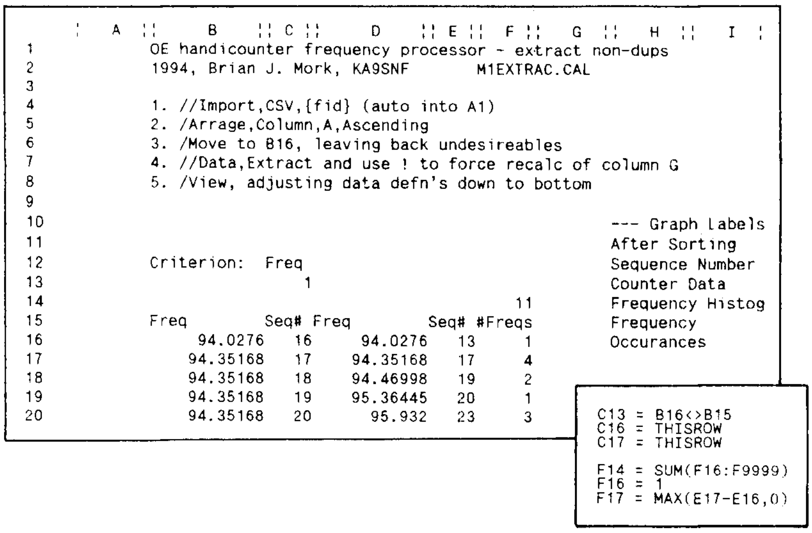
Table 4

Quick reference card for *AsEasyAs* and *Supercalc* commands you'll probably use manipulating frequency data spreadsheets. Italicized, bracketed entries will be different for various sets of data.

Activity	*AsEasyAs*	*SuperCalc*
Load Values	/File,Import,Values,*{fid}*	/Import,CSV,{fid},All
Sort	/Data,Sort,Datarange,A1.A*{R}*, Primarykey,A1,Ascending,Go	/Arrange,Column,A,A1:A*{R}*, Ascending,No-Adjust,Go
Delete Rows	/Sheet,Delete,Row,*{range}*	/Delete,Row,*{R}*
Move Cells	/Movecells,A*{1st}*:A*{last}*	/Move,Block,A*{1st}*:A*{last}*,B16
Add Sequence numbers	/Copy,C11,C12	/Copy,C11,C12:C*{R}*

Extract non-dups	/Data,Question,Inputrange,B15..C*{R}*
	/Data,Question,Outputrange,D15..E*{R}*
	/Data,Question,Criterion,C10..C11
	/Data,Question,Extract
		//Data,Input,B15:C*{R}*,
		Output,D15:E*{R}*,Extract,Quit

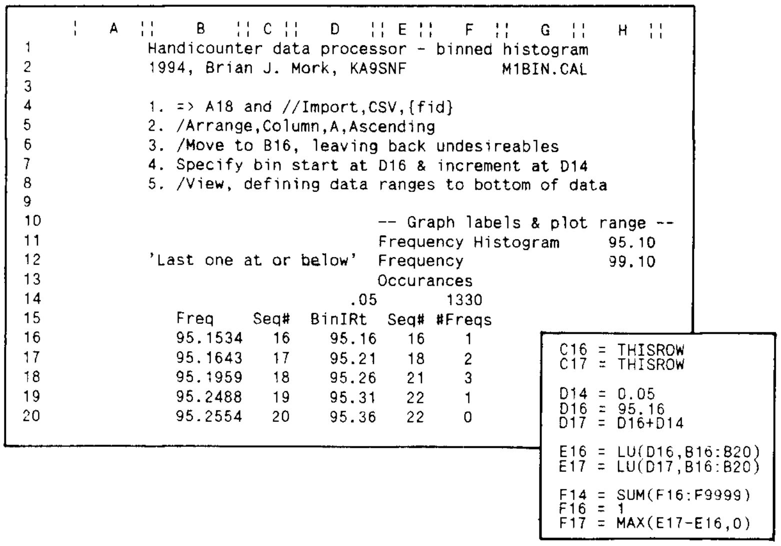
Line Graph	/Graphics,Range,X,D14..D*{R}*,Q /Graphics,Range,A,F14..F*{R}*,Q /Graphics,View	/View,1,Graph,Line,Data,B16:B*{R}* /View,1,Show
Histo Graph	/Graphics,Type,X-Y,Range,X, D14..D*{R}*,Q /Graphics,Range,A,F14..F*{R}*,Q /Graphics,View	/View,2,GraphXY, Data,D16:D*{R}*,F16:F*{R}* /View,2,Show
Save	/File,Store,*{fid}*	/Save,*{fid}*,All,...



```
     |  A  ||     B     || C ||      D      || E || F ||  G  ||  H  ||  I  |
 1           OE handicounter frequency processor - extract non-dups
 2           1994, Brian J. Mork, KA9SNF       M1EXTRAC.CAL
 3
 4           1. //Import,CSV,{fid} (auto into A1)
 5           2. /Arrage,Column,A,Ascending
 6           3. /Move to B16, leaving back undesireables
 7           4. //Data,Extract and use ! to force recalc of column G
 8           5. /View, adjusting data defn's down to bottom
 9
10                                                         --- Graph Labels
11                                                         After Sorting
12           Criterion:  Freq                              Sequence Number
13                             1                           Counter Data
14                                                 11      Frequency Histog
15           Freq        Seq# Freq          Seq# #Freqs    Frequency
16               94.0276   16     94.0276    13     1      Occurances
17              94.35168   17    94.35168    17     4
18              94.35168   18    94.46998    19     2
19              94.35168   19    95.36445    20     1
20              94.35168   20      95.932    23     3
```

```
C13 = B16<>B15
C16 = THISROW
C17 = THISROW

F14 = SUM(F16:F9999)
F16 = 1
F17 = MAX(E17-E16,0)
```



```
     |  A  ||    B    || C ||    D    || E ||   F  ||  G  ||  H  ||
 1           Handicounter data processor - binned histogram
 2           1994, Brian J. Mork, KA9SNF           M1BIN.CAL
 3
 4           1. => A18 and //Import,CSV,{fid}
 5           2. /Arrange,Column,A,Ascending
 6           3. /Move to B16, leaving back undesireables
 7           4. Specify bin start at D16 & increment at D14
 8           5. /View, defining data ranges to bottom of data
 9
10                                  -- Graph labels & plot range --
11                                  Frequency Histogram       95.10
12           'Last one at or below'  Frequency                 99.10
13                                  Occurances
14                               .05         1330
15              Freq    Seq#  BinIRt  Seq# #Freqs
16              95.1534   16    95.16   16    1
17              95.1643   17    95.21   18    2
18              95.1959   18    95.26   21    3
19              95.2488   19    95.31   22    1
20              95.2554   20    95.36   22    0
```

```
C16 = THISROW
C17 = THISROW

D14 = 0.05
D16 = 95.16
D17 = D16+D14

E16 = LU(D16,B16:B20)
E17 = LU(D17,B16:B20)

F14 = SUM(F16:F9999)
F16 = 1
F17 = MAX(E17-E16,0)
```

Fig 5—The top section shows the *SuperCalc* spreadsheet used to process data by extracting nonduplicate frequency readings. The inset shows formulas are for those cells that are not obvious from the spreadsheet extract. The lower section shows a spreadsheet extract and formulas for a binned histogram spreadsheet.

sheet program, load a template, add in the new data and then save it under a new name.

Second, throughout the rest of this article I'll refer to certain spreadsheet cells and assume you've built a spreadsheet to match. For instance, I'll assume cell B16 contains the first data point after sorting. Of course you can build your spreadsheet however you want, but then you'll have to transpose all the references I mention.

Notice that two types of spreadsheets are given in each figure. These correspond to the two types of histograms spoken of. The first extracts one copy of each new frequency and counts how many duplicates there were. The second lets you specify bins and counts how many readings belong in each bin. Later I'll walk through the design of the extraction scheme, then the binned scheme, but for now, you need to generate the blank templates. Use the figures and the following descriptions as guidance.

Columns B and C extend to the bottom of your data. Columns D, E and F will usually not extend as far for reasons that will become obvious. The top rows identify the spreadsheet and give a reminder on how to use it. These numbered steps don't parallel any division in this article. They are meant to be meaningful to an experienced user coming back to the spreadsheet after a period of nonuse.

Cell C13 contains the criterion defining what data is to be extracted. It's a mathematical version of "if this frequency isn't like the previous one, take it." Cell C13 will show a 1 (true) to indicate B16 is not the same as B15. Cell F14 sums up all the values in column F, giving the "area under the curve" for the histograms you'll be plotting. The length of column F (after you extract the plot data from column B) indicates how many unique values are plotted. Cell F14 counts how many total readings are represented.

Don't worry about column B below cell B15. You'll fill that with real data. The numbers in column C are sequential numbers. There is no special start number; they just have to increment by one. With *SuperCalc*, the formula entry THISROW works fine, or use the formula given for *AsEasyAs* cell C17 and replicate it downward using the command given in Table 4.

Columns D and E are filled in automatically by the spreadsheet once you start processing real numbers, but the first few items in each column will need special attention. Use the formulas in Fig 5 or 6.

Transfer

In my setup, data is collected using the laptop computer. The spreadsheet runs on my main computer. I use *FastWire Link* (provided on the Minisport's ROM-disk C:) to transfer the data files since disk types are not compatible. If you collect data using the same computer your spreadsheet runs on, this step in the post-processing will be unnecessary. For a computer-to-computer data transfer through serial ports, a null-modem cable is required. I strongly suggest you use one with hardware handshaking (not just 3 wires), since you'll be rewarded with speeds in excess of 115-kbit/s. The wiring for a suitable null-modem cable is shown in Table 5.

Import

At this point, you have an ASCII file on disk, with one frequency value per line. There is some extraneous text at the beginning and end of the file. The goal is to get this imported into the spreadsheet program in one vertical column.

Decide which template you'll use, make a copy, and load that spreadsheet. Position the cursor to cell A1.

Use the command from Table 4 to place the data in cell A1 and below. If there was more than one item per line in the ASCII file, separated by commas, additional columns (B, C, D, ...) would be used, but in our case all the data on each line is stuffed into the first column. Some may be blank lines. Some may be text. Most will be numbers.

Use the sort command from Table 4, or an appropriate equivalent from your program's documentation. Every spreadsheet I've come across allows the source and destination of a sort to be the same. Use this capability to keep all the data in column A.

Notice the location of the first item in column A that looks like good data. Page down and find the last good data. Move this block of good numbers to cell B16 and below. Delete any rows below the new numbers. The goal is to have only new data extend down to the bottom. If you wish, go back and blank the left-behind cells. This is particularly important if you're short on memory. If you have to delete column A to blank it (*AsEasyAs*), be sure to insert it back (now blank) or else the whole spreadsheet will be shifted over a row from my future cell references.

Clean Up

At this point, you may be interested to see the raw data in graphical form. There are two reasons. First, you'll gain appreciation of the expanded-scale histograms toward which we're working. Secondly, strange or outlying numbers will stand out and can be investigated for validity. Use the Line Graph command (Table 4) to plot only column B (row 16 and below) as the first variable. If you can, turn the point marker option off, showing only the line between data points. Fig 7 is an example of a typical graph.

It is a line plot of the sorted data, column B in Fig 4. A lot of duplicate frequency readings show up as a horizontal line. Comparing the vertical bumps to the vertical axis indicates

```
Handicounter Histograms - Extracted non-dups
1994 Brian J. Mork, KA9SNF          M1EXTR.WKS

1. /File,Import,Values,{fid}
2. /Data,Sort,Datarange,Primarykey,Ascending,Go
3. Move good data to B16 and below
4. Data,Question,{Input¦Output¦Extract}
5. /Graphic,View



Criterion    Freq
                1
                                          13
     Freq     Seq#      Freq      Seq#  #Freqs
  94.0276        1   94.0276         1       1
 94.35168        2  94.35168         5       4
 94.35168        3  94.46998         7       2
 94.35168        4  95.36445         8       1
 94.35168        5    95.932        11       3
```

```
C13 = +B16<>+B15
C16 = 1
C17 = 1+C16

F14 = @SUM(F16..F999)
F16 = 1
F17 = @MAX(E17-E16,0)
```

```
OE Handicounter Binned Histogram
1994, Brian J. Mork, KA9SNF          M1BIN.WKS

1. F5:Goto A1, then /File,Import,Values,{fid}
2. /Data,Sort,Datarange,Primarykey,Ascending,Go
3. Delete bad rows & move from Col A to B14
4. /Data,Question,Extract
5. Manually fix E14,F14..F15
6. /Graphic,View


"Bins are equal to or less than..."

                         0.15                  13
     Freq      Seq#      Freq      Seq#    #Freqs
  95.7183         1      95.7         0         0
  95.8829         2     95.85         1         1
  95.9126         3        96         3         2
  96.0217         4     96.15         5         2
  96.0768         5      96.3         7         2
```

```
C16 = 1
C17 = 1+C16

D14 = 0.15
D16 = 95.7
D17 = +D16+D$14

E16 = 0
E17 = @VTABLE(D17,B$16..C$999,1)

F14 = @SUM(F16..F999)
F16 = 0
F17 = @MAX(E17-E16,0)
```

Fig 6—These two displays are equivalent to Fig 5 except they are for an *AsEasyAs* spreadsheet rather than one built in *Supercalc*.

Table 5

Universal null-modem cable from MLHacker #4. The first column is pin numbers of a female DB-9 connector (the "loner"). The parenthetically paired numbers give the connection between another female DB-9 and a paired male DB-25 at the far end of the cable. A "-" by the pin number indicates connection, otherwise don't connect a wire to that pin. Notice pin 6 at all connectors loops back to the same connector (different pin) in addition to extending down the cable.

[9F]—— 4' 7cond cable —— [9F]— 5" 7cond cable —[25M]

0	(0 1)	Frame Ground is not used
1-6	(1-6 8-6)	Each feeds its received DSR into CD, also
2-	(3-2)	Data to loner
3-	(2-3)	Data from loner
4-	(6-6)	DTR from loner goes to others' DSR
5-	(5-7)	Signal Ground
6-	(4-20)	DSR to loner from others' DTR
7-	(8-5)	RTS from loner to others' CTS
8-	(7-4)	CTS to loner from others' RTS
9	(9 22)	Ring Indicator not used

Table 6

Telix configuration options for paced data acquisition. Italicized items indicate to use your chosen variables. The *10* selects 1 sec between data samples. The *filename* is your selection of where the data is to be saved. ONLYCR.CR is the file of only CRs sent to the M1 at a steady rate.

Baud rate	Alt-P,D
CR→CR+LF	Alt-O,Term,F,Yes
Data Rate	Alt-O,Ascii,F,*10*
Open capture	Alt-L,*filename*
Get data	Alt-S,Ascii, ONLYCR.CR,...
Close capture	Alt-L,Close

Table 7

MSDOS DEBUG script to make a paced data-acquisition file. You may enter these commands manually while running DEBUG, or enter the commands into a file (eg, MAKECR.TXT) and from the MSDOS command line enter "DEBUG < MAKECR.TXT". In either case, you'll get a file by the name of ONLYCR.CR, which is subsequently uploaded to the M1 using a paced ASCII upload.

```
F 0100,1100 0D          ;Put lots of CRs into memory(4097, actually)
R CX
1000                    ;Number of CRs to save (4096 hexidecimal)
N ONLYCR.CR             ;Select the future file name
W                       ;Save the CRs to the file
Q                       ;Exit DEBUG
```

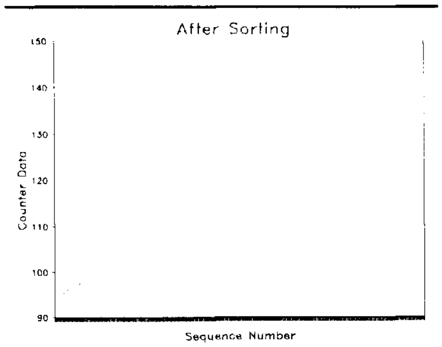


Fig 7—A line graph of the sorted raw data. This form is not very intuitive. Improvements are possible.

that frequency data ranging between 90 and 150 MHz was obtained.

The graph will consist of horizontal plateaus, with step-ups every so often when moving from left to right across the graph. Each step-up is associated with a new value that showed up repeatedly during the monitoring section. The horizontal distance of each plateau indicates how popular this frequency was. If the graph appears to be too flat (no vertical height), you probably have one or more points with really far-out frequency values making the graph autoscale the vertical axis so large that all the interesting data is squashed into a horizontal line. Far-out frequencies may actually be desirable. For example, you may have been listening to military frequencies in the 300-MHz band amidst commercial FM background noise. In this case, the graph will jump suddenly from around 100 MHz up to around 300 MHz.

It is important to realize that you have no control over the range of frequencies. Think about that again. With a spectrum analyzer, you select the range over which you want information. With frequency counters, which are inherently wide bandwidth, the best you can do is select an appropriate amplifier (10 to 200 MHz or 200 MHz to 1.2 GHz on the M1) and install filters on the input. A resonant antenna provides additional frequency filtering.

The bumps in Fig 7 do indicate that something interesting is happening. Moving over to the extracted columns of data and doing a different type of plot makes the data much more usable. Making data usable is the goal of folks who study data visualization.[8]

By looking at this graph and scrolling up and down through column B, select data points you need to get rid of. There may be some or there may be none. Many were already left behind in column A. Eliminate these cells by deleting entire rows or moving groups of numbers upward toward cell B16.

Be sure to not delete any rows above row 18, since these contain nonrepetitive formulas in other columns. All formulas in row 18 and below are all duplicates of each other and, if you delete a few dozen rows, you can just repopulate the rows with replications from those remaining.

After column B is cleaned up, extend column C down to the new bottom of column B.

Condensing Data

With histograms, instead of showing each frequency *reading* as a new spot on a plot, each *frequency* gets one location on the horizontal axis. For each frequency, you'll plot a vertical line with a length that indicates how many occurrences of that frequency were obtained. If you integrate a histogram (add up all the area under the curve), you'll obtain the total number of samples taken. If nothing else, this greatly increases the information density and makes a much more familiar display of the data. The only information you'll lose is the order in which the data was acquired.

Another variety of histogram lets you specify certain bins into which you sort each sample. For example, you may wish to count how many samples fall into the ranges 100.0 to 100.1 MHz, 100.1 to 100.2 MHz, etc. I began using this type of histogram to generate expanded scale plots and to get around some data-gap artifacts I'll talk about later.

When doing binned histograms, it's important to keep bin sizes the same in order to keep the visual appearance of the graph accurately conveying the distribution of data collected. Any nonstandard or variable bin sizing should be done for only some exotic reason and carefully annotated. This point is often overlooked by many social/population type graphs you'll find in the newspaper or other uncritiqued sources.

Extracted Histograms

Be sure cell C13 has the formula B16<>B15 (+B16<>+B15 for *AsEasyAs*). A "1" will show up because this is true (no matter what the first frequency is, it's not equal to the column label!). Use the extraction command from Table 4 to fill columns E and F. Specify all of the data in columns B and C as input and an equal number of cells in columns D and E as output. Extend the formula from F17 down to the last row filled in column F and force a recalculation of the entire spreadsheet.

To plot out a histogram, follow the guidance in Table 4 under "Histo Graph." Fig 8 is such a plot. Each frequency data selects the X (horizontal) location of a data point, and the number of times that frequency was read

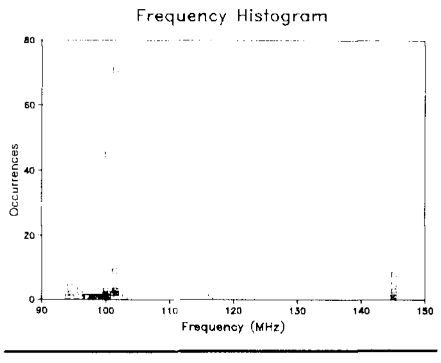


Fig 8—Extracted histogram using point markers. The strong presence of certain frequencies is discernible.

from the meter selects the Y (vertical) location of the data. The data set is really just a bunch of [X,Y] locations.

But this graph doesn't convey information at a glance. It can be improved. By *not* plotting a marker at each point, but rather connecting the now invisible points together with a line, Fig 9 looks much better. You are looking at *connections between* plot points, not plot points themselves.

Your plot may have very sharp peaks or very broad-based peaks. In general, collecting for more time gives more accurate peak shapes. One caution is in order: If column G is all ones, (only one occurrence of each frequency value), your graph will be nothing but a horizontal line! You should collect data for a longer time or decrease the resolution of the meter during collection.

Overly precise existing data can be used by rounding it to two or fewer decimals of precision. For instance, generate a new column A, equaling the rounded values from column B. In *SuperCalc*, ROUND(B*/r/*,2) would round values to two decimal places, or you could brute force it with a formula such as INT(B*/R/**100+0.5)/100. After making column A, copy the *values only* back into column B. If you're using *AsEasyAs*, you can recurse the data right back on top of itself and skip the entire column A step. For example, put @ROUND(B16,2) into cell B16. Do another extraction and you should see some piling up of similar frequencies.

Another problem may be just not getting enough values to work with. Fig 10 shows the results of a data collection that gave only one occurrence of each very precise number. After rounding to two decimal places, the graph still looks awful because peaks are built from only 1 or 2 readings. More rounding would make only more stark, blocky peaks. A larger quantity of raw data was needed.

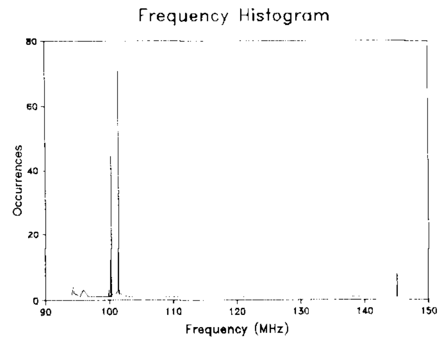


Fig 9—Extracted histogram using no markers, but rather lines between adacent points. Compare to Fig 8. I think this is the best format to look at extracted nonduplicate data.

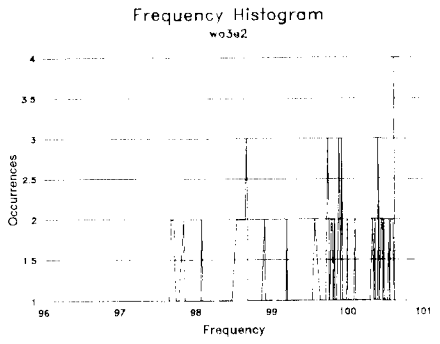


Fig 10—I collected 3 minutes of ambient FM station signals, but this was not enough. No caressing of the data would yield a satisfactory plot. I simply needed more data!

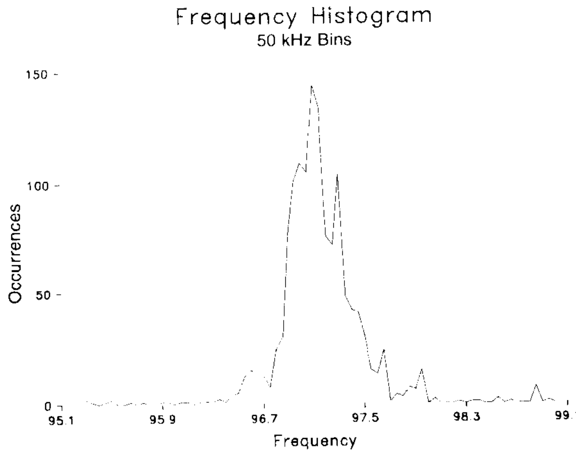


Fig 11—Expanded commercial FM range using 50-kHz bin sizes.

Binned Histograms

In the sample plots already shown, it's obvious that the signals most heard by my M1 were frequencies in the commercial FM band, collected around 95 to 100 MHz.

I wondered if the meter was actually resolving individual stations. Realize the meaning of this question. I am not asking whether the M1 can detect each of the stations. Of course it can.[9] The problem is that the M1 (not the user) selects the range of frequencies with the extracted histogram procedure described in the previous section. You could just amputate the top and bottom of the sorted list prior to extracting the frequencies, but this would discard data you may want later. Besides, it would be a waste collecting information you know you're just going to throw away.

The way I found to accomplish the goal of resolving individual stations is to define data bins over the range I want to plot and let the spreadsheet count occurrences into each bin. The bottom half of Figs 5 and 6 show the spreadsheet layout.

I wanted to see an expanded version of 95.1 to 100 MHz from Fig 9. A 10-minute collection of data ensured I had lots of samples, catching even rare frequency values. I followed the same procedure up to the extract step discussed above. Instead of extracting values into column E, I entered 95.1 into D16 and then defined each successive row as the previous row plus the value in D14. The LOOKUP command allows you to scan down a range of numbers, stopping at the last number less than the specified value. The value of the adjacent cell is then returned (sequence number in this case). Subtracting the previous LU return gives the number of rows between them—in other words, the number of frequency occurrences "in the bin" between the two specified frequencies.

Fig 11 shows the result of this procedure. Each peak has been shifted to the left by some amount less than a bin width, since frequency data gets dropped into a bin if it is less than the right bin edge. A strong peak at 99.01 MHz would show up at 99.5 MHz if you used 500-kHz bins. When using binned histograms, you have to choose the resolution of the plot (width of the bins). Looking at the same data with finer resolution indicates finer structure. Compare Figs 11 and 12.

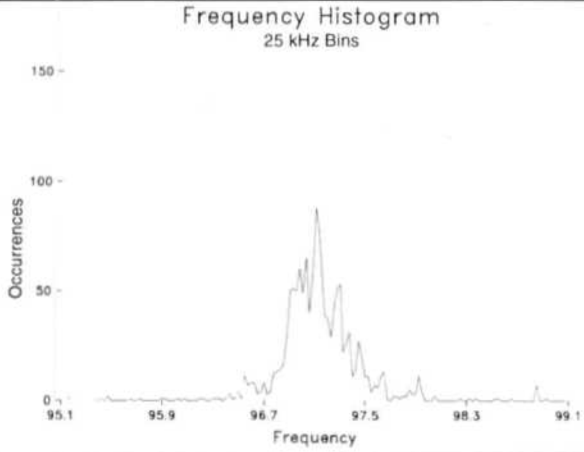


Fig 12—Expanded commercial FM range using 25-kHz bin sizes.

Photo 4— I collected some of the data for this article on a small hill surrounded by a clear horizon, in order to get away from any RF sources.

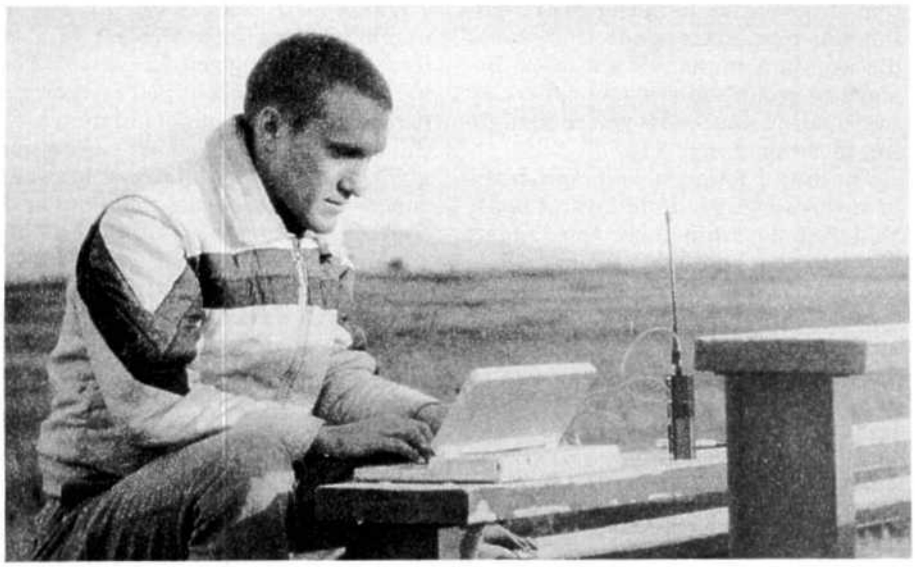


Photo 5—The system I describe is a totally portable system under battery power, good for several hours of logging data. Here I am collecting the numbers for Figs 14 and 15.

Band Gap Effect

When using extracted histograms, what happens if there is a gap of frequencies where no readings were obtained? The spreadsheet plotting logic would still connect the two points on either side, making a long sloping line that does not represent intermediate frequency readings. The effect can be dramatized by intentionally deleting a band of frequencies from a data set. Compare Fig 13 to Fig 9. Such sloping artifacts can be eradicated if you plot your histograms as bar graphs.

Pervasive Commercial FM?

One of the first additions you should make to your handicounter is a commercial FM station notch filter. Other hams in the local area have used the Radio Shack #15-577 successfully to eliminate intermodulation distortion with wideband scanners, so I tried it, as shown in Photo 3. I'm not sure it has a great effect in this application. The acquired count values are different with the filter in place, but it's not clear the FM stations are being notched out. The filter trap is designed for nominally 75-Ω systems, not the 50-Ω handicounter impedence. Maybe that had an adverse effect.

I went to an elevated location with an unobstructed view of the local horizon (Photos 4 and 5). The intent was to see how powerful commercial FM stations may affect counting against an RF background. I was more than a mile away from any RF emitters—definitely out of the range of near-field effect shown in Optoelectronics product literature. With a scanner rubber ducky installed, the M1 bar graph illuminated 8 to 10 segments, or a variable –38 to –34 dBm total RF level.[10]

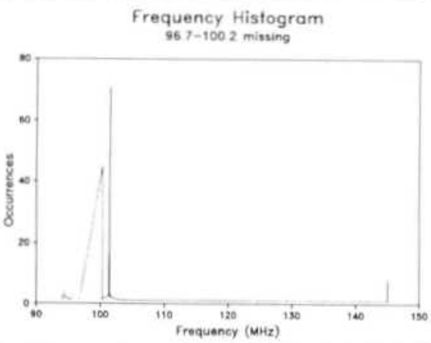


Fig 13—I intentionally created the slope artifact by deleting frequencies from Fig 9 in the range 96.7 to 100.2. The effect here is exaggerated to show you what to watch for in real data.

I took data with the notch filter in place first, watching a steady bar-graph reading of –36 dBm. These are shown in Fig 14. The approximately equal levels of background noise, compared to the no-filter ambient RF check, was the first indication that things were acting unexpectedly. Afterward, I took the no-filter data shown in Fig 15.

I used the CR send-file method with *Telix* to collect 1000 points, each at ½-second intervals. *Telix* counts the lines sent during the ASCII send on screen, providing a direct indication of how the collection is progressing. At the end of the collection, an "upload finished" beep lets you know data collection is done. I manually opened and closed the capture file.

I know FM stations are spaced at least 200-kHz apart. Where are all these peaks coming from? Strong local stations for my QTH are shown in Fig 16. Notice there aren't even any stations at the frequencies obtained. I suspect these are artifacts of the M1 sampling process, rather than true spectral energy at those frequencies. If this is true, and the M1 gives a 50% bar-graph reading on *nothing* countable, I must recommend using the M1 with only a very narrow bandwidth antenna to try to cut down on the noise! To the M1's credit, with the digital-filtering option engaged, it did record nothing, but I'm slightly miffed by the 50% bar-graph reading.

My conclusion is that the band of peaks I saw were not FM stations at all. It's background M1 counting that drifts up and down dependent on factors other than the ambient RF environment. Cumulative RF signals 10 to 2400 GHz gave the bar-graph indications, and there were no signals for the counter to lock onto. Lesson learned? If you're in the shadow of what you want to measure and not within a city block or two of a commercial transmitter, you probably don't need to worry about interference. Remember: keep the desired signal 15 dB above integrated ambient noise. Cut down on ambient noise with a narrow bandwidth antenna.

Summary

I described two data collection schemes using Optoelectronics' new hand-held counters, two communication programs and two spreadsheet programs. Hardware interfaces will vary depending on exactly what computer you have, but all are documented here or in referenced articles. Special attention was given to the Minisport laptop, an ideal computer for this application. There's plenty of room for your improvisation in *BASIC*, *C*, *C++* or whatever language you prefer. I've presented reusable details to guide you through adaptations to your software. I've tried to convince you that tweaking how you look at data is as important as the data itself.

Keep me informed of what you do. Contact information is provided. If you request a response via the mail, please include an SASE.

Notes

[1]Bergeron, Bryan, "Choosing a Digital Frequency Counter," *QST*, December 1992, pp 42-44, 80.

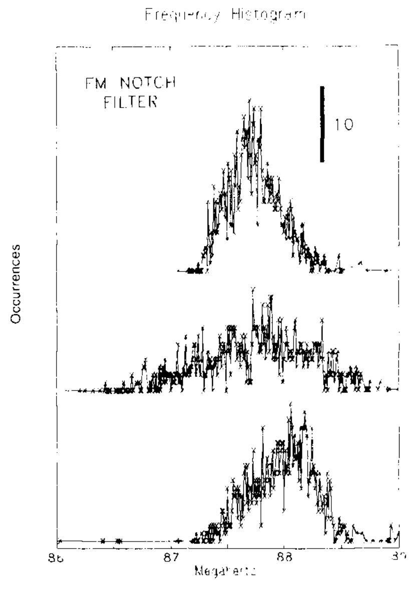


Fig 14—Three consecutive 1000-point sets of data, collected at ½-second intervals with an FM notch filter in place. The location had a clear horizon, with no significant RF emitters for at least a mile. All three graphs are plotted at the same vertical expansion; the bar shows the height of 10 readings.

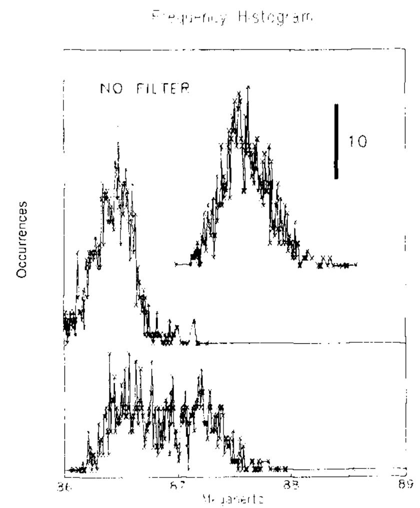


Fig 15—These data were taken under circumstances identical to Fig 14 except the FM notch filter was not used. Fig 14 and this figure are plotted by *AsEasyAs*, which didn't provide an option for turning off the point markers.

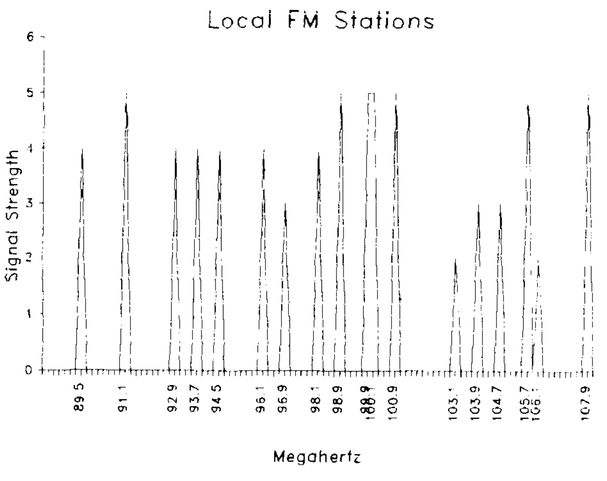


Fig 16—I recorded bar-graph readings (1 to 5 LEDs illuminated on the front panel of a Carver FM tuner) of local FM stations and manually entered the data to generate this plot. This plot is from a dot-matrix printer. Compare with other *Supercalc* output that came from a drum pen plotter.

[2] Thorn, Damien, "Electronic Privacy: An Introduction to Interception," *Nuts & Volts*, June 1993, p 74. "Electronic Privacy Part 2: Surveillance Devices and Countersurveillance Methods," *Nuts & Volts*, July 1993, pp 103-106.

[3] Feeney, F. Kevin and Payne, Andy, "Poor Man's Packet," *73 Amateur Radio Today*, August 1991, pp 8-14. *The PMP Newsletter*, distributed on Amateur Radio packet networks, is compiled by WB9BBC@WA9KEC.WI.USA.NOAM.

[4] *The Minisport Laptop Hacker* newsletter is distributed on Internet and Amateur Packet Radio by KA9SNF@WB7NNF.#EWA.WA.US. It's archived at HAM-SERVER@grafex.sbay.org and at ftp.cs.buffalo.edu.

[5] Thorn, Damien, "Radio Frequency Monitoring," *Nuts & Volts*, August 1993, pp 18-21; Mork, Brian J., "Optoelectronics M1 Product Review" and "Optoelectronics 3000A Product Review," released March 94 on amateur packet radio and Usenet news groups.

[6] Lenihan, John F., "A Parallel Expander for the PC," *Circuit Cellar INK, #37*, August 1993, pp 38-43.

[7] Blackburn, Wallace R., "Everything You Always Wanted to Know About Hardware for Computer-Controlling Modern Radios," *QST*, Feb 1993, pp 37-41.

[8] Tufte, Edward R., *The Visualization of Quantitative Information*, Graphic Press, Cheshire, CT, 1983.

[9] It has a sensitivity on the order of –50 dBm. This broadband sensitivity can't compete with a tuned receiver sensitivity, but it's certainly good enough for local broadcast stations. The problem is that for a good reading to be received, the counted signal must exceed the sum of all other signals by 10 to 15 dB. Not trivial.

[10] Optoelectronics claims the 16-segment bar graph indicates 0.4 to 12 μV RF at 150 MHz. Actual tests show my meter's bar graph indicates 0.5 to 20 μV when receiving a 2-meter signal. This corresponds to –53 to –21 dBm.

Points of Contact

Contact	*Product*
Brian J. Mork Opus-OVH 6006-B Eaker Fairchild, WA 99011 KA9SNF@KA7VV.#ewa.wa.us Internet: bmork@opus-ovh.spk.wa.us	Minisport Laptop Hacker series BBS 509-244-9260 in FILES section: /public/computer/mlhacker.zip Internet ftp ftp.cs.buffalo.edu: /pub/ham-radio/mlhacker.zip Internet e-mail server: ham-server@grafex.sbay.org
QST ARRL 225 Main St Newington, CT 06111-1494 Tel: 203-666-1541	Referenced articles Back issues, $3 each.
Microchip Technology, Inc 2355 W Chandler Blvd. Chandler, AZ 85224-6199 Tel: 602-786-7200 Fax: 602-786-7578	PIC microcontrollers: 16C5x, 16C71, 16C84, 16C64, $4.00 and up
Parallax, Inc 3805 Atherton Rd, Ste 102 Rocklin, CA 95765 Tel: 916-624-8333 Fax: 916-624-8003 BBS: 916-624-7101	PIC 16Cxx Development Tools Programmer, $179 Emulator, $299 Combo "True Flight" $349
Optoelectronics 5821 Northeast 14th Ave Fort Lauderdale, FL 33334 Tel: 305-771-2050 Orders: 800-327-5912 Fax: 305-771-2052	3000A Handicounter, $329 + $10 s&h M1 Handicounter, $229 + $10 s&h CX12 interface, $89 + $4.45 s&h
Maxim Integrated Products 120 San Gabriel Dr Sunnyvale, CA 94086 Tel: 408-737-7600 Fax: 408-737-7194	RS-232/TTL converters. Classic - MAX232 Min parts - MAX233 or MAX233A Uses Minisport port shutdown mode - MAX222 Low power - MAX220 Hardware CTS fake - MAX243
Nuts & Volts Magazine T&L Publications 430 Princeland Ct Corona, CA 91719 Tel: 909-371-8497 Fax: 909-371-3052	Referenced articles Back issues, $5 each.
Trius, Inc PO Box 249 N Andover, MA 01845-0249 Fax: 508-688-6312 Orders: 800-468-7487 Voice: 508-794-9377	*AsEasyAs* v5.5, $69 + $6 s&h Shareware available on Buckmaster's HamCall CD-ROM or Internet ftp at oak.oakland.edu: /pub/msdos/spredsht/asa55c-1.zip /pub/msdos/spredsht/asa55c-2.zip

Contact	*Product*
Computer Associates 1240 McKay Dr San Jose, CA 95131 Tel: 408-432-1764	*SuperCalc* v4.0 Check recent ads for street price.
deltaComm Development PO Box 1185 Cary, NC 27512 Orders: 800-859-8000 Other: 919-460-9313 Fax: 919-460-4531 Tech: 919-460-4556	*Telix* v3.22, $59 + $10 s&h Shareware available from oak.oakland.edu directory /pub/msdos/telix/tlx322-1.zip /pub/msdos/telix/tlx322-2.zip /pub/msdos/telix/tlx322-3.zip /pub/msdos/telix/tlx322-4.zip
DataStorm Technologies PO Box 1471 Columbia, MO 65205 BBS 314-449-9401	*Procomm* v2.4.3, $50 + $3 s&h Shareware on Buckmaster's CD-ROM or Internet ftp at oak.oakland.edu: /pub/msdos/procomm/pcm243.zip /pub/msdos/procomm/pcm243u.zip /pub/msdos/procomm/pcm243d.zip
Tiger Tronics 400 Daily Ln PO Box 5210 Grants Pass, OR 97527 Tel: 800-822-9722 Fax: 503-474-6703	Model BP-1 (PMP/Baycom clone), $49
Graphic Press Box 430-A Cheshire, CT 06410 Tel: 800-822-2454	The Visualization of Quantitative Information, $40 ppd. Envisioning Information, $48 ppd.
73 Amateur Radio WGE Publishing Co PO Box 931 Farmingdale, NY 11737	Referenced articles
Circuit Cellar 4 Park St Vernon, CT 06066 Voice: 203-875-2199 Fax: 203-872-2204	Referenced articles Back issues, $4 + $1 s&h
Surplus Trading Corp PO Box 1082 2700 No M-62 Benton Harbor, MI 49022 Voice: 616-849-1800 Fax: 616-849-2995	Minisport surplus computers $30-$100+, depending on condition

RF

by Zack Lau, KH6CP/1

Mode-S Receive Converter

The goal of this project is a simple yet high-performance 13-cm receive converter optimized for 2401-MHz OSCAR Mode-S reception. The design takes advantage of recent advances in PHEMT technology to simplify the circuitry while also improving performance. The finished unit checks out with a 0.33-dB NF and 31 dB of conversion gain according to the ARRL Lab's HP 346A/8970 noise-figure meter. This low noise figure, combined with a 15-turn helix antenna, achieves a gain-to-noise-temperature ratio of around 1. The converter also is small enough to be mounted at the focus of a dish with minimal blockage.

The biggest design simplification comes from the use of Hewlett-Packard PHEMT GaAs MGA-86576 MMICs, which are almost ideal in this application. The MGA-86576 has a relatively low noise figure of 1.5 dB, just the right output power of +7 dBm, a low current draw of 16 mA, and a gain of 23 dB that peaks in-band. The major disadvantage of the device is that it requires very good grounding for stability—Hewlett-Packard suggests four plated-through holes under each lead. Since this design is for amateurs to duplicate, rather than for commercial manufacturers, I looked for more practical methods of grounding the device. I considered using rivets, but these aren't quite as good as plated-through holes since they increase the distance between the device and ground. Instead, I decided to use very thin circuit board, 15-mil 5880 Rogers Duroid. This is the same stock used in my 10-GHz designs. Microwave Components of Michigan, one of the suppliers listed in Chapter 35 of the *ARRL Handbook*, has been selling this material for many years. With this thin board, excellent stability can be obtained by bending the leads sharply and running the leads through the board. Unlike those of some surface-mount devices, the leads of these MMICs are long enough to go through and bend back against the board for a good mechanical attachment.

Notes appear on page 30.

225 Main Street
Newington, CT 06111
e-mail: zlau@arrl.org (Internet)

Having decided on 15-mil board, I redesigned the hairpin band-pass filters used in Jim Davey's 2304-MHz transverter using Compact Software's *Microwave Harmonica*.[1] This is a good program for doing the analysis, since it corrects for the bends in the microstrip. Unfortunately, it's too expensive for most amateurs to buy. You might wonder whether the 2.256-GHz local oscillator band-pass filtering is adequate, with no filtering after the MMIC amplifier. In this case it is, because the mixer's local-oscillator-to-IF rejection, as well as the MMIC's lower gain at 2 meters, reduces the oscillator noise at the IF to acceptable levels. In contrast, previous MMICs, notably the MAR-8, have quite a bit of gain at low frequencies, resulting in a noticeable noise contribution unless it's filtered out. Because of the high gain of the MGA-86576, a single MMIC is sufficient to boost the filtered output of the local oscillator to the +7 dBm needed by the mixer.

An advantage to PHEMT MMICs is their significant power efficiency. The 564 to 2256-MHz multiplier draws only 23 mA, even when an inefficient LM317L linear regulator is used. In contrast, a pair of MAR-8 MMICs needs at least 70 mA. The driving power is similar: +10 dBm is sufficient to get +6.8 dBm of local oscillator output. The input circuitry is fairly straightforward. First I scaled my low-noise preamplifier design from November 1993 *QEX* for 15-mil board.[2] Then I took advantage of the stability of the PHEMT MMIC to redesign the amplifier for higher gain and lower noise figure, while maintaining unconditional stability for the cascaded amplifiers. Thus, due to the relatively low noise contribution of the MMIC, the calculated noise figure of the pair is actually better than the noise figure of the single-stage amplifier! Measurements bear this out: the system noise figure, including the loss of the band-pass filter and mixer, is about the same as the noise figure of the single-stage preamplifier. The dc blocking capacitor, C7, could probably be designed out of the circuitry, but it was left in to get a nice 50-Ω output match for preamplifier testing without the filter.

The mixer is a Mini-Circuits SYM-11, a surface mount device with a reasonable RF port impedance. Its 50-Ω input SWR is roughly 2:1. The SCM-2500 sold by Mini-Circuits is a poor choice in this application, having an RF port SWR around 8:1! This is important, since the mixer terminates the band-pass filter. A poor termination may upset the filter response, though it is certainly possible to measure the actual impedance and modify the filter accordingly. You can't just stick in an amplifier to isolate the two

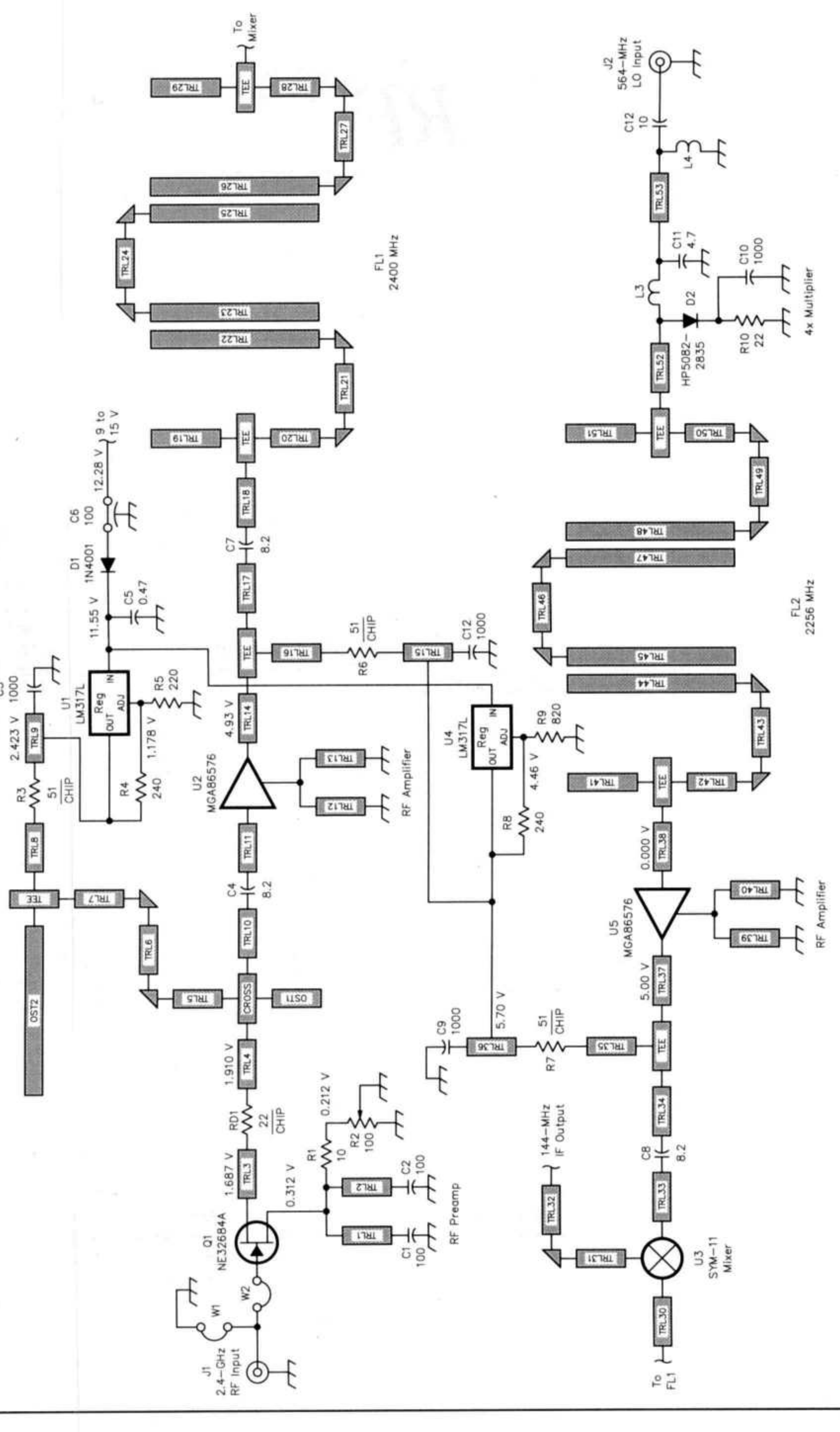

J1
2.4-GHz
RF Input
W1
W2
Q1
NE32684A
0.312 V
R1
10
0.212 V
R2
100
C1
100
C2
100
RF Preamp
1.687 V
RD1
22
CHIP
1.910 V
CROSS
OST1
OST2
TEE
C4
8.2
U2
MGA86576
RF Amplifier
4.93 V
R3
51
CHIP
2.423 V
C3
1000
U1
LM317L
Reg
OUT
ADJ
IN
R4
240
R5
220
1.178 V
11.55 V
C5
0.47
D1
1N4001
C6
100
12.28 V
9 to
15 V
C7
8.2
R6
51
CHIP
C12
1000
To
Mixer
FL1
2400 MHz
U4
LM317L
R8
240
R9
820
4.46 V
5.70 V
C9
1000
R7
51
CHIP
5.00 V
U5
MGA86576
RF Amplifier
0.000 V
144-MHz
IF Output
C8
8.2
U3
SYM-11
Mixer
To
FL1
FL2
2256 MHz
HP5082-
2835
D2
R10
22
C10
1000
4x Multiplier
L3
C11
4.7
L4
C12
10
J2
564-MHz
LO Input

Fig 1—Schematic of the microwave S-band converter.

C1, C2—100-pF ATC 100A chip capacitors. Substitution is not recommended.
C3, C9, C10—1000-pF chip capacitors.
C4, C7, C8—8.2-pF, 50-mil chip capacitors. 100-mil capacitors may work.
C5—0.1-μF or larger capacitor to keep the regulator stable.
C6—100-pF feed-through capacitor. Any value from 10 pF to 0.1 μF should work fine.
C11—4.7-pF chip capacitor.
C12—10-pF chip capacitor.
D1—1N4001 rectifier diode to protect against reverse polarity.
D2—Hewlett-Packard 5082-2835 Schottky diode. You may substitute other Schottky switching diodes.
L3, L4—4 turns # 28 enameled wire, 1/16-inch inside diameter.
RD1—22-Ω, 1/10-W chip resistor, 50×80 mils.
R3, R6, R7—51-Ω, 1/10-W chip resistor.
U1, U4—LM317L adjustable voltage regulator.
U2, U5—Hewlett-Packard MGA 86576 PHEMT MMIC.
U3—Mini-Circuits SYM-11 mixer.
W1—# 32 silver-plated wire taken from 20-gauge Teflon stranded wire (8-mil diameter). This is a loop whose ends are 200 mils apart using 250 mils of wire (plus more for the connections). One end is 55 mils above ground while the other end is grounded.
W2—# 32 silver-plated wire. This loop is between the center pin of the coax and the transistor gate. The gate lead is 20 mils long. Not counting connections, the length is 310 mils. The ends of the loop are 182 mils apart.

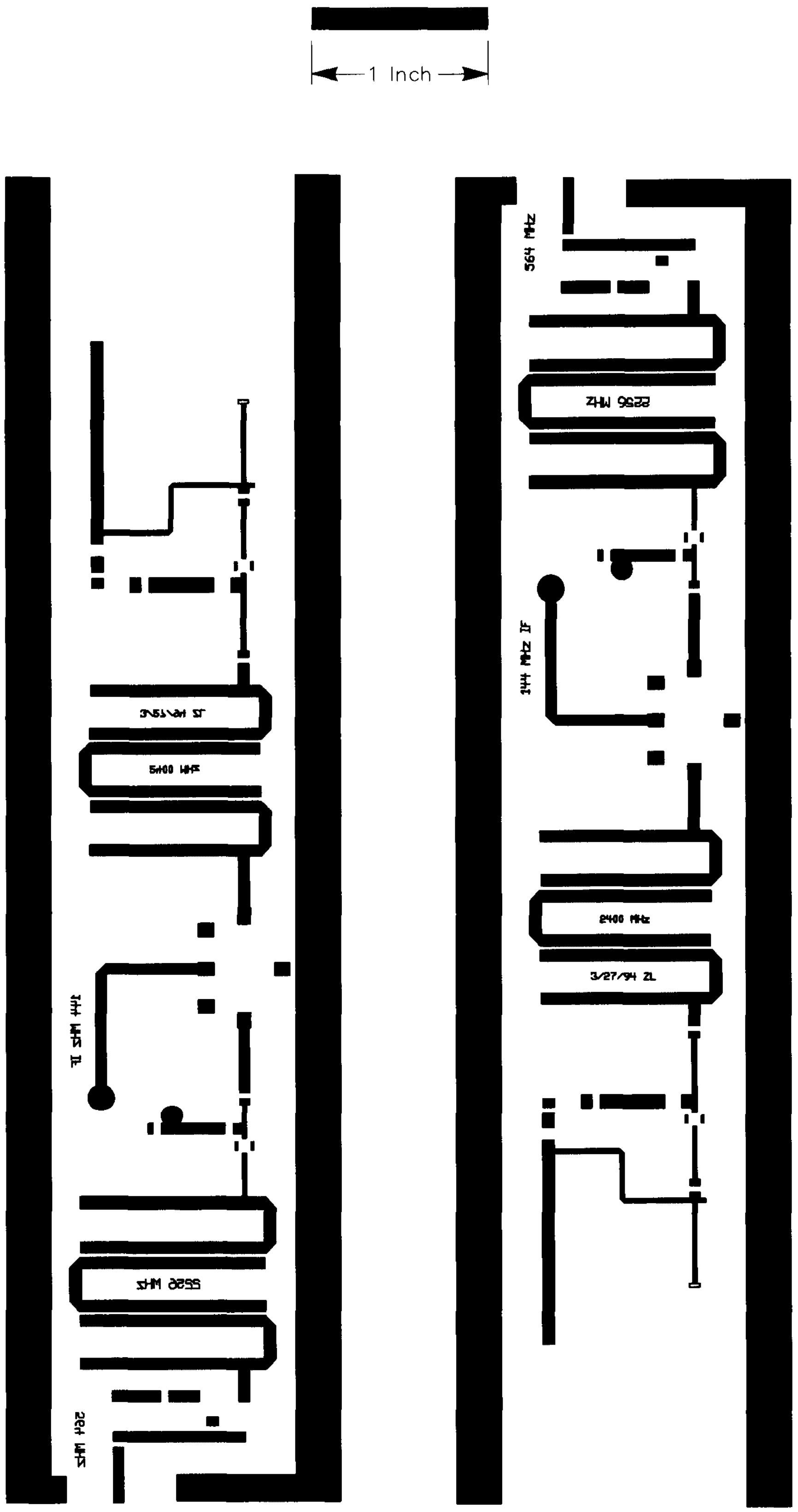


Fig 2—Etching pattern for the 13-cm converter microwave circuitry. The board material is 15-mil 5880 Duroid. (A Postscript image file is available from the ARRL BBS, 203-666-0578, or via Internet FTP at ftp.cs.buffalo.edu in the /pub/ham-radio directory.)

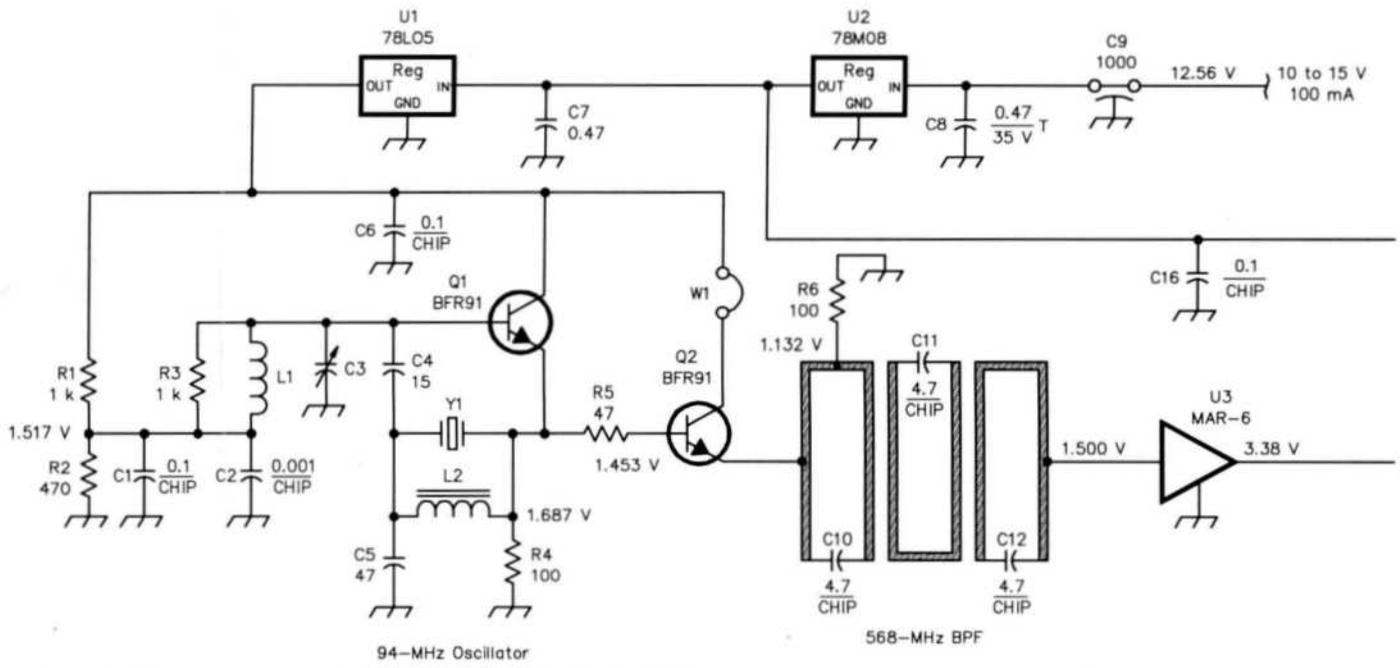


Fig 3—Schematic of the 564-MHz local oscillator.

C3—2 to 10-pF trimmer capacitor.
C4—15-pF NP0 capacitor.
C5—47-pF N1500 capacitor. A commonly available NP0 could be used, but the temperature compensation improves stability. (See text.)
C8—0.33-μF or larger capacitor to prevent U2 from oscillating.
C9—1000-pF feed-through capacitor. Any value from 100 pF to 0.1 μF should work well.
J1—SMA panel jack. At 564 MHz, connectors are optional.
L1—7 turns #28 enam wire, 0.1-inch diameter, closewound. 8 turns may work if C3 tunes low enough.
L2—14 turns #26 enam wire on a T-25-10 core. However, I've seen as many as 17 turns needed on a T-25-6 core, though 10 turns are usually specified. It depends on the crystal's shunt capacitance.
R7—270-Ω resistor. Replace with a 470-Ω resistor to run the MMIC from +12 V (no 8-V regulator).
R8—82-Ω resistor. Replace with a 220-Ω, ½-W resistor to run the MMIC from +12 V.
RFC1—8 turns # 26 enam wire, 0.10-inch inside diameter, closewound.
U1—78L05 5-V regulator.
U2—78M08 8-V regulator.
U3—MAR-6, MSA-0685 MMIC.
U4—MAR-3, MSA-0385 MMIC
W1—Wire jumper.
Y1—94-MHz crystal. International Crystal Manufacturing part number 473390. For better temperature stability, mount the crystal on the ground-plane side of the board and cover it with antistatic foam.

either, since the amplifier will generate undesirable noise at the image frequency. Assuming in your design that the impedances of the mixer are close to 50 Ω allows the most leeway for substitution.

Since the two MMICs and PHEMT draw only about 50 mA, I decided to redesign the local oscillator board as well. The goals were reduced size and current consumption. I was able to cut both in half. A major part of the size reduction was obtained by using smaller filters. This was done by capacitively loading the hairpin loops of the filter with 4.7-pF chip capacitors. This adds an interesting variable to no-tune designs. A disadvantage is that capacitor tolerances can skew the center frequency of the filter, perhaps unacceptably. On the other hand, it also can compensate for variations in board material. Thus, a bad batch of boards might well be salvaged by merely changing the value of the chip capacitors. I built five of these. They all worked, although one covered 555 to 594 MHz while another covered 542 to 570 MHz. All were etched on FR-4 board. This circuit might also be used for a 561.6 or 568-MHz LO. You multiply the former frquency by 10 for a 5616-MHz, 6-cm LO and the latter by 18 for a 10224-MHz, 3-cm LO.

I found that a major variable in dealing with G-10 or FR-4 glass-epoxy board is not the dielectric constant, as some authors have claimed. Instead, it is the board thickness. I've measured variations as large as 14% using a digital dial caliper. This is quite significant in determining the resonant frequency of a microstrip filter. Actually, this isn't surprising—does the exact board thickness really matter when a digital circuit is sitting on it? On the other hand, you can pretty much expect line accuracy from good board shops to be excellent. After all, with a 2.5% scaling error, a 40-pin IC won't fit on the board. By contrast, Rogers advertises an available thickness tolerance of ± 1.5%. Still, this isn't much better than the accuracy you can get out of a good photocopy machine that lets you vary the size in 1% increments.

The local oscillator is still the circuit used in the no-tune transverters—I've not developed a better circuit to use. However, this circuit isn't recommended if you wish to set the oscillator to a precise frequency. Like many overtone circuits, this circuit may be difficult to get running properly, since there are at least three things that can go wrong. The most insidious is a parallel resonance in the tank bypass circuit at around 100 MHz. This may prevent the circuit from oscillating properly. This problem can be prevented by changing the value of the bypass capacitors or by changing the spacing between them so the stray inductance changes. The next possibility is that the tank circuit may not resonate at the desired frequency. The easiest solution is to install a 47-Ω resistor in place of the crystal and resonating in-

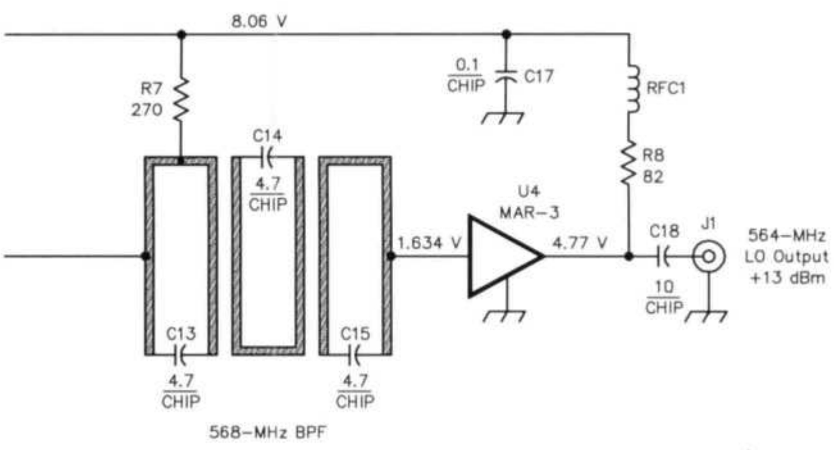


Table 1
Measured Performance of the 564 to 2256-MHz Multiplier

Input (dBm)	*Output (dBm)*
7.0	3.3
7.9	5.0
8.8	6.0
10.0	6.8
10.7	7.0
13.0	7.3

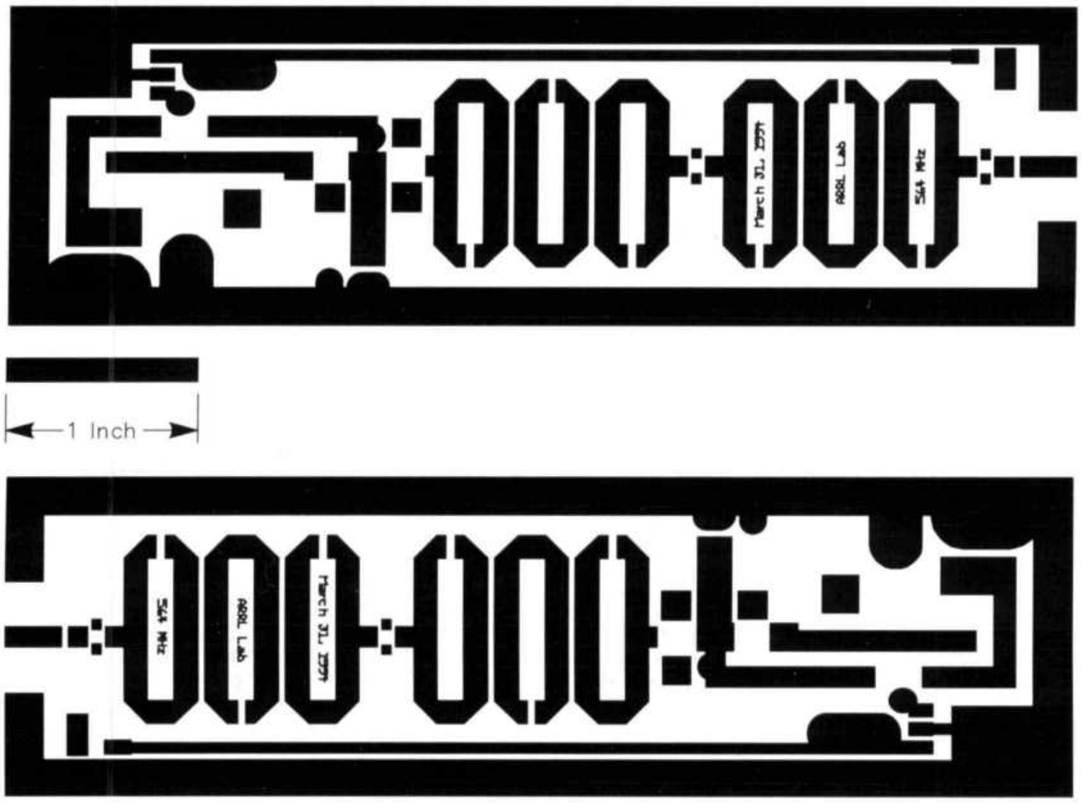


Fig 4—Etching pattern for the 564-MHz local oscillator circuitry. The board material is 1/16-inch, double-sided FR-4 or G-10 glass epoxy.

ductor to see what the tuning range of the tank circuit is. Be sure to verify that the oscillator is operating at 94 MHz. It is entirely possible that the output is near 564 MHz but the oscillator is operating at some other frequency, such as 80.6 MHz. Finally, the parallel resonating inductor has to resonate near the desired overtone. Since the shunt capacitance across the crystal seems to vary, the inductance value also has to vary. This shunt capacitance could be measured with a 1-kHz capacitance meter—if the meter reads a few picofarads accurately.

With a temperature compensating capacitor for C5, a 561-MHz local oscillator I built drifted 368 Hz over a temperature variation of 41 degrees (0 to 41°C). This performance seems typical. Unfortunately, I've not found a place to buy small quantities of temperature compensating capacitors of specific values, although you do find them in "bargain assortments" at Radio Shack.

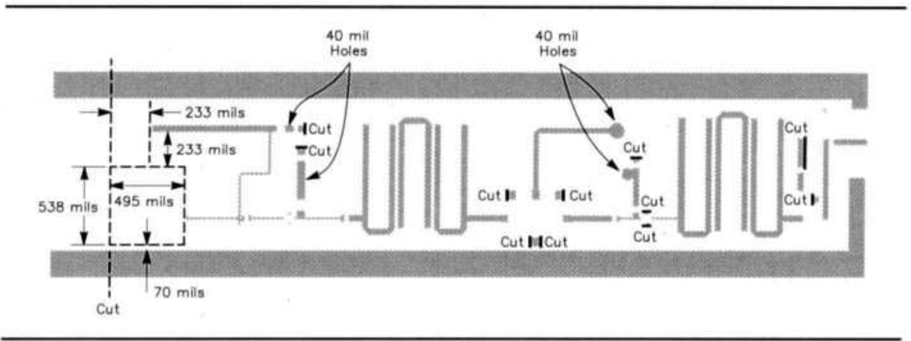


Fig 5—Cutting and drilling guide for the Teflon microwave board.

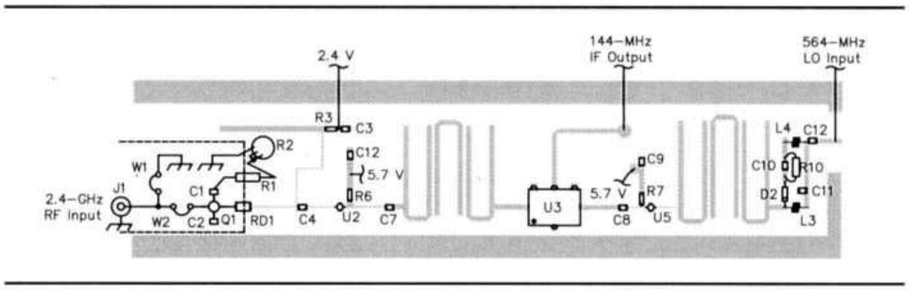


Fig 6—Parts-placement diagram for the Teflon microwave board.

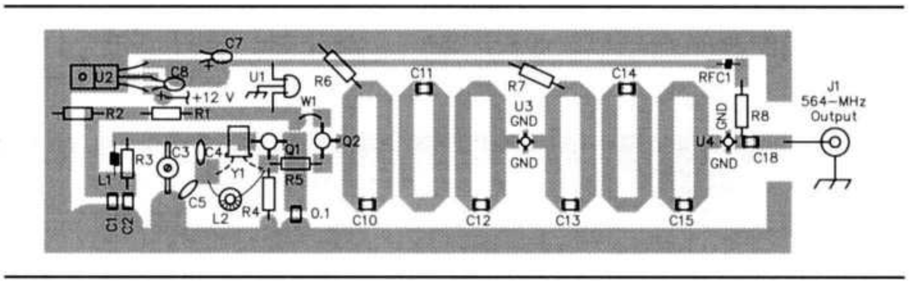


Fig 7—Parts-placement diagram for the 564-MHz local oscillator board.

Construction

After etching the Teflon board, the first thing you want to do is cut slots in the board for ground foils and transistor source leads. To make it easier to cut slots for the source leads, I've marked these with thin pads. I use a no. 11 X-acto knife with a sharp blade. The slot should just touch the outside of the pad, so that the transistor will cover the copper pads. In addition to the four slots for the transistors, there are seven slots to ground pads with thin copper foil (roughly 1 mil thick). Next, cut the "U" for the RF preamplifier. This allows a piece of unetched circuit board to be soldered in place for the ground plane. Finally, three holes are needed for the IF and dc power connections. I ran some coax between the IF connection and a panel-mount connector. I clear copper away from the holes with a hand-held drill bit. This takes a bit of practice with such thin board material.

A disadvantage of 15-mil board is the lack of mechanical rigidity. Thus, it is necessary to mount the board in a brass frame. I chose to use 0.025 × 1-inch brass strip. I attach the connectors to the brass strips with screws, then solder the strips to the circuit board. The strips without connectors are then soldered to the circuit board. An extra brass strip was added at the center of the enclosure near the mixer to add stiffness.

The preamplifier construction is almost identical to that of the design published in the November 1993 *QEX*. But instead of an SMA connector, I used a nonstandard N connector—one with a panel-mount flange similar in size to a BNC connector. One of the preamps I built didn't require tweaking, while another did. I don't know about the third; I started tweaking it before realizing that an SMA-to-N adapter between the noise source and the preamp had a half dB of loss!

R2 is set so that the voltage across R1 is 0.10 V, making the bias current of Q1 10 mA. W1 and W2 may have to be tweaked for best noise figure. It is easy to damage Q1 via electrostatic discharge while doing this, so take proper precautions.

Operating Hints

If you are going to use this converter with a typical transceiver, you need to protect the mixer from the transmitter. When you turn off many modern transceivers, they momentarily transmit! So you probably want to install a postamplifier and attenuator if you wish to connect this converter to the antenna port of a transceiver. This unit was really meant to be hooked up at the antenna, much like a mast-mounted preamplifier. There is little benefit to using an expensive GaAs FET preamplifier if it is fed with many feet of lossy coax. However, you can separate the preamp/MMIC amplifier and the rest of the converter, mounting the RF amplifiers at the antenna and having the converter in the station. Doing the calculations to determine the noise figure of the new system is recommended.

Notes

[1]Davey, Jim, WA8NLC, "A No-Tune Transverter for the 2304-MHz Band," *QST*, December 1992, pp 33-39.

[2]Lau, Zack, KH6CP, "RF," *QEX*, November 1993, pp 20-23.