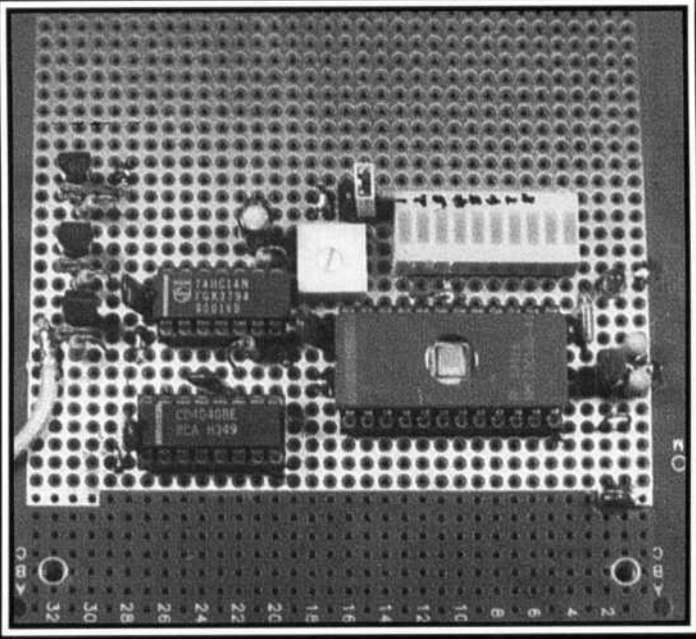
QEX

$1.75

ARRL Experimenter's Exchange

February 1994



A Universal Morse Code ID Generator

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, Deborah Strzeszkowski
Senior Production Assistants

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.

QEX
ARRL Experimenter's Exchange
February 1994
A Universal Morse Code ID Generator

About the Cover:
This test circuit for VE2HOT'S Morse code generator shows one viable construction technique.

ISSUE NO. 144

Features

February 1994 QEX Advertising Index

American Radio Relay League: 24
Communications Specialists Inc: Cov III
Down East Microwave: Cov III
L.L Grace: Cov II
LUCAS Radio/Kangaroo Tabor Software: Cov III
P.C. Electronics: 32
Peet Bros Company: Cov III
Tucson Amateur Packet Radio Corp: 8
Yaesu: Cov IV
Z Domain Technologies, Inc: 30

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*.

Empirically Speaking

219 MHz

On January 8 and 9, an ad hoc committee appointed by the ARRL Board of Directors met in Cleveland, Ohio, to discuss the opportunities created by the proposed allocation of 219-220 MHz for amateur use on a secondary basis. This possible new band ("possible" because we await the FCC Report and Order on this matter) poses both challenges and opportunities for amateurs. One major challenge lies in the fact that we expect amateurs to have secondary status in the band, meaning that we will not be allowed to interfere with the primary users (the AMTS service, which conducts communications along the waterways of the US), and we must accept any interference caused to our operations by the primary users.

Our secondary status will require new techniques for coordinating our activities with the primary users. While sharing spectrum with other users isn't new to amateurs, the 219-MHz band adds new wrinkles, as the primary user is commercial, rather than military, and requires coordination by amateurs. That was one of the issues the 219 committee addressed at their January meeting.

It appears likely that amateur stations will require the *permission* of the licensee of a primary-user station when the amateur wishes to operate a station in close proximity (50 miles) to one of their stations. Amateur stations further from primary stations, up to 150 miles, probably, will have to notify those stations of the amateur operations.

The reason the FCC is considering allowing amateurs access to this band is largely because the loss of 220-222 MHz a few years ago caused amateurs to shelve plans to use that band for high-speed (up to 56 kbit/s) intercity linking. Thus the focus of the FCC's effort to get us onto 219 is high-speed packet. That likely will result in limitations on possible amateur operations in this band, such as allowing only point-to-point operation and restricting use to data modes. And that's good, because we *need* some wide-bandwidth (relatively speaking) channels that provide the kind of propagation available at VHF.

The import of this for *QEX* readers is this: We presently have little in the way of off-the-shelf, high-speed digital radios for 219 MHz. Neither do we have a plan for making use of 219-220 MHz. If we are to make good use of these frequencies to build useful networks, we need to develop both hardware and network designs that will lead to a nationwide network. That network need not—in fact, cannot—exist wholly on 219-220 MHz. For one thing, there will be areas, mostly near major rivers, where the proximity of primary users makes it difficult to occupy the spectrum. For another, it would be foolish to waste VHF spectrum to make a hilltop-to-hilltop link of 10 miles, for example, for which low-power microwave links would be better suited. And 56 kbit/s isn't really the fastest speed we need, either.

So, the opportunity exists to design a wide-area network starting from a blank sheet of paper. Of course, to be useful, such a network will have to interface with existing packet systems. The committee views the most desirable use of 219 MHz to be a packet "backbone" network that carries data transparently, be it BBS messages, DX spotting nets, TCP/IP frames, ROSE virtual circuits, or what-have-you. *QEX* readers can contribute to this process by suggesting, and eventually implementing, viable hardware and network designs for the use of this spectrum.

You'll be hearing more about this subject in the coming months. If, as we hope, we hear good news from the FCC in the spring, it would be really nice if we had efforts well under way to put new, capable packet systems on the air at 219 MHz. The members of the 219 committee, Dakota Division Director Tod Olson, KØTO, Gordon Beattie, N2DSY, Jim Fortney, K6IYK, Joel Kandel, KI4T, and Dave Prestel, W8AJR, and ARRL staffers Paul Rinaldo, W4RI, and Jon Bloom KE3Z, are interested in hearing your thoughts about this exciting opportunity. You can send comments on this subject to:

219 Committee
225 Main Street
Newington, CT 06111
email: 219@arrl.org (Internet)
215-5052 (MCI)

This Month in *QEX*

Morse code is alive and well in Amateur Radio, and sometimes you need to generate it automatically, be it for a repeater ID, contest generator or whatever. If that's what you need to do, take a look at "The Multipurpose Morse Code Generator Circuit," by Nick Ciarallo, VE2HOT.

GPS receivers are getting cheaper almost daily. This makes for a natural marriage of GPS receivers and amateur packet stations for automatic tracking. In "Interfacing GPS or LORAN Receivers to Packet Radio," Bob Bruninga, WB4APR, shows us how to make the marriage work.

The last of three parts "All About Phase Noise in Oscillators," from Ulrich L. Rohde, KA2WEU, gives us examples of oscillator circuits that exhibit noise performance as predicted—both good and bad.

Coherent CW is a technique that has languished, despite its obvious advantages for weak-signal work, because of the need for high-stability, high-accuracy frequency references. Bill de Carle, VE2IQ, has developed "A DSP Version of Coherent CW (CCW)" that eliminates the need for specialized radios.

In this month's "RF" column, Harold Price, NK6K, discusses more about FEC, with the able assistance of two leading amateurs: N6RCE and N6GN.—*KE3Z, email: jbloom@arrl.org.*

The Multipurpose Morse Code Generator Circuit

Need to generate Morse code keying automatically? This simple circuit will adapt easily to your needs.

by Nick Ciarallo, VE2HOT

Some time ago, I built a 220-MHz repeater. Since this was to be a low-budget project, the more home-brew equipment the better. One of the circuits necessary for the repeater was a Morse code ID circuit. Searching through amateur radio publications turned up many interesting articles, but none that I really liked. I consulted with other hams and came up with a nice circuit that, as it turned out, provided more functions than just a Morse code ID for the repeater!

This article describes the evolution of this circuit from its beginnings as a Morse code ID circuit for a repeater to a contest Morse code generator, and finally its use as an automatic Morse code generator circuit for propagation beacons.

Basic Circuit Description

The heart of the circuit is the EPROM. Over the last few years, as technology has raced forward creating larger capacity and less expensive memory devices, the availability of older memory chips has grown to such an extent that they can easily be obtained for 10 to 20 cents each at hamfests! Other advantages of using EPROMs are their capacity and reprogrammability. At 10-15 WPM a 2716 (a 2-kbyte device) can hold several minutes of Morse code information per output line and, if incorrectly programmed, can be easily erased and corrected.

The basic circuit consists of a 74HC14 hex inverter, an MC4040 binary counter and a 27C16 EPROM. Variations of this basic circuit are shown in Figs 1-3. Any type of nonvolatile memory can be used here: a 2732, 2764, 2532 or even a PROM. Of course, the circuit will have to be modified for the selected device. The 27C16 is the CMOS version of this memory device and consumes far less power than the standard device. I recommend use of the CMOS version. The clock (oscillator) consists of one section of the 74HC14. In the circuits shown here, the oscillation frequency, and hence the Morse code speed, can be set approximately between 5 and 45 words per minute. The output of the oscillator is fed to the clock input of the MC4040, the outputs of which (Q1-Q12) are connected to the address lines of the EPROM. (Only Q1-Q11 are used with the 2716.) When the oscillator clocks the MC4040 and its reset pin is low, the outputs strobe the address lines of the EPROM. Any information that has been programmed into the EPROM will be output on D0 through D7.

The EPROM's outputs (D0-D7) cannot source or sink much current, so they must be buffered before they are used to drive anything, such as the transistors in Fig 3. Remaining sections of the 74HC14 can be used to provide buffering and logic level inversion where necessary.

The EPROM's data lines are used not only for the Morse code information, but also to provide some control functions. In Fig 1, the repeater ID circuit, the Morse code information is output on D0, while D7 is used to reset the circuit at the end of the ID sequence.

The logic section of the circuit is straightforward and can be built in any number of ways. I built the logic sections of these circuits on a piece of Vero board, but any good wiring technique can be used. See the cover photo. This is the test circuit used to verify EPROM programming. Note the capacitors which are placed close to each chip to provide proper power supply decoupling. The LED bar-graph dis-

SR Telcom
Microwave Group
8150 Trans Canada Hwy
St. Laurent, Quebec
Canada H4S 1M5
email: ve2hot@ve2fkb.pq.can.na (packet)
nick@vlsi.polymtl.ca (Internet).

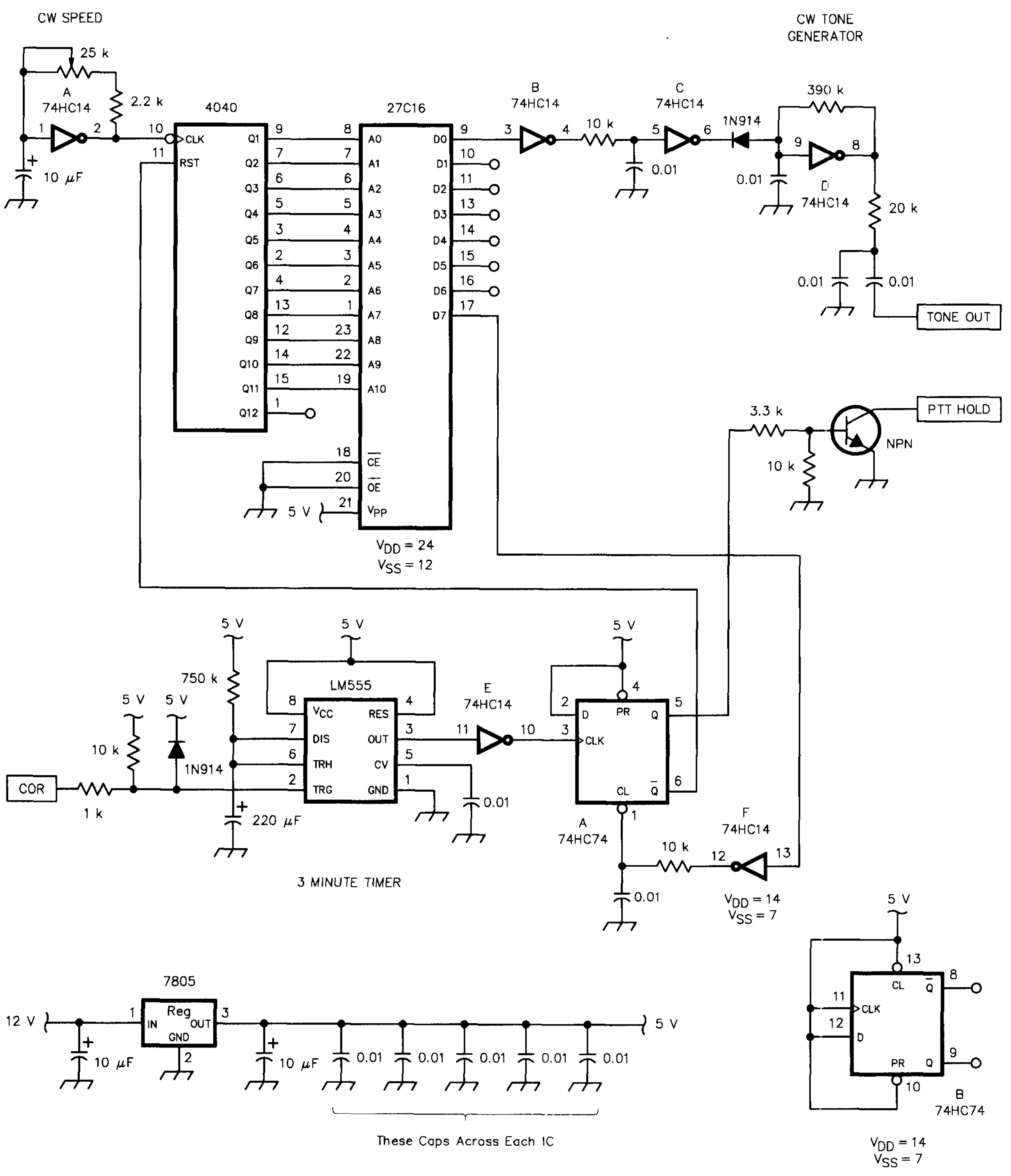
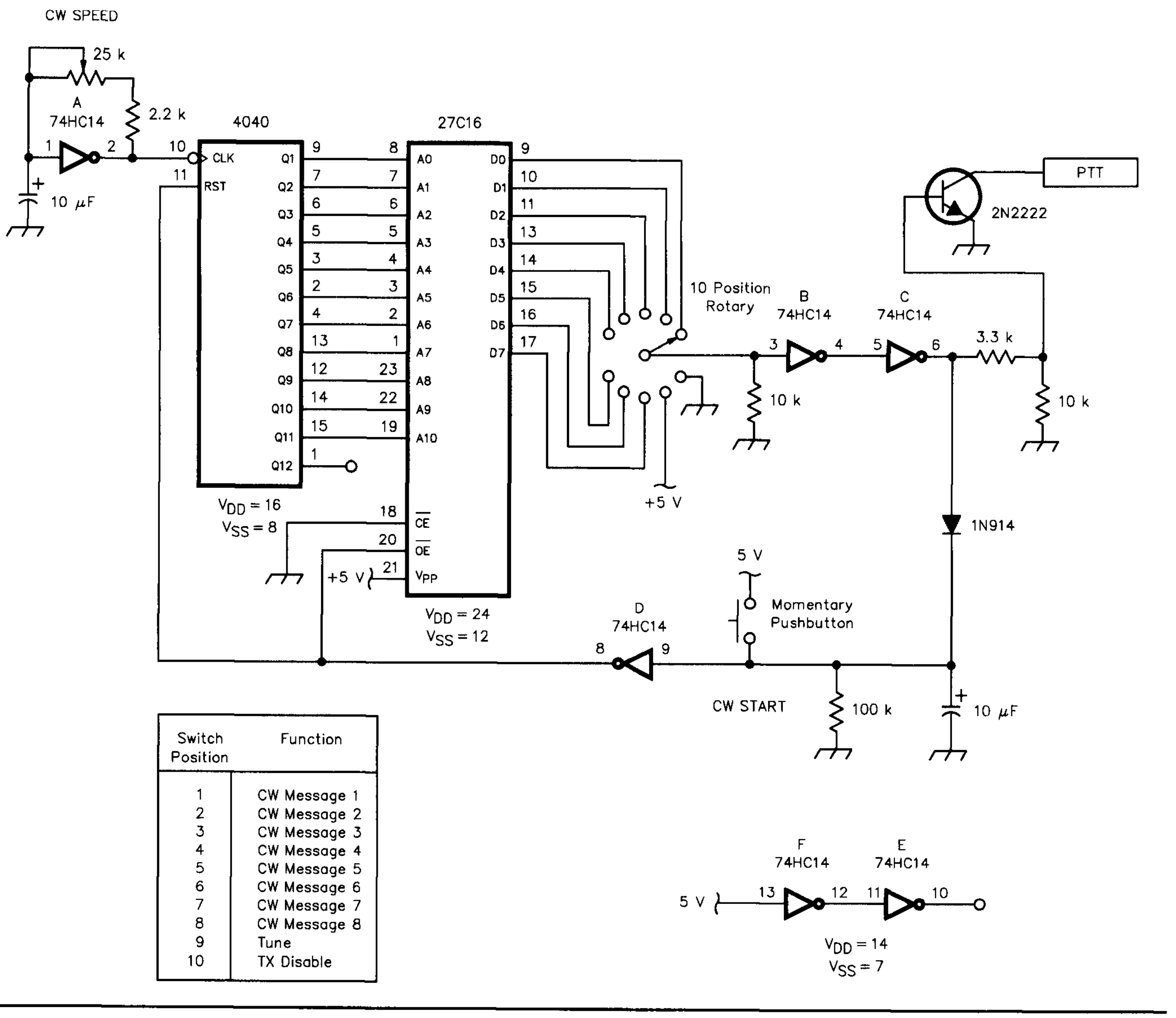


Fig 1—A simple repeater ID circuit using the multipurpose Morse code generator. Many variations are possible.

play in the photograph is used to simultaneously monitor all the EPROM output lines. This is easier than checking one output at a time and, in the case of the automatic beacon circuit, shows the state of the control lines with respect to the Morse code message.

ID Circuit

The circuit lends itself well to use as a repeater IDer. In this case, the receiver COR signal, a one-shot timer circuit and some "glue" logic were used in the design of the circuit. Refer to Fig 1 more details. The repeater ID information is programmed in D0, with D7 programmed to clear the flip-flop,



Switch Position	Function
1	CW Message 1
2	CW Message 2
3	CW Message 3
4	CW Message 4
5	CW Message 5
6	CW Message 6
7	CW Message 7
8	CW Message 8
9	Tune
10	TX Disable

Fig 2—A low-cost and effective contest Morse code generator circuit. Note that the rotary switch has TUNE and TX INHIBIT positions.

which inhibits the 4040 from addressing the EPROM at the end of the ID sequence. The other output from the flip-flop is used to hold the transmitter on the air until the ID is finished. The one-shot timer provides a three-minute time-out period and is initiated when the COR signal is first activated. An audio oscillator is created using one section of the 74HC14, however a sine wave oscillator like a twin-T or phase-shift oscillator will provide a better sounding tone. The 74HC14 oscillator circuit was used to keep the parts count down to a minimum.

Since every amateur repeater system is different, the circuit will have to adapted to fit the particular application. Many variations are possible.

Contest Morse Code Generator Circuit

Fig 2 shows a contest Morse code generator. The 8-output data lines from the EPROM are connected to a rotary switch, and each bit is programmed with a different Morse code message. A Field Day example of this would be:

D0: CQFD CQFD DE VE2HOT CQFD CQFD DE VE2HOT K
D1: R R - YOU ARE 1A IN QUEBEC KN
D2: TNX ES 73 GL DE VE2HOT K
D3: QSL?
D4: QRZ?

When the CQ message on D0 terminates and there is no reply, the operator simply presses the CW START button to send the "CQFD" message again.

When programming the EPROM, make sure to leave enough space after the longest message to allow the R-C circuit on pin 9 of the hex inverter to discharge, providing the reset to stop the MC4040 from counting. Too short a delay here will cause a premature reset. This will be evident at slow (<10 wpm) Morse code speeds where the circuit will be tricked into thinking that the space between words is the end of the message. A few seconds of "dead" space are all that is required.

Automatic Morse Code Beacon Circuit

The automatic beacon circuit is shown in Fig 3. Since this circuit is an automatic beacon, the Morse code in-

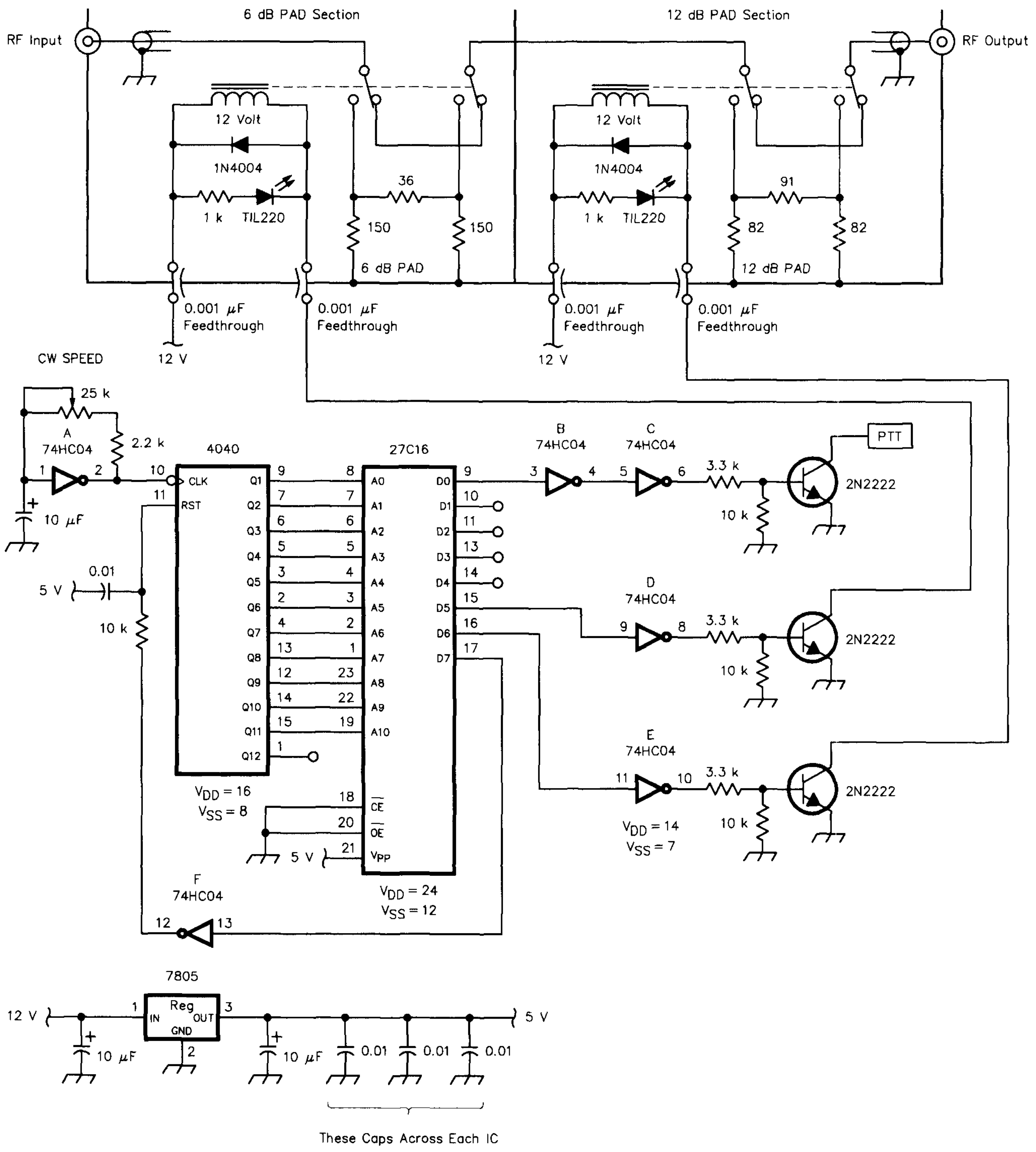


Fig 3—Schematic of the automatic beacon circuit.

formation must be sent repetitively. This is easily achieved using one of the outputs from the EPROM. At the end of the beacon text, simply program one of the outputs to provide a reset pulse. In Fig 3, the pulse is output on D7. The MC4040's reset line is connected to D7 through one section of the 74HC14 and an R-C low-pass filter. The R-C filter is needed to remove noise which may cause the MC4040 to inadvertently reset during the beacon message. Once the pulse at D7 is output, the MC4040 is reset and the circuit loops back to the beginning of the Morse code message and repeats.

The only part of the automatic beacon circuit that requires special attention is the RF attenuator section. As seen in Photo 1, the attenuators are enclosed in cast aluminum boxes. All

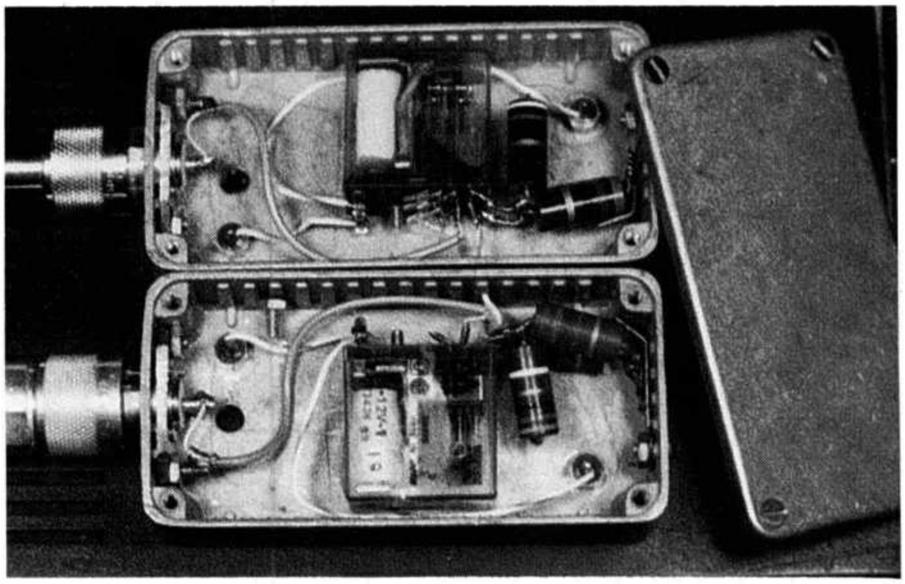


Photo 1—The finished attenuator sections. Note the feedthrough capacitors for the dc input/output lines. The diodes across the coils are not visible from this angle, but they are there. Don't forget them! (VE2BJR photo)

connections in and out of the cast enclosures were made through feedthrough capacitors to eliminate any unwanted RF coupling. This technique is used to ensure the attenuation levels are accurate when all the attenuators are in the RF path. This may seem like overkill, but it is sound engineering practice. Standard 5% carbon resistors are used in the pads. The required power rating for the resistors will depend on the transmitter output level. In this application, the maximum power output is 5 watts. Keep all RF connections and associated wiring as short as possible.

The Software

This scheme of generating Morse code is quite nice as the component count is modest, however it has drawbacks. The task of manually programming messages is time consuming and tedious. The manual procedure to generate a Morse code message is as follows:

1. Write down the required Morse code message(s) and control signals on paper.
2. Translate the Morse code message into hex.
3. Input the hex data into an EPROM programmer.
4. Program the EPROM.
5. Test the programmed EPROM Morse code message.

For this to be a useful circuit, you need to have the ability to change the Morse code message and/or control signals without too much trouble. Enter some BASIC programming.

I approached a coworker to develop a program that could convert an ASCII text file into some form an EPROM programmer could recognize. Several formats were considered, and eventually we decided on straight hex and binary formats, since the majority of EPROM programmers can use either one or the other file format. The hex file format is considered straight hex, that is, it is not formatted as an Intel, Motorola or any other hex configuration.

The result is the BASIC program MORSE.BAS. The program went through several iterations before satisfactory results were obtained. Initially, C was used as the language for the program, however since this is an amateur radio application, we decided to write it in BASIC. BASIC is a relatively well known language, and widely available to amateur radio operators; it allows for easy modification to the code, if necessary.

Program Description

In order to use the program, a few rules must be observed. The program must be told how to distinguish between the Morse code data (the message) and the control signals. Note the following ASCII string:

\7,1\\0,data\ A B C \7,0\ \7,1\

The \ character in the above ASCII string is the escape command for the program. The program executes whatever is between the \ characters as a program command.

The **0** in the \0,data\ string informs the program to put the Morse code data in the least significant bit (D0) output of the EPROM. The **data** in the \0,data\ string informs the program that whatever follows this command sequence, until it sees another \, is to be converted into Morse code. Therefore in the above example, the program converts the ASCII string **A B C** into Morse code.

The **\7,1** before the data command instructs D7 to go high, which removes the MC4040 from its reset condition. Bit line D7 will remain high until it sees the **\7,0\ \7,1** command sequence. At this point in the ASCII string, D7 will toggle low to high. Once the reset is applied to MC4040, the sequence repeats.

Note: The BASIC program was compiled using Microsoft *Quick Basic* 4.5. If this program is run under GWBASIC or some other BASIC, it may produce memory allocation errors, as well as run slowly. It is highly recommended that a compiled version of the program be used.

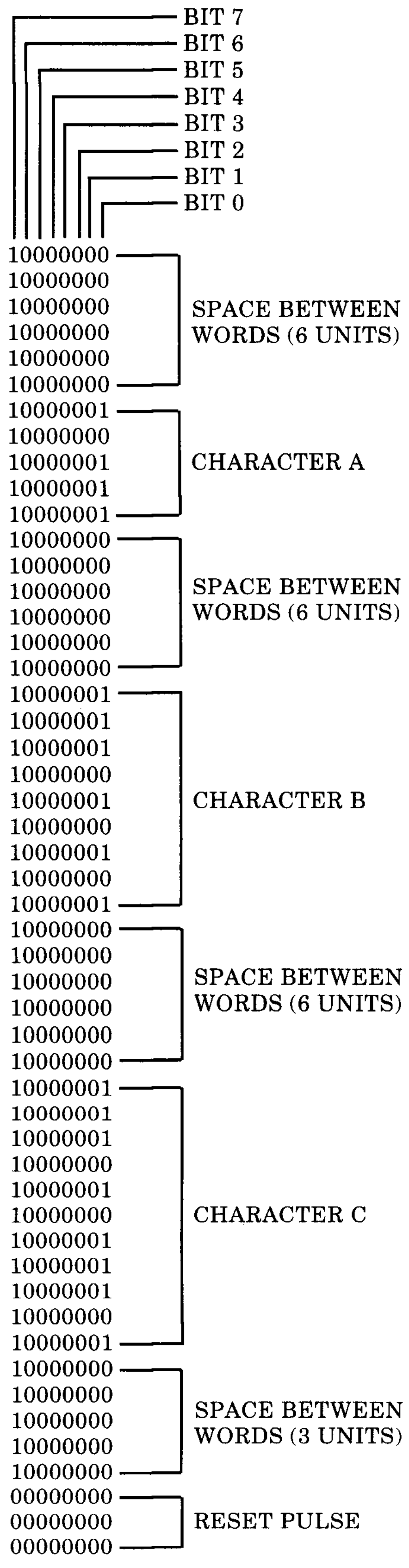
Program Use

When the program is invoked, all files that are to be converted must be in the same directory as the program files. The program has a Morse code lookup table it must reference for the code conversion. This lookup table is contained in a file called CW.DAT. This file must also be in the same directory as the program file. Using the above ASCII string as an example, a step by step procedure follows:

1. Type Morse at the DOS prompt (providing a complied version of the basic program is being used). The program will display:

```
TEXT to CW to EPROM conver-
sion program        Dec 1990
Written  for  Nick  Ciarallo
[VE2HOT] by K.C. Ng Ching Hing
________________________________
Initializing. Please wait...
Program start...
Enter filename to convert >
```

2. At this point, enter the file name to be converted. Note: the whole file name, including extension must be entered.



```
                 BIT 7
                 BIT 6
                 BIT 5
                 BIT 4
                 BIT 3
                 BIT 2
                 BIT 1
                 BIT 0
10000000
10000000
10000000    SPACE BETWEEN
10000000    WORDS (6 UNITS)
10000000
10000000
10000001
10000000
10000001    CHARACTER A
10000001
10000001
10000000
10000000
10000000    SPACE BETWEEN
10000000    WORDS (6 UNITS)
10000000
10000000
10000001
10000001
10000001
10000000
10000001    CHARACTER B
10000000
10000001
10000000
10000001
10000000
10000000
10000000    SPACE BETWEEN
10000000    WORDS (6 UNITS)
10000000
10000000
10000001
10000001
10000001
10000000
10000001
10000000    CHARACTER C
10000001
10000001
10000001
10000000
10000001
10000000
10000000
10000000    SPACE BETWEEN
10000000    WORDS (3 UNITS)
10000000
00000000
00000000    RESET PULSE
00000000
```

Fig 4

```
Enter filename to convert >
test.txt
Converting text...
```

3. The program will now ask the user if sound is needed. The user simply responds yes or no. If sound is selected, the program will ask what bit to play.

```
Do you want sound? (y/n) >y
Bit number to play ? (0..7) >0
```

In the above example, if the user wants to listen to his Morse code message, he selects bit 0 to play. The program plays the message at a speed of approximately 15 words per minute. The code speed can be changed by modifying the variable in line 1550 of the program. A larger number here slows the code speed down.

Once the program finishes playing the Morse code message, it produces the binary, hex and audit trail files. (If the sound option is not requested, the program will output the binary, hex and audit trail files immediately.)

At first, the audit trail file was used as a debugging tool, however it proved to be a useful feature so we incorporated it into the final program. Basically, the audit trail file is an exact representation of the EPROM's bit lines going through their low to high transitions as the address lines are strobed. The audit trail for the above test string would look like Fig 4.

As seen, the audit trail can be very useful and time saving. It shows the result before the EPROM (or PROM) is programmed. The Morse code weighting (dash-to-dot ratio) the program produces is 3:1, as seen in the audit trail file. This weighting is considered typical and produces easy-to-copy Morse code. The on-air tests produced favorable results.

To save a prolonged exercise at the keyboard, the compiled MORSE.EXE program, the BASIC listing and the CW.DAT lookup table are available on the ARRL BBS (203 666-0578, and via the Internet by anomymous FTP at ftp.cs.buffalo.edu, in the pub/ham-radio directory). They, along with three sample text files, are all compressed into the file TXT2EPRM.ZIP. Clearly, the software makes this a very flexible circuit with many possibilities and variations. Please contact me if you have any comments or suggestions regarding this circuit.

I would like to express my sincere thanks to K.C. Ng Ching Hing for writing the software and putting up with my many requests, Josee Leger for her inspiration and patience, Irving Lustigman for some design ideas, and to Henry Szczawinski, VE2BJR, for providing his photographic services to help complete this article. □□

Tucson Amateur Packet Radio Corp.

1994 Annual Meeting

March 4th, 5th, & 6th, Tucson, Az.

Friday (12noon):
- Registration
- Hospitality Suite
- Dinner
- Round Tables:
  - Regional Networking
  - National BBS

Saturday:
- TAPR Projects
- Contributed Papers/Presentations
- Lunch
- Contributed Papers/Presentations
- Futures Symposium
- Dinner
- Future of TAPR Forum

Sunday: Half-Day Workshop on NOS TCP/IP
by: Johan Reinalda, WG7J/PA3DIS author of JNOS

TAPR • P.O. Box 12925 • Tucson, Az • 85732 • Office (817) 382-2825
TAPR is a Non-Profit Research and Development Corporation

Interfacing GPS or LORAN Devices to Packet Radio

With GPS receivers dropping in price, consider automatic packet tracking using GPS.

by Bob Bruninga, WB4APR

Abstract

The marriage of global positioning satellite (GPS) technology to amateur packet radio is the new frontier. By its very nature, the most important aspect of any radio communication network is the knowledge of the location of all of its participants. With the price of GPS receivers set to fall under $300 this year, and with new units being the size of a matchbox, there is no reason why all mobile units, whether voice or packet, should not periodically transmit their location. This article begins with a brief overview of the rapidly evolving GPS marketplace and a description of the National Marine Electronics Association (NMEA) standard data interface. Then we address three major categories of GPS/LORAN-to-packet interfacing: the direct interface to a PC running the APRS software (see sidebar), using any GPS unit with the PacComm TNC, and using the programmable Magellian or Motorola GPS unit with any TNC.

GPS/Packet Radio History

This article has evolved radically over the last year of rapid developments in the GPS marketplace. We first began doing packet radio tracking of GPS units when Magellian GPS circuit cards became available for $445 (down from over $1000) in September 1992. Later, the Motorola GPS card came down to the same price range, and these two devices were the only ones that we could find that were cheap *and* which had user programmability so that they could be set up to operate stand-alone, with only a TNC and radio, as a tracking device. We used this scheme for our first major event, the tracking of the Army/Navy game football run from Annapolis, Maryland, to Philadelphia in December 1992 using the APRS software. We have since used these devices for several local events, and for the Marine Corps Marathon in Washington, DC.

The next improvement in tracking occurred during the summer, when we added an optional serial interface to the APRS software so that any standard GPS or LORAN receiver could be plugged into the PC's COM port, and

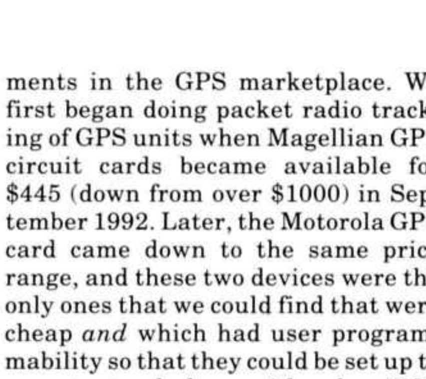


Photo 1—Photo comparing the size of a ball-point pen to the Magellian 5-volt GPS card and antenna and a PacComm Handi-Packet TNC.

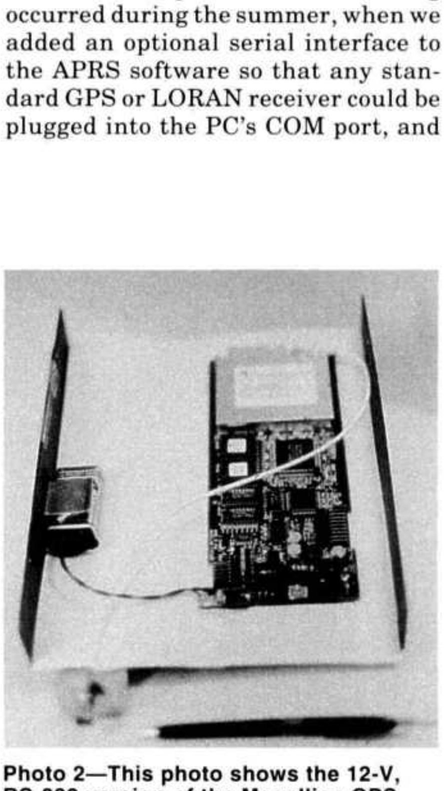


Photo 2—This photo shows the 12-V, RS-232 version of the Magellian GPS card mounted in the lid of an MFJ 1274 TNC. The only external evidence of the GPS is the antenna bracket on the back, and the two RS-232 switches on the front panel.

115 Old Farm Ct
Glen Burnie, MD 21060

Automatic Packet Reporting System (APRS)

APRS is the result of 13 years of experience with trying to use packet radio for real-time communications for public service events. Packet radio has rarely been used effectively in real-time events, where information has a very short life time. To solve this problem, APRS avoids the complexity and limitations of trying to maintain a connected network. It uses UI frames to permit any number of stations to participate and exchange data, just like voice users would on a single voice net. Stations that have information to contribute simply transmit it, and all stations monitor and collect all data on frequency. Secondly, APRS recognizes that one of the greatest real-time needs at any special event or emergency is the knowledge of where all stations and other key assets are located. APRS accomplishes the real-time display of operational traffic via a split screen and map displays. The major display subsystems and a number of other minor displays are as follows:

Latest Beacons—This display maintains a list of the latest UI frame received from each station. In effect, this is a multi-station, one-line broadcast message system. Since the lines contain the latest time of receipt, this display shows if a station was on line within the last few minutes.

Position—This display maintains a separate list of the position of each station. Each position report can also contain a brief comment, weather report, DF bearing, or other important information.

Maps—Maps to any scale from 0.5 miles up to the whole world can be displayed. Stations are instantly displayed when they transmit a properly formatted position beacon. Stations with a reported course and speed are automatically dead-reckoned to their present position. You can center the map anywhere in the world.

Messages—In addition to the beacon text, which is used to broadcast information to all other stations on the net, there is an operator-to-operator message capability. Any station can send multiple one-line messages to any other station. On receipt, the messages are acknowledged and displayed on the bottom of the receiving station's screen until the operator kills them.

All Traffic Log—This display is a time sequenced log of every new beacon or message line sent. Beacons are logged only the first time they are received.

When Heard—This display maintains a count of the total number of transmissions from each station per hour. These statistics are ideal for displaying the connectivity of the network over varying paths, such as HF, or to see when stations enter and leave the net.

Station Tracking

Although APRS automatically tracks mobile packet stations interfaced to GPS or LORAN navigation, the graphic capability of the maps works perfectly well with manual tracking or with grid squares. Any station on HF or VHF that includes its grid square in brackets as the first text in the beacon text will be plotted by APRS. Additionally, any station can place an object on its map, including itself, and within seconds that object appears on all other station displays. In the example of a parade, as each checkpoint with packet comes on line, its position is instantly displayed to all in the net. Whenever a station moves, the operator just updates the station position on the map, and that movement is transmitted to all other stations. To track other event assets, only one packet operator needs to monitor voice traffic to hear where things are. As that operator maintains the positions and movements of all assets on his screen, all other displays running APRS software display the same positions.

Using Dumb Terminals in an APRS Network

APRS automatically computes positions by latitude and longitude for all stations, based on the position of the cursor on their map display. No GPS or special hardware is needed in most cases. Even the simplest of portable packet stations with dumb terminals can report their positions if a pre-printed map which has a latitude/longitude grid reference is

the mobile station could see itself tracked on the map. By making this interface conform to the NMEA-0183 interface standard, any GPS or LORAN device could be used instead of just the Motorola and the Magellian. The only disadvantage to this arrangement is the requirement for a PC (laptop) on board each mobile to be tracked.

Next, PacComm added a universal GPS interface into each of its amateur TNCs. This capability reversed the previous situation by now permitting *any* GPS to be used with only PacComm TNCs instead of *only* Magellian/Motorola GPS units with *any* TNC. This makes it easy to assemble autonomous tracking devices but has the disadvantage that the PacComm amateur implementation only reports position, and not course and speed, and is not compatible with LORAN devices. (LORAN is caught below the falling GPS price curve, and LORAN devices can be purchased for less than $200.) Hopefully, PacComm and other TNC manufacturers will improve on this idea and include provisions in TNCs not only to accept both GPS/LORAN course and speed, but also to permit manual entry of position by fixed stations.

The final observation, as this article goes to press, (December 1993), is that GPS and LORAN prices have dropped by 50% just in the last year to below $500 and $200 respectively. Magellan confirms that their new, smaller 10-channel circuit card GPS will be $295 in single unit quantities by May, although it has only a binary output instead of the NMEA-0183 output.

NMEA-0183 Interface

NEMA has developed a serial interface standard for all marine electronics devices. This standard uses 4800-baud serial data (no parity, 8 data bits and 1 stop bit) with ASCII characters. Every data format begins with a $ and ends with a carriage return and line feed. The first two characters after the $ indicate the type of device (GP for GPS, LC for LORAN), and the next three characters indicate the data format. For GPS and LORAN, there are several important data formats as follows:

$GPGLL - Position only ($LCGLL for LORAN)

$GPVTG - Course and Speed ($LCVTG for LORAN)

$GPGGA - GPS Position and altitude

$GPRMC - GPS position, course and speed

$LCRMA - LORAN position, course and speed

After these headers comes the data, separated by commas. The fields of interest to amateur-radio applications are shown here:

made available to all net participants. The operator of a portable station just looks at the map and enters his latitude and longitude into his beacon text. Using the same map, he can plot with pins the location of all other stations as he sees their position reports go by.

Space Applications

APRS could be a solution to the effective use of orbiting terrestrial style packet radio digipeaters in space such as on the shuttle, MIR, AO-21 and ARSENE. The problem with space digipeaters is the saturation on the uplink channel which makes the use of a normal connected protocol impractical. For a connected contact, a total of five successive—and successful—packet transmissions are required. Not only does APRS reduce this to one packet, but it also capitalizes on the most fascinating aspect of the Amateur Radio hobby by displaying the location of those stations on a map. If all stations inserted their latitude and longitude or grid square as the first characters of their beacon text, everyone within the satellite footprint would see the location of every successful uplink. Since the shuttle is a rapidly moving object, the locations of successful uplink stations will move progressively along the ground track. No changes onboard the shuttle or MIR would be required to implement this capability!

Weather Station Reporting

APRS also supports an optional weather station interface to the ULTIMETER-II home weather station. The wind speed, direction, temperature and rainfall are inserted into the station's periodic position report. The station shows up on all APRS maps as a large blue dot, with a white line showing the wind speed and direction. Several automatic APRS weather reporting stations, supported with additional manual reporting stations, can form a real-time reporting network in support of SKYWARN activities across your state.

Fox Hunting or Direction Finding

APRS is an excellent tool for direction finding (DF). The X command (cross fiX) has been added to permit displaying the intersection of bearing lines from a number of reporting stations. The DF stations can either be manually placed on the map, or they can automatically be plotted if the DF bearing is included in their BText position report. All stations running APRS can simply hit the X key to display the intersection of these bearing lines. As of APRS version 3.00, there is an optional Doppler DF interface for automatically plotting and transmitting instantaneous DF bearings.

Protocol

Since the object of APRS is the rapid dissemination of real-time information using packet UI frames, a fundamental precept is that old information is less important than new information. All beacons, position reports, messages and display graphics are redundantly transmitted, but at longer and longer repetition rates. Each new beacon is transmitted immediately, then again 20 seconds later. After every transmission, the period is doubled. After ten minutes only six packets have been transmitted. After an hour this results in only three more beacons; and only three more for the rest of the day! Using this redundant UI broadcast protocol, APRS is actually much more efficient than if a fully connected link had to be maintained between all stations!

Conclusion

APRS is the newest frontier in amateur packet radio and it parallels the explosive growth in the GPS technology. Although there are now—and will be—some amazing software products in the consumer marketplace, none have been written from the ground up to support amateur packet radio. Even without GPS, APRS is a fresh approach to using packet at real-time events.

APRS runs on any PC with a CGA, EGA or VGA video card and is available as shareware or for $24 from the author. The option for direct connection of your own GPS is an additional $9.

$GPGGA,TIME,LAT,N,LONG,W,,,, AntHeight,,,

$GPGLL,LAT,N,LONG,W

$GPVTG,COURSE,T,MAGNETIC,M, SPEED (kts or statute),SPEED (km),K

$LCRMA,,LAT,N,LONG,W,,,SPEED (kts),COURSE(true),MagVar

$GPRMB,TIME,,LAT,N,LONG,W, SPEED(kts),COURSE(T),DATE, MagVar

In a typical GPS/LORAN device, any, some, or all of these data formats are output approximately once every two seconds. This is why you cannot just connect *any* GPS or LORAN to *any* TNC: there is just too much data for a packet channel. Also note that the NMEA-0183 interface is only "sort of" compatible with RS-232. NMEA-0183 does not require both positive and negative voltages, although it allows them, so you may have to add a 5-kΩ pull-down resistor to a negative voltage on the output of the GPS/LORAN receiver to make it compatible with RS-232. Usually, this can just be a single added resistor in the serial port connector between TXD and RCD, since the TXD line rests at a negative voltage while not sending data.

Direct APRS PC Interface

The easiest implementation for GPS is to just plug the NMEA-0183 output into a PC running APRS with the optional GPS routines built in. This places your station on the map, James Bond style, and you can drive around town and see yourself go. If you also have your TNC interfaced, everyone else on frequency will also see you move. APRS allows you to set both your own map refresh rate and the packet transmission rate. We have found that one position beacon about every two minutes is fine for long trips, and about one every minute is good for special events on a shared frequency. To reduce channel loading, APRS decays this period out to once every 10 minutes or more if the station is not moving. The disadvantage of this arrangement is the need for a PC on the moving vehicle.

PacComm GPS Interface

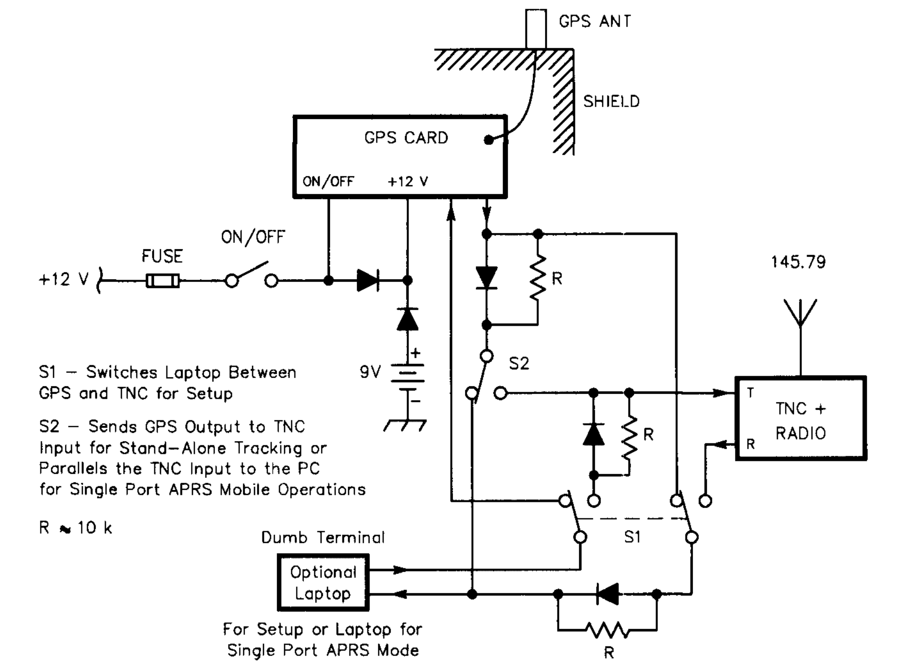
PacComm TNCs with firmware version 3.1 or later have a GPS ON command that allows you to hook up any GPS device to the serial port. With this function turned on, the $GPGGA position report will be automatically inserted into your beacon text. Just set your beacon period and away you go! This is a significant advance for the application of this technology in Amateur Radio since it obviates the need for the special Magellian and Motorola GPS devices and makes it easy for any ham boater to use his existing GPS. The only disadvantage of this arrangement is that it only sends the $GPGGA data for position, not the $GPVTG course and speed, and does not yet include the LORAN formats. PacComm sells an amateur version of their vehicle tracking system, which is a TNC with built-in GPS receiver, to satisfy off-the-shelf applications.

Future TAPR-2 Compatible Interface

It would be best if PacComm and other TNCs using this option didn't require a specific NMEA sentence but were more generic. Howie Goldstein, N2WX, suggests a MASK command that tells the TNC to look for any user-defined character strings (such as $GPGGA, etc) and then output those strings as a UI frame. This command option should accept a minimum of two such definitions. In addition, the user should be able to specify the periodicity for this data. This way, the TNC could be told to transmit only the GLL and VTG sentences once every two minutes for a mobile, or the RMC once every 15 minutes for a boat, or the GGA and VTG sentences once a minute for a balloon or aircraft. Secondly, there should be a special "location text" setting in every TNC for the user to manually enter his position. This LText is handled just like BText, but is separate. So, you would also need an L E n command for setting how often you want the location text to be transmitted. Placing location reports in their own LText frame would keep BText free for other applications. APRS already separates position-report UI frames from beacons. The LText should be a free text format so that it is compatible with any future specific formats (currently APRS parses $GPGGA, $xxGLL, $GPRMC, $xxVTG, APRS, both 4- and 6-digit grid squares and a future 8-character compressed L/L format, and there will probably be others, too). The minimum L period, L E 1, should result in a transmission once every minute. Also, each time the L E n command is executed, the LText should be transmitted immediately to start the new timing period so that a user can report a new position immediately.

Autonomous GPS/LORAN Tracking Devices—Mobiles Without PCs

This section describes how to interface a GPS/LORAN device directly to a TNC—without a PC in the middle—so that very compact tracking devices can be assembled for tracking moving objects. To be usable on a shared 1200-baud AX.25 packet link, the GPS device must permit the user to specify not only the data reporting rate, but also which of the dozens of NMEA formats to use. Our initial search found that some LORAN devices with a separate "printer" port can be configured by the user to output a report once every *N* minutes, or even hours, but the only inexpensive GPS devices that we found with this user programmability were the Magellian and Motorola OEM cards.



GPS ANT
SHIELD
GPS CARD
ON/OFF +12 V
ON/OFF
FUSE
+12 V
145.79
R
S2
9V
S1 – Switches Laptop Between GPS and TNC for Setup
S2 – Sends GPS Output to TNC Input for Stand–Alone Tracking or Parallels the TNC Input to the PC for Single Port APRS Mobile Operations
R ≈ 10 k
T
R
TNC + RADIO
Dumb Terminal
S1
Optional Laptop
For Setup or Laptop for Single Port APRS Mode

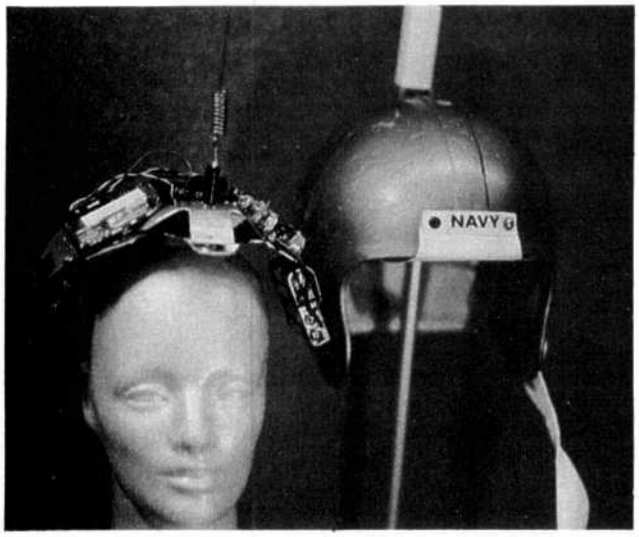
Fig 1—Interfacing the GPS card is very easy. The SW-1 switch allows the user to switch his laptop to talk to either the GPS or TNC. The SW-2 switch enables all position reports from the GPS card to go into the TNC for a stand-alone tracking device or to parallel the TNC data into the PC for single port mobile operations. The diodes and resistors allow both the laptop and GPS card to be connected simultaneously. The TNC should be in the UNPROTO-CONVERSE mode when GPS reporting is enabled. The 9-volt backup battery is also shown with its isolation diode.

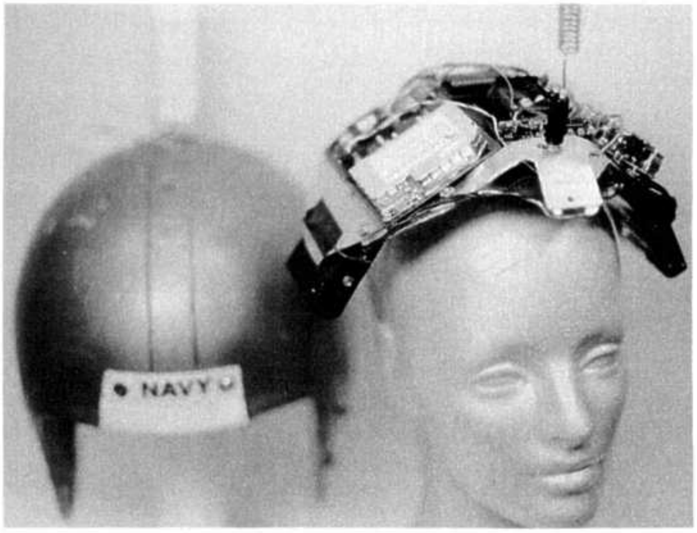
The Magellian OEM 5000 circuit board is a 12-volt (250 ma) GPS engine on a 3.5 × 7-inch circuit card that costs $445 and produces RS-232 output in NMEA format. They offer $60 (passive) and $130 (active) antennas. It has full user programmability and includes GLL, VTG and GGA sentences. The standard card only outputs altitude to 999 meters, but a version for balloons and aircraft that reports to 56,000 feet can be ordered. It can be set to output at a variety of periodicities between one second up to five minutes. The evaluation kit, including all cables and documentation, is an additional $60. Call Emiel Yakoub at Magellian, 960 Overland Ct, San Dimas, CA 91733, phone 909 394-5000.

Tom Clark, W3IWI, has found that the 12-volt Motorola OEM prototype card also has user programmability. This card includes the RMC message, which contains both position and course-and-speed data for land mobile in one NMEA sentence. The card has an NMEA-0183 output and comes with an active patch antenna for $484. Although I do not personally have one of these, the combined receiver and antenna pricing, smaller size, and availability of the combined RMC sentence is very attractive. It outputs altitudes to 56,000 feet in the GGA message. Unfortunately, the full Motorola evaluation kit is $1200. But it does include a precise timing option. Call Jennifer Spitzen at Motorola, 708 480-5699 and ask for the OEM circuit board.

An automatic vehicle tracking system can be assembled using either of these two devices set to output at a low data rate by simply connecting their RS-232 output directly into the TNC. The TNC must be placed into the UNPROTO-CONVERSE mode, and from then on, a GPS position report will be transmitted periodically. The APRS software will decode the raw NMEA position reports and plot the station on the map!

Both of these GPS cards include RTCM-104 differential correction capability in the standard price and also accept an external 3.6-volt battery to keep them alive between uses. This





Photos 3a and 3b—These photos show the smaller, 5-volt Magellian GPS card with a PacComm Handi-Packet and a small VHF transmitter, all mounted inside a football helmet for tracking the running of the Army/Navy game football from Annapolis to Philadelphia (128 miles).

allows them to come up from a power-off state quickly and get an almost instantaneous fix, as well as remembering the user specified reporting rate.

TNC Operation in the UNPROTO-CONVERSE Mode

The only problem with operating the TNC in the UNPROTO-CONVERSE mode is keeping it there. TNCs default to command mode when turned on. Until the manufacturers put an UNSTART command in their TNC to cause it to power up in UNPROTO-CONVERSE, you must either keep the TNC permanently turned on after setting CONVERSE mode, carry along a terminal to issue the CONV command, make a firmware patch to the TNC code, or wait for the TNC manufacturers to change their code! Fortunately, Howie Goldstein, who wrote the original TAPR-2 code, identified a software patch to the DRSI version of the ROM that will make it power up in UNPROTO-CONVERSE. This ROM should work in most TAPR-2 clones. I have used it in the MFJ-1274, and it should easily work in the PacComm Tiny-2. DRSI has agreed to make this ROM available at a nominal cost of $27.

Other TNC Set-Up Details

Unfortunately, the simple direct connection from the Magellian card to the TNC is slightly more complicated since that card does not have the RMC sentence and must therefore transmit both the GLL and VTG sentences to report position, course and speed. These two sentences are separated by a few milliseconds, and as a result, the TNC generates two packets, one right after the other. This is a problem if a digipeater path is used because the digipeater will begin digipeating the first position-fix packet and cover up the trailing velocity packet. To solve this problem, since most applications require a digipeater path for longer ranges, the sending TNC needs to be instructed to send packets not on receipt of every carriage return, but on a timing function. Set CPACTIME ON and change the SENDPAC character from $0D to anything else (say $01). This way, both the position-fix and velocity sentences will be sent together in the same packet one second after the last character is received from the GPS. This packet, containing two frames, will then be relayed all together by the digipeater, with no break in between. If you use the Motorola unit, which implements the RMC sentence, then only one sentence is required. Even for balloons, or when you want the GGA altitude in addition to the other sentences, the problem is not significant, since you do not need digipeaters for balloons!

Linefeeds and Flow Control

Since the GPS is sending each line with a CR/LF on the end, your TNC will end up placing the superfluous linefeed at the beginning of the next packet. To defeat linefeeds, set LFIGNORE on (for some non-standard TNCs, try linefeed suppress, LFS ON). For the Magellian, your terminal program must send CR/LF after each command to the GPS card. When you try to talk to your TNC with CR/LF, you will experience a lockup condition since the extra LF will look to the TNC like the beginning of a new command line and will hold off all TNC output. To overcome this problem, set FLOW OFF. Here are the commands which must be changed from factory defaults for most TAPR-2 TNCs:

```
ECHO OFF, FLOW OFF, LFIGNORE ON,
CPACTIME ON, SENDPAC $01
```

Dumb Terminal Setup

In order to see the command being sent to the Magellian GPS card, I configure my terminal device as half duplex. Since the Magellian GPS card needs the CR/LF sequence at the end of each command, I set the terminal to translate CR to the CR/LF sequence. In order to use the same terminal with the TNC, then, I turn ECHO and FLOW off in the TNC. My GPS/TNC box has one DB-9 serial connector and two switches: one to select whether the terminal is talking to the GPS or the TNC and the second switch to enable the data output from the GPS to go into the TNC after all configuration is complete. (See Fig 1.)

Tracking Symbols

While using the direct GPS-to-PC interface, the mobile packet station can tell his APRS tracking software which of the 28 or more different symbols to use for transmitting his position report to other monitoring APRS stations. For the direct GPS-card-to-TNC interface, however, there has to be a way for the TNC to identify its desired symbol. APRS handles this in two ways: first, APRS will assume that all stations outputting raw NMEA data that have a –7, –8 or –9 SSID are air, marine, or mobile platforms. Secondly, any of the APRS symbol designation characters can be placed at the beginning of the TNC BText surrounded by curly braces {}. Once the BText with that symbol is received, the station will then appear with the proper display symbol.

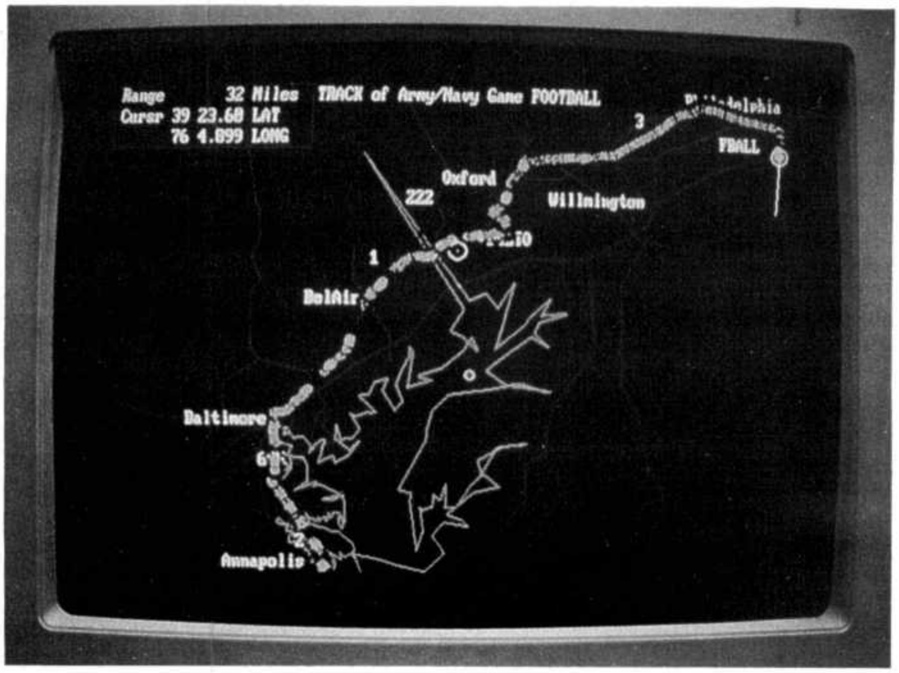


Photo 4—This photo shows the APRS software used to track the football runners from Annapolis, through Baltimore, to Philadelphia. The once-every-two-minute packets from the runners were relayed through voice repeaters for simplicity and to avoid having to change digipeater paths during the run.

Single Port Laptops

To make mobile operation of both GPS and a TNC possible on the same serial COM port for most laptops, the APRS software has a single port mode. In this mode, the software will separately recognize both the TNC data and the GPS data, although they are both coming in from the same COM port. If you have a programmable GPS, and program it to a very low data period, such as once every 30 seconds, you can just combine the data using the isolation diodes shown in Fig 1. If your GPS is not programmable, replace S2 with a momentary single-throw push button. Then, every time you want to update your position, simply press and hold the button for one second. Preferably, do this between received packets to avoid a data collision with the TNC data. In either of these modes, no special configuration of the TNC is needed since the GPS does not interface directly to the TNC. The position reports transmitted through the TNC are originated by the APRS software after parsing the raw GPS format.

Operation

With the special UNPROTO start-up ROM, and after initialization of the other TNC parameters once, all future tracking evaluations are initiated by simply applying power to the GPS, TNC and radio. In over 6 months of daily operation, I have never had to reinitialize the GPS engine. (At the seventh month, the 9-volt battery died!) Without the special ROM, every tracking evolution requires applying power, turning on a dumb terminal, and sending the TNC the CONVERSE command. Then the terminal can be removed or turned off until the next activation. If you do not have the UNSTART ROMS, be careful if you use a battery supply of C or D cells with a spring-loaded battery holder! A bicycle equipped with this system reset the TNC after hitting the first bump, and there was never time to stop and reset the TNC until the race was over. This shows the problem of the TNC not having a power up CONVERSE mode!

We have assembled a number of these GPS/packet tracking devices. In fact, the 7 × 3-inch Magellian card fits nicely against the inside cover of the MFJ 1270 or 1274 TNC. The only evidence that the TNC is GPS-equipped is the kludge on the back panel to hold the GPS antenna connector and the presence of the two switches added to the front panel that select whether the external terminal device is talking to the GPS or TNC and enable or disable GPS packet reporting. Other, smaller packages have been made using PacComm and DRSI TNCs and the TTL-only model of the Magellian GPS card, which is only about 5 × 3 inches. I shy away from this card for the casual experimenter because of the absence of any data or power supply buffering. One wiring error or static charge and you have blown a $395 card! The $445 model with onboard 12-volt regulators and RS-232 buffers is much more forgiving.

HF Tracking of Boats and RVs

Automatic GPS tracking is not just for public service events, but is a perfect way of tracking the wanderings of the large contingent of amateur radio operators with boats and recreational vehicles. We have begun to operate a position and status reporting net on both 7.085 and 10.151 MHz using lower sideband mode and 300-baud, 200-Hz shift AFSK. (Note that the 10.151 LSB signal is 700 Hz inside the band and is perfectly legal with a clean packet signal.) These frequencies assume a TNC running 1600- and 1800-Hz tones. If you are using the AEA TNCs, which are centered at 2210 Hz, then tune 510 Hz higher. Boaters can either use the automatic GPS interfaces described in this article or simply type a position report into the BText of their TNCs to report their position to all stations on the net. Stations in port could beacon about once every hour or so, while boats underway might want to beacon every 15 minutes. Mobile RVs just move their cursor on their APRS screens to the approximate location of their campground, and their position report goes out automatically. As propagation comes and goes, eventually, everyone will appear on APRS maps.

Conclusion

GPS is here to stay! As of this writing, there are more than a dozen amateur GPS mobiles out there driving around, and double that many under construction. As in the early days of packet radio, the number of stations seems to be doubling every few months! Similarly, there is no reason why the hundreds of ham boaters can't begin interfacing their navigation equipment to their radios immediately. At your next club budget meeting, instead of throwing another $500 at the repeater monster, buy the components to build a GPS/TNC tracking device into a cigar-box-size package. Then at all future public service events, you have a package with whip antenna on top that can be duct-taped to the top of any vehicle for automatic vehicle tracking. Let your imagination roam! □□

All About Phase Noise In Oscillators

Part III—Example oscillator circuits and their noise performance.

by Ulrich L. Rohde, KA2WEU

This last of three parts presents verification examples for the nonlinear mathematical approach of calculating phase noise in oscillators.

Example 1—A 10-MHz Crystal Oscillator

Fig 18 shows the abbreviated circuit of a 10-MHz crystal oscillator. It uses a high-precision, high-*Q* crystal made by companies such as Bleily. Oscillators like this are made by several companies and are intended for use as both frequency and low-phase-noise standards. In this particular case, the crystal oscillator being considered is part of the HP 3048 phase-noise measurement system.

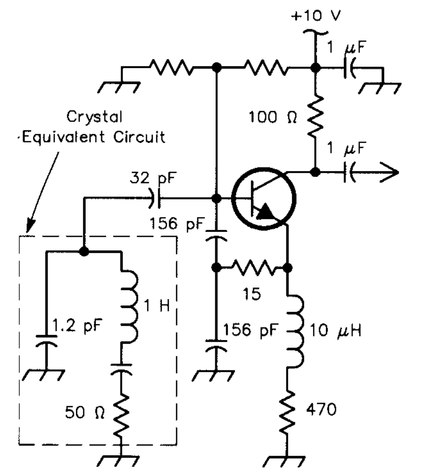


Fig 18—Abbreviated circuit of a 10-MHz crystal oscillator.

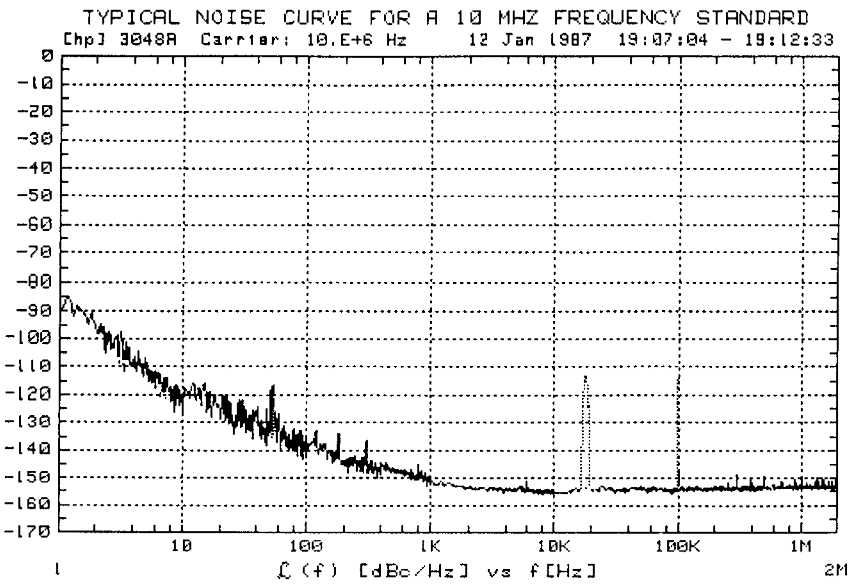


Fig 19a—Measured phase noise for this frequency standard by HP.

Fig 19a shows the measured phase noise of this frequency standard by HP, and Fig 19b shows the predicted phase noise using the mathematical approach outlined above.

Example 2—A 1-GHz Ceramic Resonator VCO

A number of companies have intro-

52 Hillcrest Dr
Upper Saddle River, NJ 07458

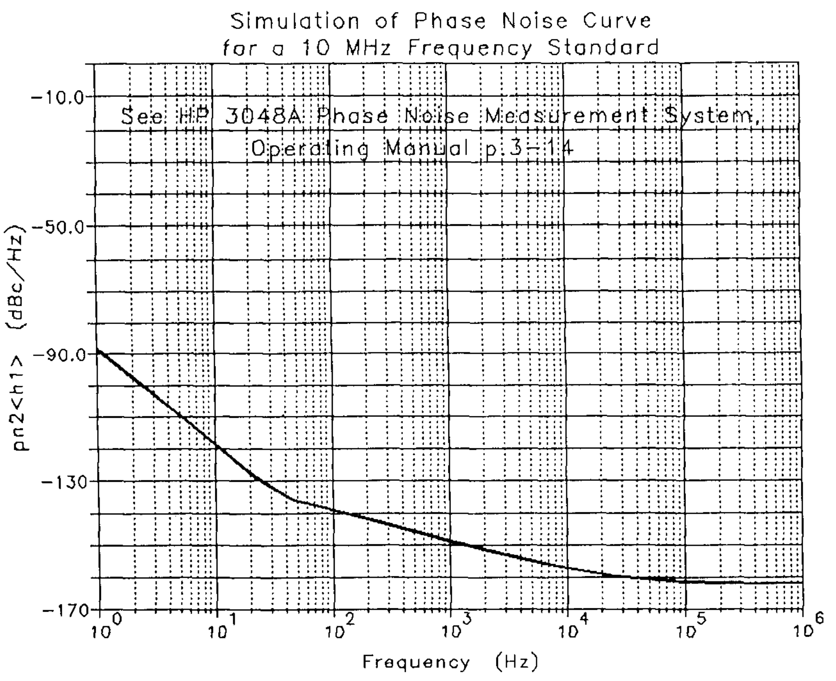


Fig 19b—Simulated phase noise of the oscillator as shown in Fig 18.

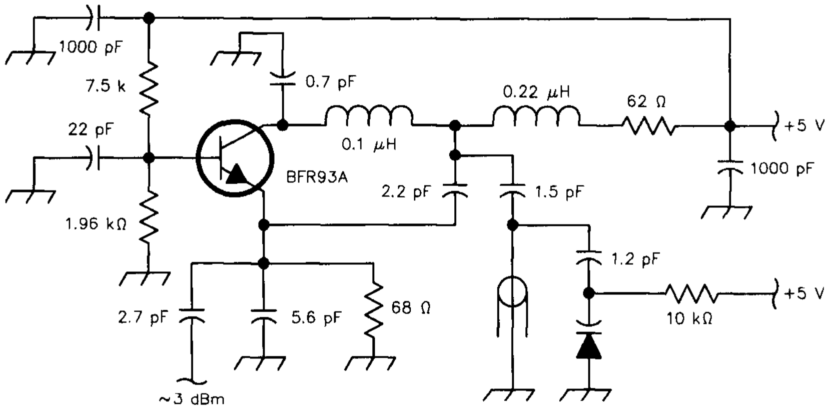


Fig 20—A typical test circuit for use in a ceramic resonator. These resonators are available in the 500-MHz to 2-GHz range. For higher frequencies, dielectric resonators are recommended.

duced resonators built from ceramic materials with an epsilon ranging from 20-80. The advantage of using this type of resonator is that they are a high-*Q* element that can be tuned by adding a varactor diode.

Fig 20 shows a typical test circuit for use in a ceramic resonator. These resonators are available in the range of 500 MHz to 2 GHz. (For higher frequencies, dielectric resonators are recommended.) Fig 21 shows the measured phase noise of the oscillator shown in Fig 20. The noise pedestal above 100 kHz away from the carrier is due to the reference oscillator, a model HP 8662 generator.

Fig 22 shows the predicted phase noise of the 1-GHz ceramic resonator VCO without a tuning diode, and Fig 23 shows the predicted phase, noise of the 1-GHz ceramic VCO with a tuning diode attached. Please note the good agreement between the measured and predicted phase noise.

Example 3—A Low-Phase-Noise FET Oscillator

A number of authors recommend the use of a clipping diode to prevent the gate-source junction of an FET from becoming conductive, thereby lowering the phase noise. Claims also have been made that the diode was necessary to obtain long-term stability. In reality, it turns out that these are misconceptions. A popular VCO circuit described in the ARRL *Handbook,* and reproduced in Fig 24 has been analyzed with and without the diode.

Fig 25 shows the measured phase noise of an oscillator of this type, and Figs 26 and 27 show the simulated phase noise of the type of oscillator as shown in Fig 24, with and without a clipping diode. Please note the degradation of the phase noise if the diode is used. (These two plots do not have the same vertical scale.) ARRL Senior Assistant Technical Editor David Newkirk, WJ1Z, found that removing the diode did not change or degrade the long-term stability, but the diode did degrade the phase noise close-in. We have, however, developed a VCO which clips the negative peaks in the sense that it prevents the oscillator from shutting off. This VCO, shown in Fig 28, was incorporated in a scheme with a digital direct synthesizer. This synthesizer will be the subject of a later article in *QST*.

The phase noise of the combined system was significantly improved. The phase noise of the oscillator, shown in Fig 29, despite having only one VCO for the total range from 75 to 105 MHz, compares well to a recent switched-range design like the synthesizer in the TS-50. Our design gives a 10-dB better S/N ratio than that of the TS-50S, shown in Fig 30, at 10 kHz and further away .

Previous authors have tried to build wideband oscillators with varying degrees of success. The VCO shown in Fig 31 violates several rules of designing a good VCO. First, resistor R2 of 68 kΩ, together with C2, provides a time constant which gets close to the audio frequency range. This opens the possibility of building a super-regenerative receiver, which of course is counter- productive. Second, the diode from the gate of Q1 to ground working as a clipping diode also generates more noise, as outlined above. Finally, the feedback selected between the two tuning diodes reduces the operating *Q* of the resonator to unreasonably small values. If this particular circuit is favored, then diode D2 should be made out of several (3-5) diodes in parallel. It is therefore not surprising that the measured phase noise of this circuit,

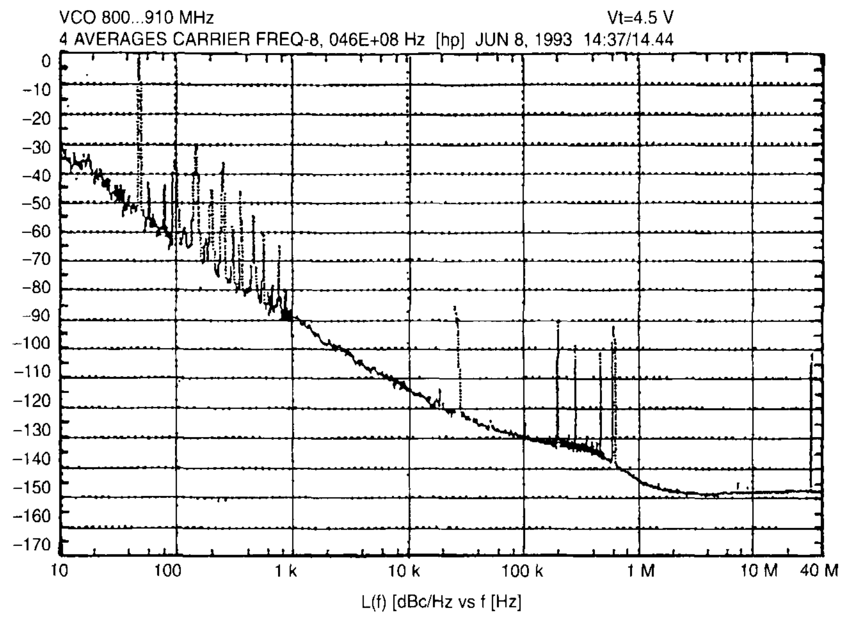


Fig 21—Measured phase noise of the oscillator shown in Fig 20.

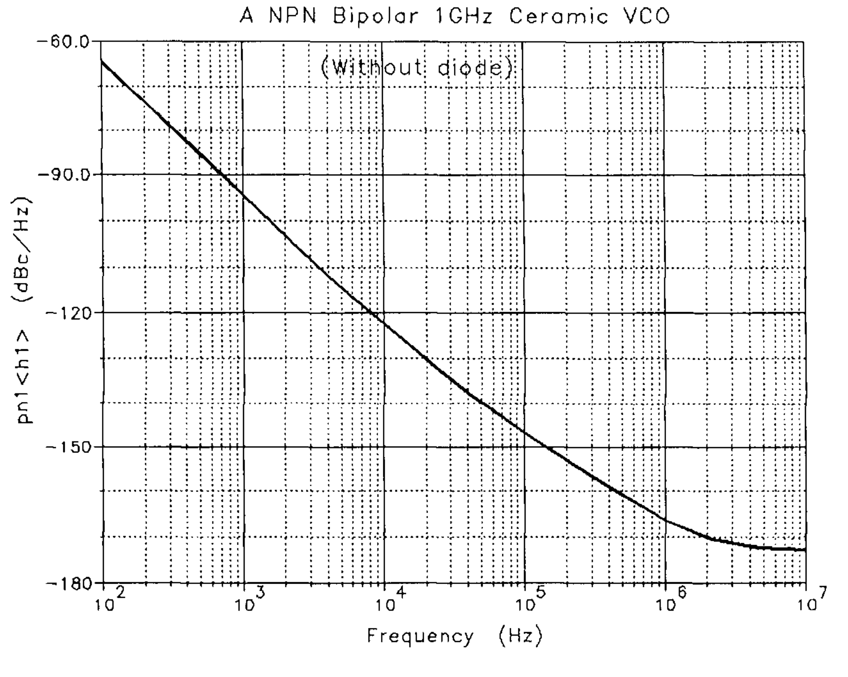


Fig 22—Predicted phase noise of the 1-GHz ceramic resonator VCO without the tuning diode.

shown in Fig 32, is significantly below state-of-the-art.

Example 4—Recommended Circuits for Higher Frequency Application

The following VCOs are ideal for low phase-noise oscillators. Fig 33 shows a schematic which is a spin-off of Fig 28 and uses a 3-dB power divider at the output. Also, the loop filter for the synthesizer application is shown.

Fig 34 shows a high-power, low-phase-noise VCO system recommended for the frequency range from 400 to 700 MHz. Please note that the tuning element again uses several diodes in parallel.

Fig 35 shows a recommended VCO circuit covering the frequency range from 700 MHz to 1 GHz. The rule of thumb is that FETs do not have enough gain for high-Q operation in oscillators above 400 MHz and bipolar transistors are a better choice. Only at frequencies above 4 or 5 GHz should we consider GaAs FETs, because of their higher flicker noise contribution.

Example 5—Millimeter-Wave Applications

In millimeter-wave applications such as smart weapons which use small radar units for the tracking of enemy targets, MMICs with VCOs are used. One of the most severe tests of software is the combination of millimeter-wave accuracy and nonlinear phase-noise calculation. As a last example I am showing the layout of such a VCO that operates at 39 GHz. While a detailed circuit description of this proprietary design is beyond the scope of this presentation, it should be noted that it is now possible to analyze such complex structures.

Fig 36 shows the actual layout of the 39-GHz VCO, and Fig 37 shows its schematic presentation. Both the transmission lines used as a resonator and the varactor have fairly low Qs. The resulting phase noise therefore is significantly worse than we have seen in other examples. Even if we took low-frequency oscillators and multiplied them up to 39 GHZ, we would get better performance. VCOs like this are being used in PLL systems which "clean up" some of the noise.

Fig 38 shows the clean up from a PLL and at the same time shows the phase noise of the same oscillator in free running mode. The operating conditions used were a 10-MHz reference and 100-kHz loop bandwidth. The clean up is dramatic if one considers the multiplication factor up to 39 GHz. The crystal oscillator's reference is the best low-noise crystal, type 10811A made by Hewlett-Packard.

Conclusion

In the past, the determination of phase noise required a great deal of guess work. The analytical approach presented herein allows—for the first time—not only an accurate prediction of the phase noise, but also allows the use of CAD tools to optimize the circuit. From a designer's point of view, the most critical assessments have been the issues of determining the proper loaded Q and the noise figure under large-signal conditions. This has always been subject to wild guesses. The nonlinear mathematical approach allows—again for the first time—a combination of all these things and provides the correct answer.

I am particularly grateful to my colleagues at Compact Software, Inc and to Professor Vittorio Rizzoli, University of Bologna, who made this work possible.

[Note: Most figures in this article were not redrawn. Therefore, they contain European circuit symbols.—*Ed*]

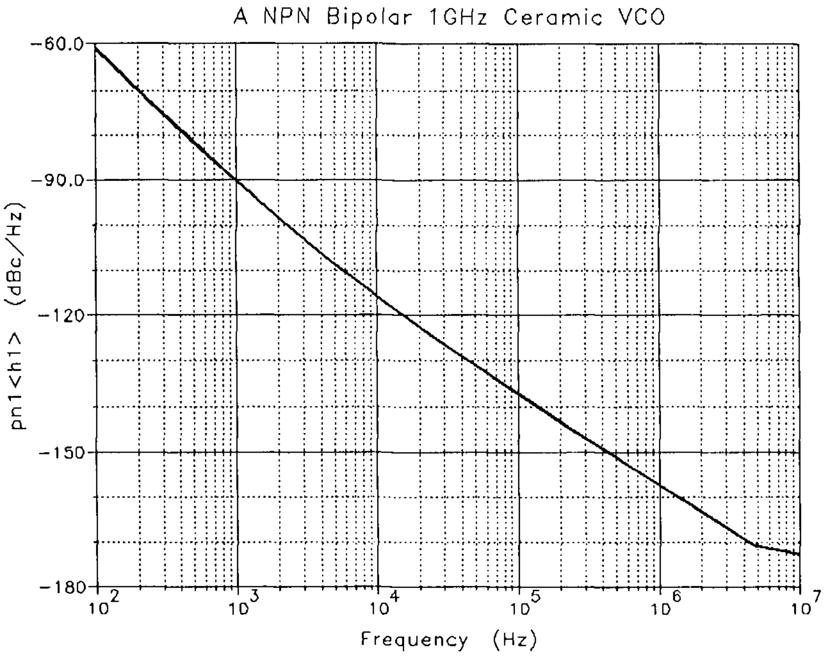


Fig 23—Predicted phase noise of the 1-GHz ceramic resonator VCO without the tuning diode attached. Note the agreement between the measured and predicted phase noise.

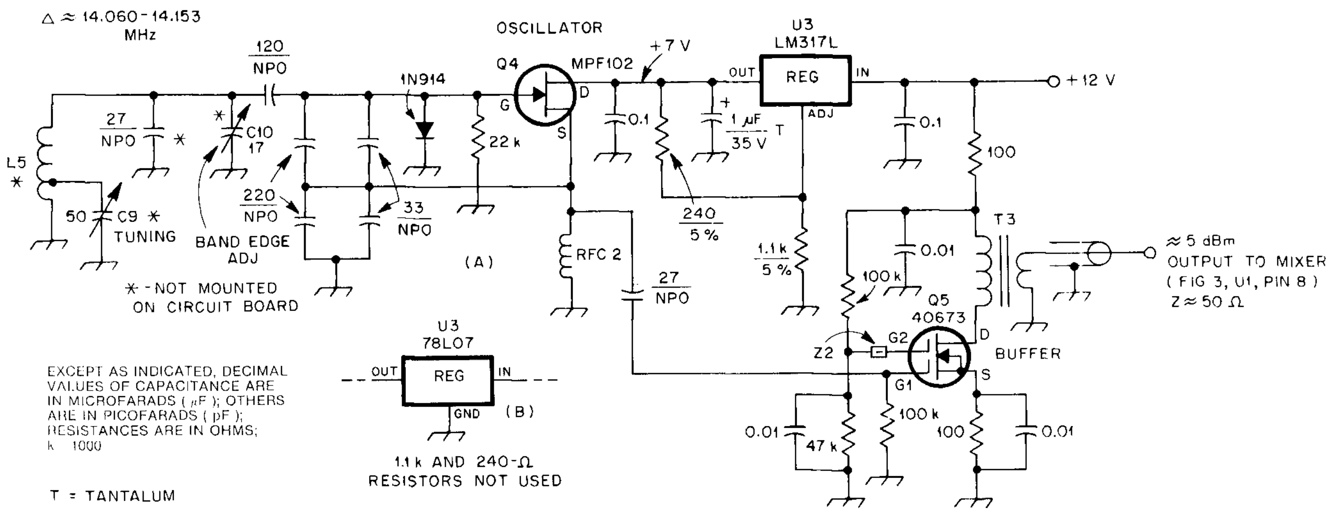


Fig 24—20-meter VFO circuit from the 1993 *ARRL Handbook*.

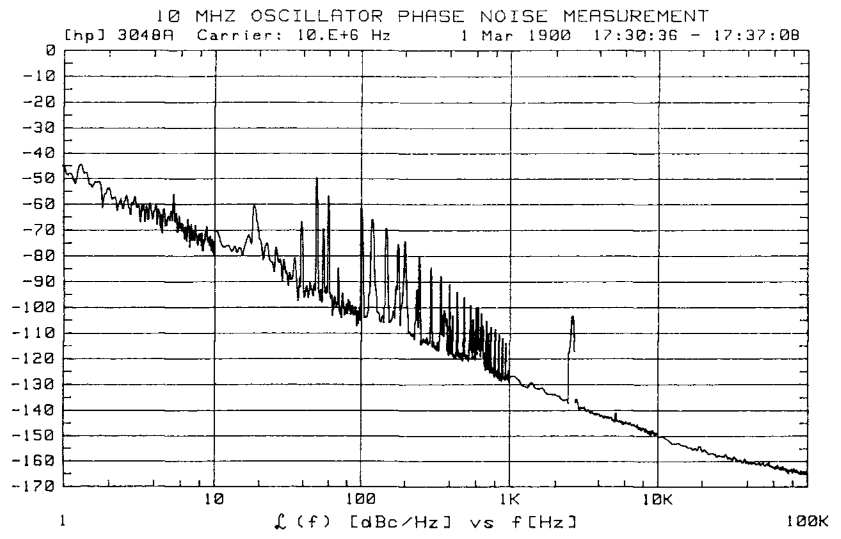


Fig 25—Measured phase noise of this type of oscillator.

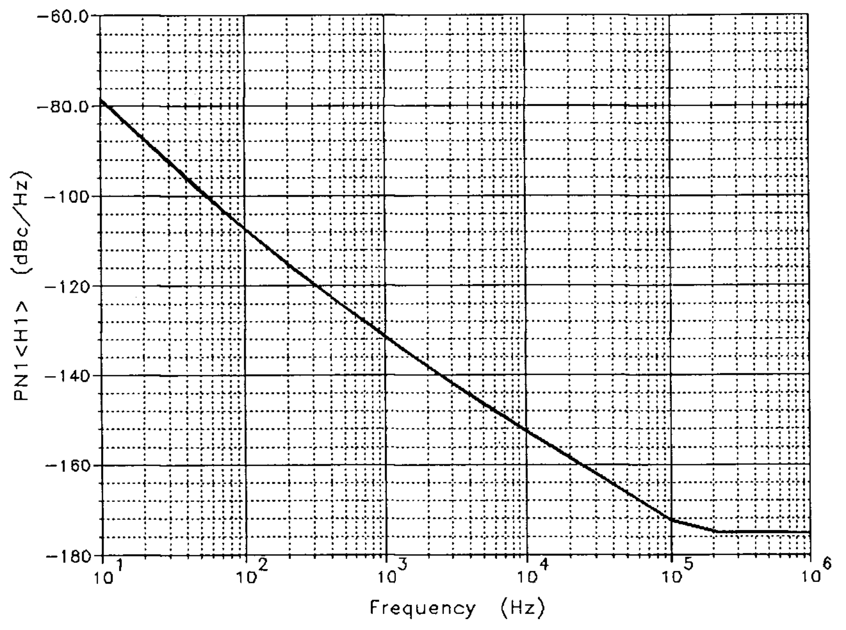


Fig 26—Shows the simulated phase noise of this type of oscillator with a clipping diode attached.

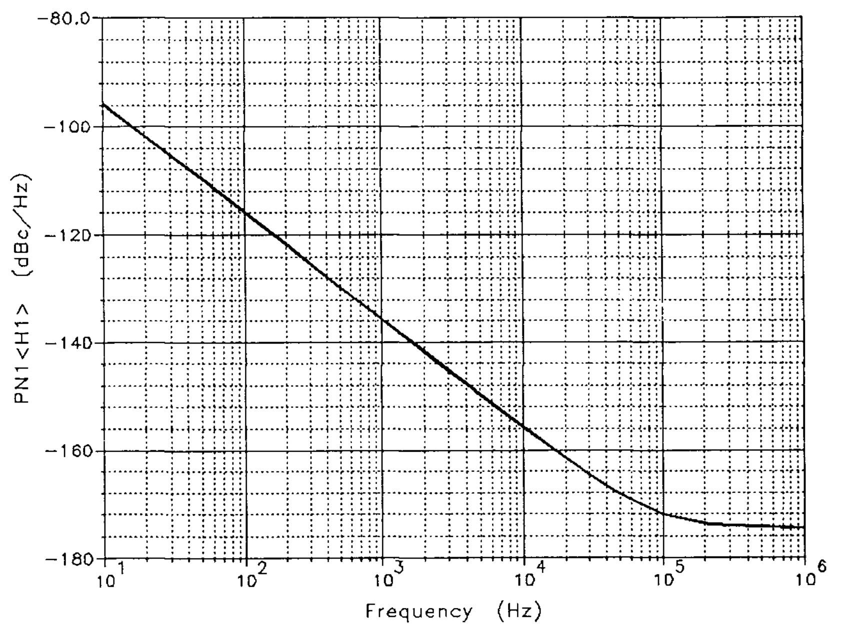


Fig 27—Simulated phase noise of this type of oscillator without a clipping diode attached.

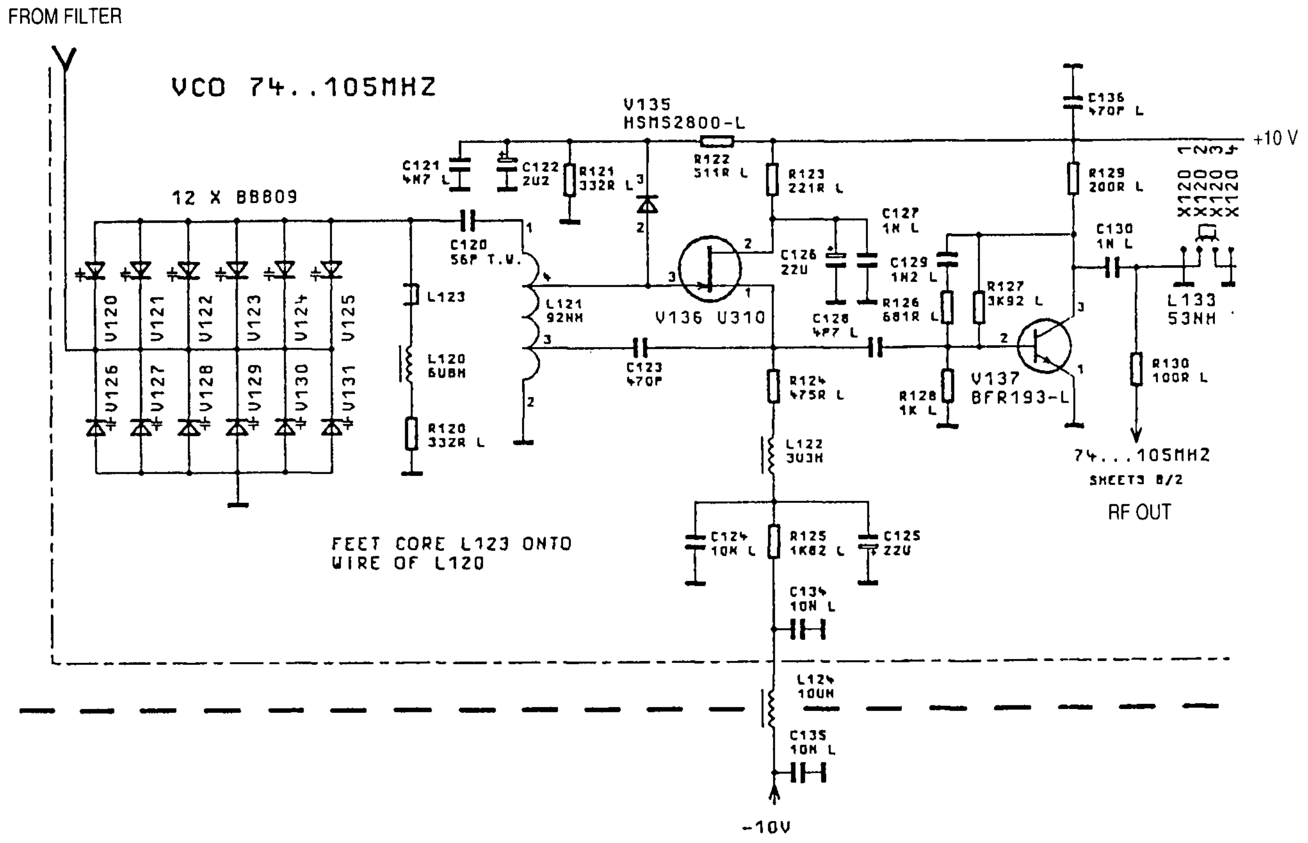


Fig 28—Wideband VCO with a large number of tuning diodes to improve phase noise. Note that the diode is biased in reverse and does not follow the positive clipping as published by other authors.

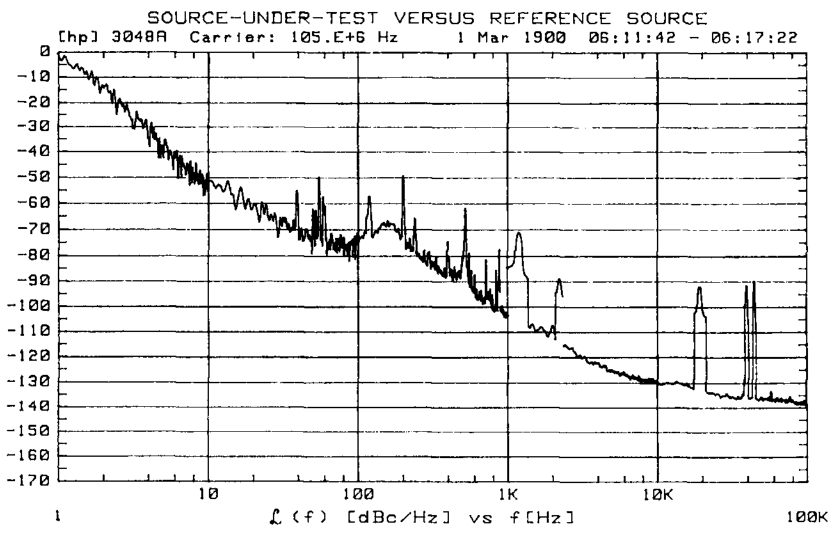


Fig 29—Phase noise of the multi-diode VCO in a PLL system.

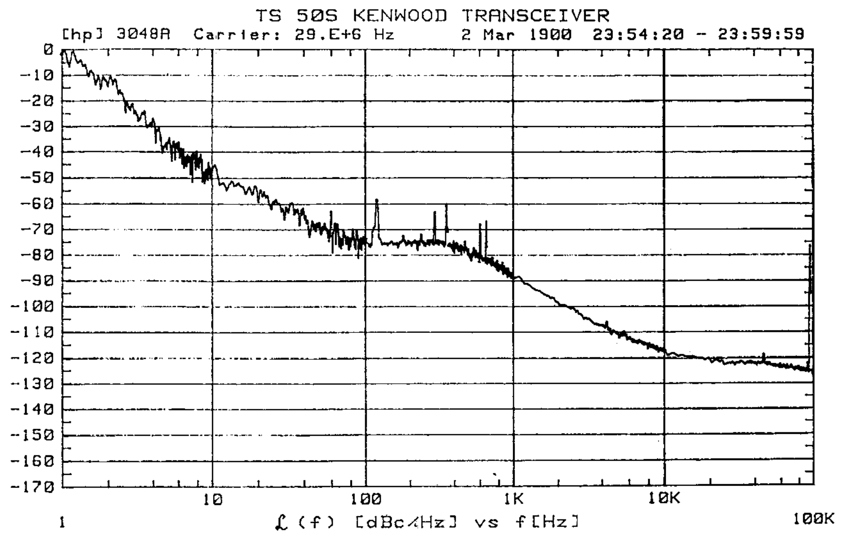


Fig 30—Phase noise of the TS-50S, which is 10 dB worse than the multi-diode system.

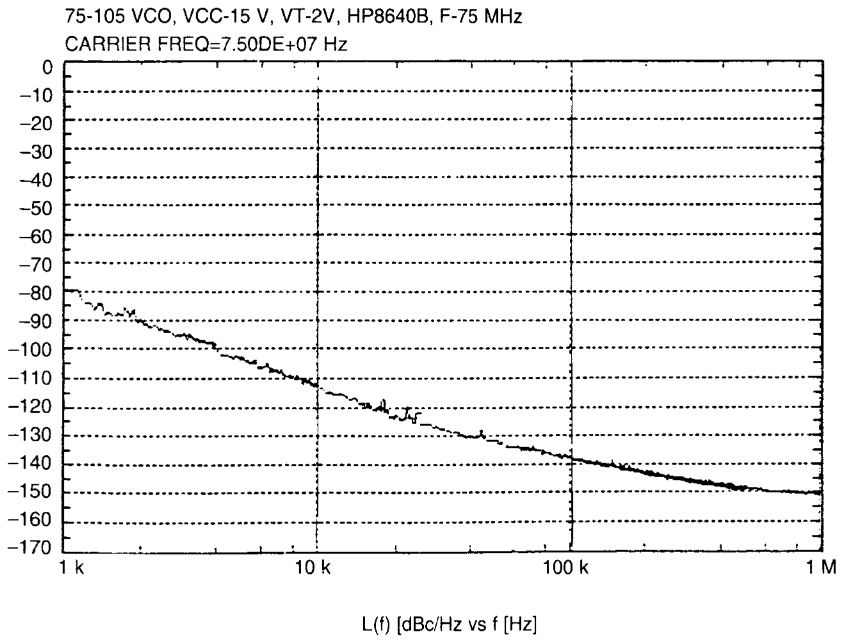


Fig 32—Phase noise of the VCO of Fig 31 which is significantly below state-of-the-art.

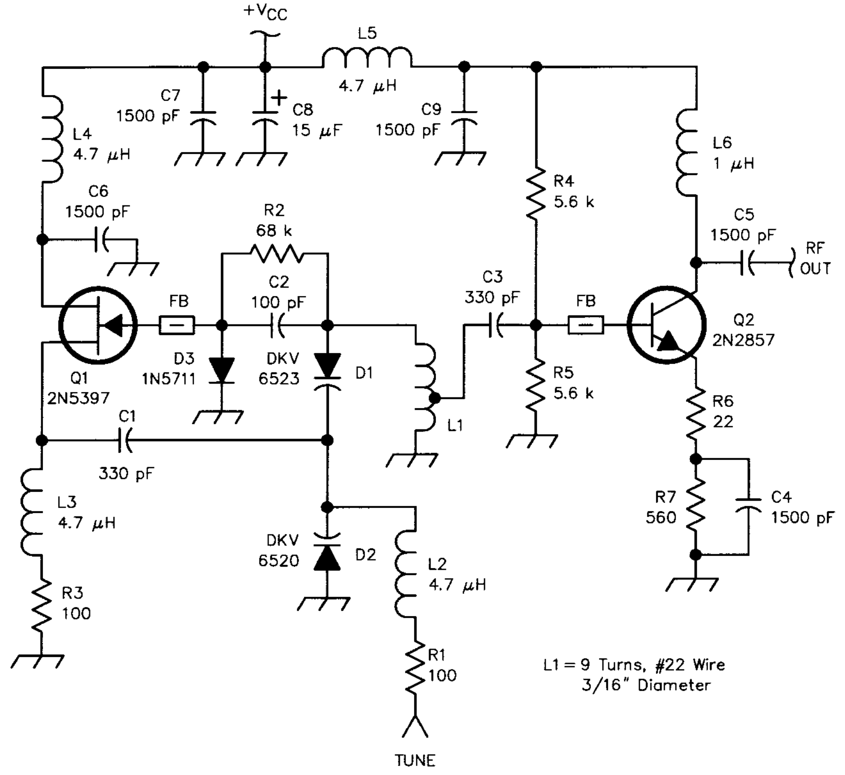


Fig 31—A VCO which violates several rules in designing a good VCO.

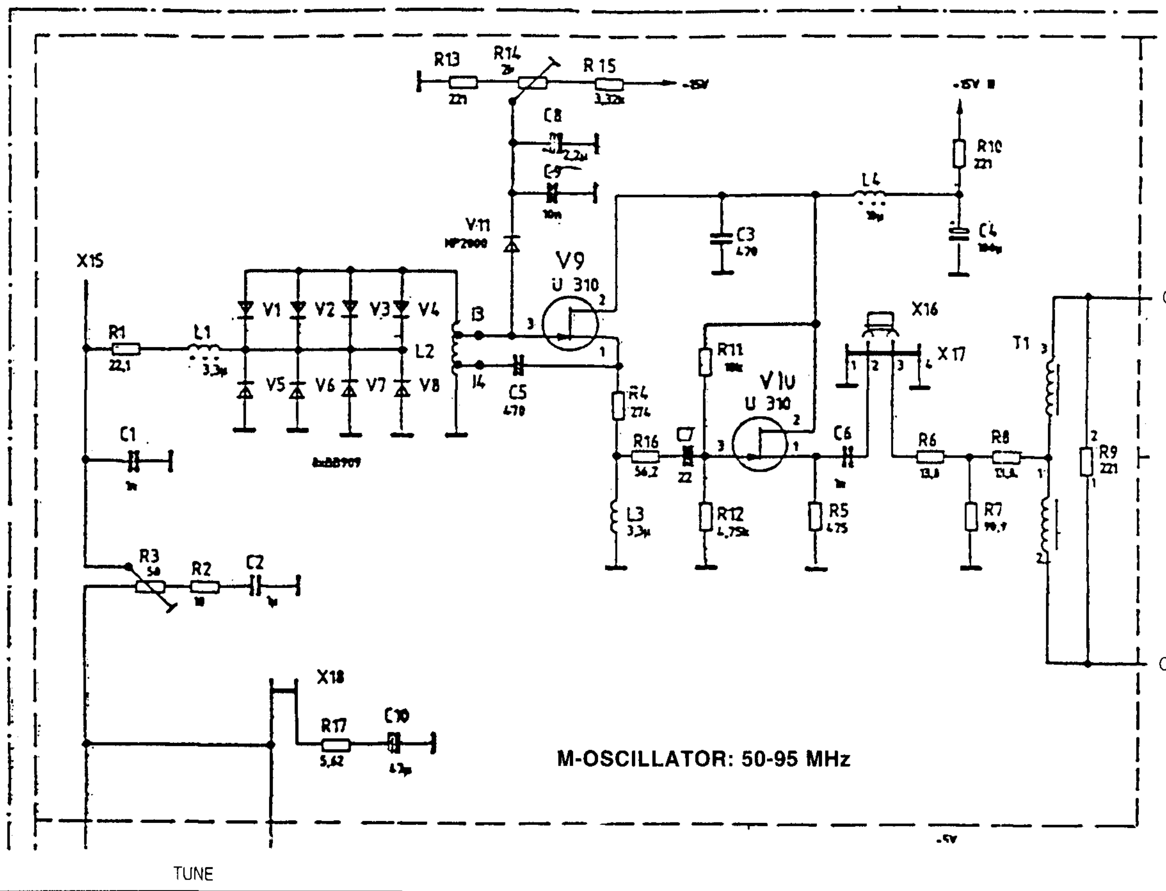


Fig 33—A schematic which is a spin-off of Fig 28 and uses a 3-dB power divider at the ouput.

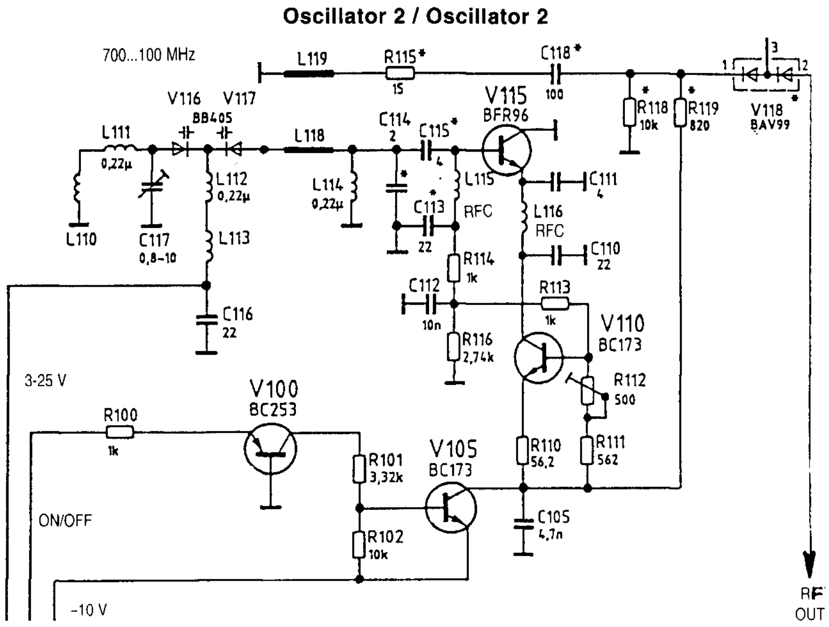


Fig 35—A recommended VCO covering the 700-MHz to 1-GHz frequency range.

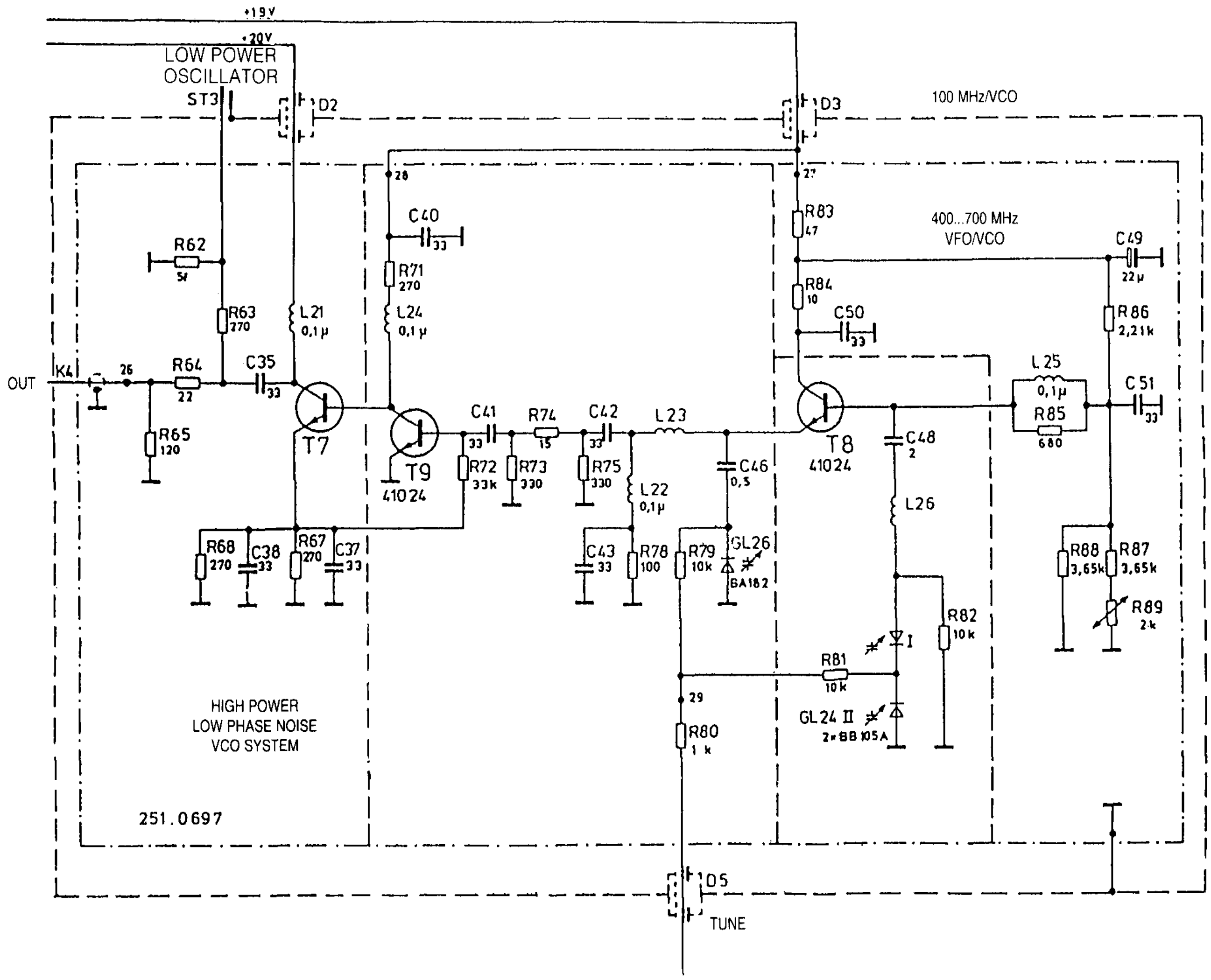


Fig 34—Shows a high-power low-phase-noise VCO system recommended for the 400-700 MHz frequency range.

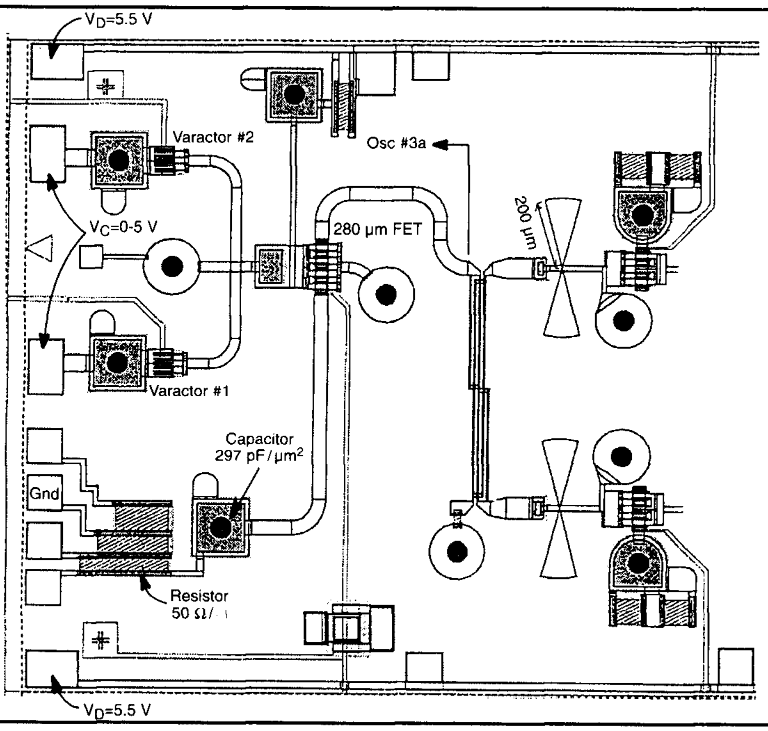


Fig 36—The layout of a 39-GHz VCO. (courtesy Texas Instruments)

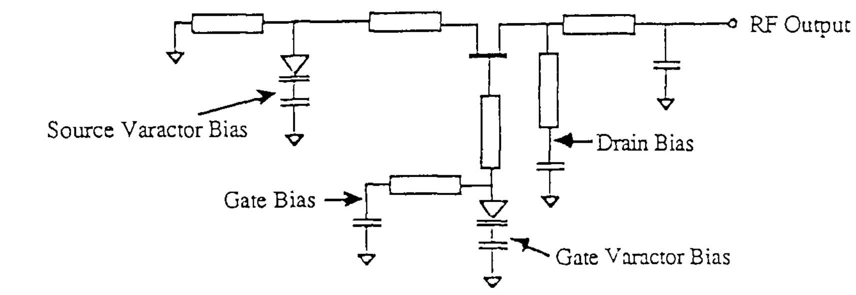


Fig 37—The schematic presentation and topology of a millimeter-wave VCO.

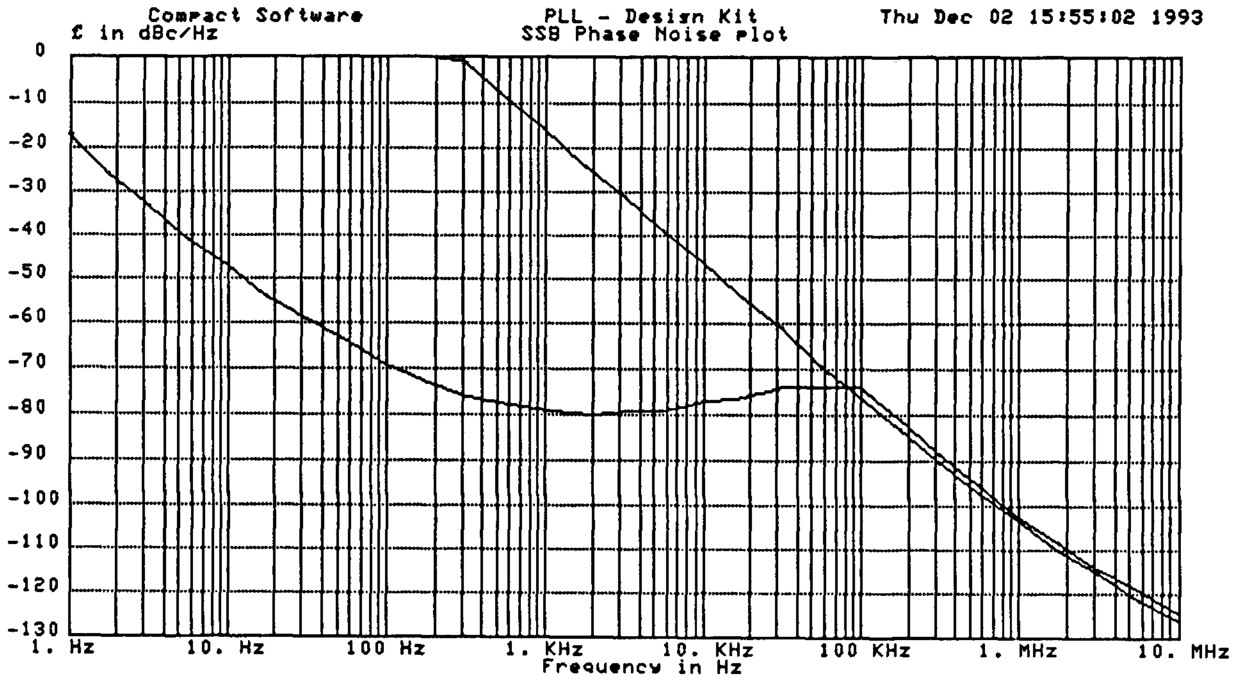


Fig 38—The "clean up" from a PLL and the phase noise of the same oscillator in a free-running mode. ◻◻

Are You An ARRL Volunteer Examiner?

If you're **18 or older**, and a qualified **Advanced** or **Extra** class licensee, you're invited to join us. For details call:

1-800-9-ARRL-VEC

ARRL VEC

QEX Subscription Order Card

American Radio Relay League
Newington, CT USA 06111-1494

QEX, The ARRL Experimenter's Exchange is available at the rates shown at left. Maximum term is 12 issues, and because of the uncertainty of postal rates, prices are subject to change without notice.

For 12 issues of QEX in the US
☐ ARRL Member $12.00
☐ Non-Member $24.00

In Canada, Mexico and US by First Class Mail
☐ ARRL Member $25.00
☐ Non-Member $37.00

Elsewhere by Airmail
☐ ARRL Member $48.00
☐ NonMember $60.00

Remittance must be in US funds and checks must be drawn on a bank in the US. Prices subject to change without notice.

☐ Renewal ☐ New Subscription

ARRL Membership Control # ____________

Name ____________ Call ____________

Address ____________

City ____________ State or Province ________ Postal Code ________

Payment Enclosed

☐ Charge
☐ MasterCard ☐ VISA ☐ American Express ☐ Discover

Account # ____________ Good thru ______

Signature ____________

Date ____________

A DSP Version of Coherent-CW (CCW)

Obtain the advantages of coherent detection of Morse signals using DSP.

by Bill de Carle, VE2IQ

Coherent CW (CCW) is an old technique for digging weak CW signals out of the noise. The system synchronizes the receiver to the sender's keying. Thereafter, due to the rules of Morse code (each "dah" equals exactly three "dit-times," and spaces between marking elements are always an integral multiple of one dit time) the receiver knows precisely when the carrier is allowed to turn on or off. This is useful information that would otherwise have to be transmitted—at the expense of additional power and/or bandwidth. Of course, the sender must have an absolutely rock-steady rhythm for the scheme to work. For this reason, CCW stations employ special keyers or keyboards which can generate perfectly timed code. Traditionally, CCW uses a keying rate of 12 WPM (100 milliseconds per dit) and a received audio tone of 800 Hz.

The receiver divides its time into 100-ms windows, or frames. Once synchronized, there is complete certainty that the incoming tone is either present or absent for the entire 100-ms window—the rules of Morse code prevent it from switching somewhere in the middle. This knowledge makes it possible to use a matched filter (sometimes called an *integrate-and-dump* filter) to reduce the receive bandwidth down to about 9 Hz (main lobe), eliminating most of the noise while letting the CW tone pass through unscathed (see Fig 1). All coherent 800-Hz energy received during the window is integrated (accumulated), and only at the very end of each window does the receiver make its decision—and the question is: "during the previous 100 milliseconds, was the key at the transmitter up or down?"

The question is answered by comparing the measured amplitude of the 800-Hz tone to some threshold value—more received energy means the key probably was down, less energy means it probably was up. If the receiver decides the key was down, it sounds a local sidetone oscillator for the next 100 ms. Apart from the 100-ms delay between the incoming audio and the regenerated tone, the sidetone signal follows the received code faithfully, and of course there is absolutely no electrical noise present from that point on since the signal has been completely rebuilt by the receiver. It is like a repeater on a fiber-optic cable. At each stage, hardware makes a firm decision based on a marginal situation then announces confidently whether the window was marking or spacing. Noise (at least in its common form) is completely removed from the incoming signal, so a human operator can copy it without stress. That is, as long as the hardware makes the right decision at the end of each window! If the signal-to-noise ratio is so awful that the matched filter gets a wrong answer, a different kind of noise enters the picture: the system still generates perfect code at its output, but it's no longer the same message the transmitter originated. Then the human operator can step in and say, "Hey, that didn't make sense!," and try to figure out what the real message must have been based on context, prior knowledge, etc.

It has been argued that a seasoned, skilled CW operator can perform the same filtering operation as CCW performs with his brain, concentrating on the incoming code while blissfully ignoring all the other stuff in the receiver's passband at the time. Maybe so, but it's pretty hard work. An equally narrow (9-Hz) conventional audio filter wouldn't help either: the circuit's response time would be so slow it would spread out the leading (attack) and falling (decay) edges of the keyed waveform in time, blurring the distinction between marks and

29 Sommet Vert
St. Adolphe d'Howard, QC
J0T 2B0 Canada
24-hr BBS: 514 226 7796

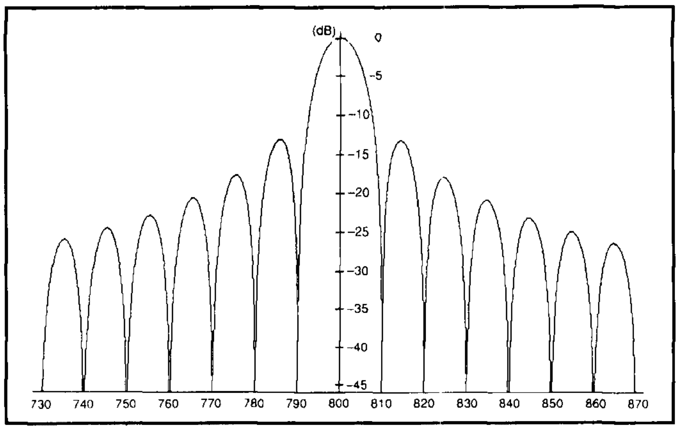


Fig 1—Amplitude response of an integrate-and-dump filter for a small range of audio frequencies near 800 Hz.

This is the characteristic response for both the traditional matched filter built with analog parts (operational amplifiers, resistors, capacitors) and for the DSP version described in this article, built with computer code. Notice the notches every 10 Hz (810, 820, 830 Hz...), and the peaks in the sidelobes—also every 10 Hz, but centered on 815, 825, etc. These sidelobes sometimes allow enough energy from QRM to get through the filter to hurt us; they could be removed (at least drastically attenuated) with further digital processing if we were willing to pay the price: additional computing time. We chose to go with the classic filter response to satisfy the CCW purists and to keep the number crunching requirements reasonable so the program can run on slower computers.

Those infinitely deep notches every 10 Hz can be put to practical use. A Lowfer (160..190 KHz) beacon operator might choose a carrier frequency of say 187.530 kHz, placing it halfway between two power-line harmonics. When the integrate-and-dump filter is centered on this carrier, *all* harmonics of the 60-Hz power line will be notched out simultaneously. Fine business!

spaces. After all the votes are in, it would seem that coherent CW carries an advantage of some 20 dB over regular CW at 12 WPM. So why doesn't everybody do it that way? Answer: up until now it has been quite difficult to get a CCW station on the air. Once synchronization had been established, in order to keep the transmitter and receiver from drifting apart, expensive frequency standards were needed at each end of the link. Transceivers had to be stabilized, usually by phase-locking their master oscillators to some common external standard such as WWVB. And that integrate-and-dump filter circuit was no piece of cake to build and align. These technical challenges have kept all but the most dedicated devotees away from the mode until now.

Can DSP Help?

Some years ago I designed and built a DSP engine dedicated to receiving CCW. It worked, but it was far too complicated—some fifty ICs on the board. I concluded that no one else would ever build one of those things, and I was right. At the time, personal computers were just coming on the scene and I didn't think there was enough processing power in them to do the job. But then there appeared ATs, 386s, 486s, and I thought it might be worth another look. Maybe, with very careful coding, we could do the DSP function on existing ham-shack computers with a minimum amount of external hardware. If so, the cost would be next to nothing and many more people could get in on CCW. After doing an audio spectrum analyzer project (see "A Receiver Spectral Display using DSP," in Jan 1992 *QST*) I realized that a CCW program using the same analog interface was feasible. This interface is a Sigma-Delta analog-to-digital coverter circuit that measures about 4 inches by 1.8 inches and runs off a 9-V battery. It uses a handful of common CMOS chips and can be built in an evening. It also can be purchased assembled and tested, ready to hook up. This circuit samples the received audio 7,200 times each second, converts the measured voltages to numbers, and passes these to the computer through one of its serial ports (eg, COM1) running at 115 kbaud. No modifications to the radio or the computer are required. It is exactly the same board described in the *QST* article and also in the '93 *Handbook*. Some of you will already have one.

What Does the Software Look Like?

The primary thing to keep in mind is that we are going to be hard pressed for time. At 7,200 samples per second, a new sample point's numerical value is fed into the computer every 139 microseconds. The software has to service the UART (serial port) interrupt, read the measured voltage, and perform the DSP filtering operation 7200 times per second. With real tight coding it can be done, and in the time left over we can do some other pretty neat things as well. The challenge is to reduce the time spent handling interrupts to the absolute minimum. The computer has to be able to process the incoming numerical samples at least as fast as they are acquired (ie, in less than 139 µs per sample, on average).

For this algorithm to work, we cannot afford to turn off the interrupts while we do some heavy computing; the interrupts have to be always enabled to guarantee that we don't miss a single sample. DOS systems usually have several "background" tasks that become active periodically, sometimes shutting off the system interrupts for up to a millisecond at a time. So the program's first action is to take over the DOS timer interrupt to make sure no other program gets control of the machine—even for a millisecond—while the DSP algorithm is running. In writing the program, I concentrated on shaving cycles from the serial-port interrupt service routine. In real-time programming, what's crucially important is to reduce the amount of time the computer spends on things which are done often (eg, 7,200 times every second). Things which happen less frequently can be coded a little more sloppily. How to minimize the interrupt service time? Well, we would like very much to avoid any particularly long machine instructions, such as multiply or divide.

The classic integrate-and-dump (I&D) filter works by first shifting the frequency of the incoming 800-Hz tone

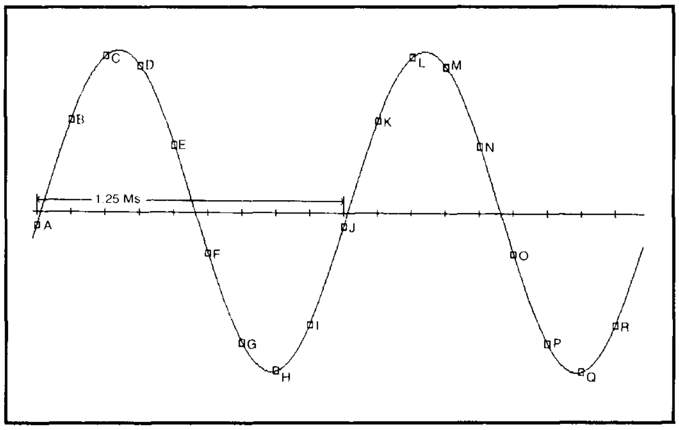


Fig 2—Each cycle of an 800-Hz sinewave takes 1.25 milliseconds. Digitizing the audio at the rate of 7,200 samples per second, we measure the instantaneous voltage nine (9) times during this same period. The analog waveform advances through forty (40) degrees of phase between samples.

To measure the amplitude of this signal (which may be buried in the noise), we use the principle of least squares to "fit" an 800-Hz sinewave on to the sampled data points. A least-squares fit gives us the best estimate of the signal's amplitude and phase. The formula requires us to multiply each voltage sample by the sine of angle X, and then to sum the resulting product into an accumulator. We must also multiply by the cosine of X, summing the result into a different accumulator. The angle X is advanced by 40 degrees after each sample. This normally takes two multiplications and two additions per sample, plus the overhead required to service the interrupt, read the voltage, advance the phase angle, etc.

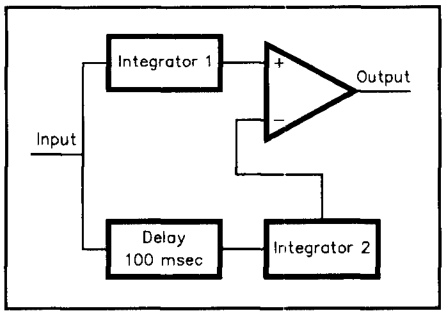
Notice that after the first nine samples (A through I), the tenth sample (J) will occur at exactly the same relative position on the second cycle as our (A) sample did on the first cycle. In other words, the angle X for sample J should be the same as the angle we used for sample A. Similarly, sample K will have the same phase angle as sample B and so on. This is a stroke of luck for us, and happens only because we have chosen a sampling rate which is an exact integral multiple of the frequency of the sinusoid whose amplitude we want to measure.

down to "baseband" (dc). This is done by mixing the audio with an 800-Hz reference tone. Once at baseband, the energy in the signal is split into two separate channels called *in-phase* (I) and *quadrature* (Q). These channels are then independently integrated over the 100-ms window. This is essentially an averaging-over-time operation and is only feasible because at 0 Hz (dc) the I and Q channels don't change their values with time. However, if the incoming tone is not *exactly* at 800 Hz, a beat note will be generated and will cause problems for us, at least to some degree. Let's take an example. At 800 Hz, a 100-ms frame consists of exactly 80 cycles of that sinusoidal waveform. We mix it with our 800-Hz reference tone and we get exactly 0 Hz, or dc, which can be averaged. Now let's say the incoming frequency was not 800 Hz, but rather 790 Hz. The beat note will be 10 Hz. But if you look at it over 100 ms, or a tenth of a second, there will be only one complete cycle. If we take the average value of its voltage over that 100-ms period we come up with zero! It's positive for half the time, equally negative for the other half, and the average value is zero volts. It matters not one iota what the starting amplitude of that 790-Hz tone was; the output of our I&D filter will be zero. And likewise for any frequency that is spaced away from 800 Hz by an any exact multiple of 10 Hz. Between 800 Hz and 790 Hz, the filter's response takes on intermediate values, ranging all the way from 0 dB at 800-Hz down to minus infinity dB (total attenuation) at 790 Hz.

At the end of each 100-ms integration period, the classic hardware filter freezes the instantaneous voltages on its integrating capacitors and, based on the values of the I and Q channels averaged over the interval, computes the average amplitude of the 800-Hz tone during that period. (It could also compute the phase of the tone relative to that of the 800-Hz local reference oscillator, but that generally is not done.) The hardware version of an I&D filter then has to dump all the charge from those capacitors instantly to start measuring the signal in the next window. This is physically impossible with real hardware integrators, so in practice we always waste a little of each frame right at the beginning while the capacitors are discharged of the voltages built up during the previous frame. (Well, not absolutely impossible: one could set up two complete filters and switch between them on alternate frames, I guess.)

So, how can we get the same answer with a DSP algorithm, get around the old problems, and hopefully avoid having to build the hardware filter at all? Our basic challenge is to estimate the amplitude of an 800-Hz sinusoid which is assumed to be unvarying throughout the entire 100-ms window, and which is likely to be much weaker than any number of interfering carriers present in the receiver's passband at the same time. The classical DSP solution to this would be to emulate the hardware I&D filter in software. We multiply the incoming samples with a unit vector rotating at 800 Hz to get instantaneous values of the I and Q components of the baseband signal, then we integrate (add up) all these values over 100 ms, eventually dividing by the total number of samples taken to obtain the averaged I and Q values. Then we compute the square root of the sum of the squares of the two mutually-orthogonal components, and that's the answer. (We could also calculate the phase of the received sinewave by computing the arctangent of the ratio of the Q and I components.) In this particular case, at 7,200 samples per second, each 100-ms window would consist of 720 samples. Which would mean 720 × 2 multiplications, 720 × 2 additions, plus a whole bunch of other time-consuming stuff at the end of each window (such as calculating the square root, clearing the accumulators, etc). All of this is mathematically equivalent to "fitting" an 800-Hz sinusoid to the sampled data points by least-squares, then solving for its amplitude and phase. That's a lot of number crunching to get done in a tenth of a second, even for today's faster home computers. And there isn't a whole tenth of a second available either—a good portion of the time

Fig 3—This shows how the "running average" version of an integrate-and-dump filter is configured. We integrate the input signal continuously without ever clearing (resetting) the integrator circuit. But we also integrate a delayed version of the same signal (it's delayed by 100 milliseconds), then we subtract the outputs of the two integrators. The result is the integral of the signal over the last 100 milliseconds, and it can be sampled at any time, not just at the end of some particular window. The *Coherent* program uses this scheme to sample the DSP filter's output three times near the end of each window: a little early, at the nominal time, and a little late. These numbers are then compared to see if we need to adjust the timing. In CCW it is important to keep the tracking window on top of the sender's window with the smallest possible amount of stagger. For example, if the window is lagging during a mark-space sequence, not only is there less mark energy available to detect, there is also more (erroneous) space energy due to the part of the mark pulse which is received in the following window. Since the recovered intelligence depends on our ability to distinguish between mark and space, this is a double-whammy that can rapidly degrade copy if we allow the transmitter and receiver to start drifting out of sync.



goes for overhead (pushes, pops, etc) in servicing those 720 interrupts—the DSP algorithm has to operate in whatever time is left over after all that. So it's tough, eh? Well, here is where we start getting lucky. On most computers a multiply instruction takes much longer to execute than a simple addition, so we would be far ahead if we could eliminate those two multiplies per sample—and it just so happens we can. We know that at 7,200 samples per second, each individual cycle of an 800-Hz tone takes exactly 9 samples to cover. In other words, during the time between each sample and the following one, an 800-Hz sinusoid will have advanced through 40 degrees of phase. And 40 × 9 = 360 degrees, which is one complete revolution exactly. Eureka!

After 9 samples have been processed, the coefficients we have to multiply the samples with will start to repeat, taking on the same sequence of values for the next 9 samples, and so on. So of our 720 samples in total, 80 of them will be multiplied by two particular numbers, another 80 will be multiplied by a new two-number set, etc. This good fortune is entirely attributable to the fact that our sampling rate just happens to be an exact integral multiple of the frequency of the sinusoid whose amplitude we want to measure. We can take advantage of it by using something called the distributive law (in algebra): A×B + A×C + A×D equals A×(B+C+D). You get exactly the same answer in the end, but one way needs three multiplies and two adds, the other way needs only one multiply and two adds. We will use this to solve for the average I and Q component values with only 18 multiplies per 720 samples instead of 1440!

But there is another trick available. It turns out that the cosine of 40 degrees (one of our sample phases) has the same value as the cosine of 320 degrees (another one of our points). Likewise, of the remaining seven coefficients, six of them can be paired up in this way, the seventh is unity, and it's real easy to multiply by one! On the sine side, 8 of the 9 coefficients can be paired (allowing for sign changes), the other coefficient is zero (it's even easier to multiply by that, hi!). The bottom line is that by combining terms we can get by with just 8 multiplications instead of the 18 that would otherwise be needed. This is starting to look doable!

What we have to obtain during each 100-ms window are the end values of nine (9) accumulators, and on each of the 720 interrupts we only have to add the current sample into one of those nine accumulators. For each interrupt we sum into another accumulator, and after the ninth sample has been processed we start over with the first accumulator. Thus, aside from the house-keeping associated with servicing the interrupt, checking for overruns and/or clipping, figuring out which accumulator to address and such mundane things, we are left with a simple addition to perform. For sure, at the end of the 100-ms window we must do some further calculations, but we have all the time in the world (relatively speaking) to get them done in the 100 ms during which the data for the next-following window are being acquired.

Now, there is one little complication: I would like to run not one, but three (3) I&D filters concurrently. These three filters should overlap in time so that by comparing the three outputs I can decide if the window phase is drifting and it would be advantageous to make a slight adjustment to the phasing cycle that determines when each window starts and ends. The most economical way to run three such I&D filters concurrently is never to clear the various accumulators. Instead, I keep a "delay line" in memory, consisting of the last 720 samples taken. For each new sample, I add it into the appropriate accumulator, then subtract off the value of the sample taken 720 time slots earlier. In this way the accumulators are guaranteed not to overflow, because in the long run we subtract out as many counts as we add in. Furthermore, at the end of every 9 samples (one cycle of the 800-Hz audio signal)—the nine accumulators always hold exactly the same numbers you'd get if you started 100 ms ago with cleared accumulators and integrated through one complete window's worth of samples. Try doing *that* with analog integrators! It works on the computer because digital fixed point arithmetic is absolute—there are no errors such as would arise in an analog integrator due to charge leaking off a capacitor, component values changing slightly with temperature, etc. Any analog integrator would eventually saturate at one rail or the other due to such errors. The digital integrator can run for hours (or days) with absolutely no accumulated long-term error.

We still must "sample and hold" the values in those nine accumulators whenever we have to compute the amplitude of the measured component over that particular window. This involves moving nine 16-bit numbers to secondary storage positions. It is done with a single block move instruction and has no impact whatsoever on the processing of subsequent samples, so the computer version can start to process each new window immediately (not throwing away any of the incoming energy) as opposed to the analog version, which has to wait for the capacitors to discharge fully before starting a new integration cycle.

At the end of each window, we must then compute the amplitude of the measured 800-Hz component. This involves those eight multiplications mentioned above, then taking the square root of the sum of the squares. This is done in tightly coded assembly language in order to execute in the shortest possible time.

As a fine-tuning aid for the operator, the program also measures the frequency of that 800-Hz component (to the nearest tenth of a Hz) and displays it on the computer screen, updated at the end of each 100-ms window. There is a trick to this: the frequency has to be measured *after* the DSP filtering operation. Otherwise, any nearby strong carrier could disrupt the measurement and give an erroneous reading. In fact, during each *marking* window (once we have decided that the 800-Hz tone was indeed present during that window) we also measure its phase (averaged over the entire 100-ms window) and save this for later reference. On the next *marking* window, as long as it is not too far in time away from the last measured one, we measure the phase again. If there is any slight discrepancy between the frequency of the incoming 800-Hz tone (from the receiver) and our precise 800-Hz reference tone (which is actually determined by the 1.8432-MHz crystal oscillator on the Sigma-Delta board), there will be some phase shift in the detected signal (in the same way the amplitude of the "beat note" varies regularly whenever there is a slight difference between two compared frequencies). If we know the amount of phase shift as well as the time interval over which this phase shift accumulated, we can figure out how much the received frequency differs from the nominal 800-Hz value, and there you have it. The operator can set a software switch to enable this phase comparison to also occur across an intervening *space* frame (or not). If the transmitter is phase-coherent from one key-down to the next, then it is feasible to use those two keydown periods to measure his frequency. If not, then the software only uses phases measured in consecutive marking frames (eg, during a "dah"), when the key is presumably held down for 300 ms continuously and the transmit carrier could not change its phase during that period.

That is the essence of a DSP version of the integrate-and-dump filter. The algorithm runs on just about any IBM-compatible computer. The program incorporating this algorithm is called *Coherent*. There is enough time left over after the DSP calculations to allow for lots of other CCW goodies. For instance, *Coherent* has an autotune feature. It is important to keep the incoming audio tone centered in the filter's rather narrow passband. *Coherent* tracks the incoming signal's frequency. If it deviates more than half a hertz from the nominal 800-Hz value, *Coherent* issues a pulse on one of two RS232 control lines to make the receiver tune up or down by 1 Hz. Many modern rigs can tune in precise 1-Hz steps by pressing the MIC up/down buttons, so *Coherent* makes the signals needed to do this automatically. Once the signal has been tuned in initially, the operator can sit back, put his feet up, and leave the driving to the computer. Even if his receiver drifts in frequency (or if the transmitter drifts) it's no problem because the software will retune the radio as necessary to maintain the CW tone at 800 Hz.

Coherent also has a frame-phasing tracking loop. After each 100-ms frame is processed, the program looks at whether the SNR would have been better had the window ended 1 cycle (1.25 ms) earlier or 1 cycle later. If there is consistent evidence that going to a slightly earlier window would improve the SNR, then the program does this automatically. What this means is that once synchronization has been achieved, the operator can let the program track the incoming signal and adjust the phasing as necessary to maximize the SNR advantage the mode is capable of. With this system, special frequency standards and rig stabilization are no longer needed. The only equipment you need to operate CCW is a *reasonably* stable transceiver, the little Sigma-Delta interface board, and a computer.

And, of course, the *Coherent* program also lets you send CCW just by typing on the keyboard. That should go without saying. As well, the program has a "beacon" mode, where a prestored CCW message can be scheduled to go out at precise time intervals based on the computer's clock.

And Now, Something to Think About...

We have seen that by synchronizing our receiver to the keying at the remote transmitter it is possible to shrink the passband of our receiving filter down to a rather astonishing nine hertz or so, in the process eliminating much of the noise that would otherwise render the signal unreadable. That is fine for coherent CW stations, but what about ordinary CW—where the guy at the other end is sending by hand and his carrier can turn on or off at any arbitrary time? After all, the overwhelming majority of amateur stations around the world don't use coherent CW. Is there anything we can do with DSP to help dig these weak signals out of the mud?

Consider a train of RF pulses, where a carrier is switched on for, say 100 milliseconds, then switched off for the next 100 milliseconds. Let's assume this signal is received by a normal amateur sideband rig with its local oscillator tuned 800 Hz away from the incoming carrier. Looking at the audio coming out of the speaker, what frequency components are present? Well, that depends on your point of view! On the one hand, if we take the position that for a frequency component to exist it must be present always with unvarying amplitude and phase, then we must presume many frequencies, all adding up to make the on/off pulsed waveform. On the other hand, if we examine the waveform on an oscilloscope, we see 80 cycles of a pure (800 Hz) sinewave inside each pulse with no other frequencies present during either the pulses or the silent periods. Common sense tells us there is but one frequency (800 Hz), and that it is only there some of the time. Since there is only one frequency in the signal, it would make a lot of sense to use an arbitrarily narrow filter (ie, 0-Hz wide) centered on that 800-Hz tone. Such a filter would eliminate *all* the noise (QRM, QRN) except that which happened to be on *exactly* the same frequency. The way we usually design highly selective (narrow) filters is to take many samples spaced over a long interval of time and combine them mathematically to isolate the contributions of all the individual frequencies. Analog filters use the same technique, storing energy in reactive components. The narrower the filter (the higher the "Q") the longer the energy from any given cycle stays around inside the tank circuit. The drawback with all these filters is that it takes a long time for them to attain their final output value after a step change in the input signal (as is the case when a CW carrier is keyed). The sharper a filter is in frequency, the longer it takes (in time) for it to respond. This is why experienced CW operators will tell you it doesn't pay to use IF filters much narrower than about 250 Hz when trying to copy code by ear. Narrower filters actually make it *harder* to copy because they obliterate (smear) the sharp leading edges of the keyed tones which the ear needs to recognize code patterns. But the characteristic time spreading of such filters is not a result of some insurmountable law of physics! It follows entirely from the particular way they were designed: they observe a signal over a long timespan to make fine distinctions in frequency in order to realize the narrow response.

The ideal filter for copying ordinary CW would be 0-Hz wide and have an instantaneous response time. When the key went down at the transmitter, the output of the "sliver" filter at the receiver would reflect the amplitude change immediately.

What approach can we take in designing such a filter? A good question is: for any given signal, how much of it do we need—how long do we have to observe it before we can break it down into its component frequencies and state what the amplitude at any specific frequency must be? Could we take just a momentary snippet out of a waveform, analyze it extensively on a fast computer and figure out its complete spectral content just from that tiny portion we looked at? The answer will surprise you. The answer is yes! In fact, there is no minimum amount of time for which we need to observe a waveform in order to completely characterize it. In theory, we could sample a complex signal for just one instant and immediately know the amplitude of an 800-Hz sinusoid in it—regardless of whatever other frequencies might be present. The calculation gets a whole lot more complicated when many other frequencies are present, but it still can be done. The most straightforward case is when we know there is only one sinusoid, say at 800 Hz, and we want to ascertain its amplitude with just one instantaneous glance at the waveform. Hmmm—if we knew the phase it would be easy. With only a single voltage measurement taken at a known point along a sine curve (phase), we can determine its amplitude. Not knowing the phase in advance, we have to solve for it. Which means we need at least two (2) independent measurements, both taken at the same instant in time. The actual sampled voltage will do for one of them. For the other we can use the first derivative—the rate the voltage is changing at that particular moment. This derivative can be obtained without taking any time: convert the voltage to a current, run it through an inductor, and measure the instantaneous voltage across the inductor. Here we have an implementation of our "ideal" CW filter for the simplest case where there is only one frequency component to resolve. When there are many frequency components to separate, we will obviously need more information, but it is all available in that same instant; the higher order derivatives are mutually orthogonal, hence independent, and they're there for the measuring. So it is possible (at least in principle) to design a CW receiving filter with arbitrarily narrow bandwidth (approaching 0 Hz) and a virtually instantaneous response time. What's needed is hardware to differentiate a signal repeatedly and a very fast computing machine to crunch the numbers.

Obtaining the Software

The following can be ordered from the author:

Coherent CCW software package, $20

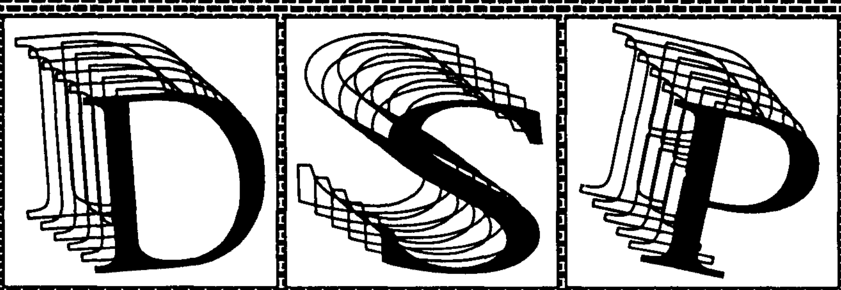
Bare circuit board for constructing Sigma-Delta interface, $24

Assembled and tested Sigma-Delta board, ready to hook up, $95

All prices in US dollars, and please include $5 for airmail shipment to anywhere on the planet.

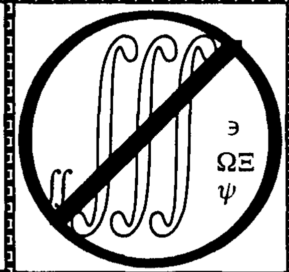
For more information on CCW, contact:

CCW Interest Group
Peter Lumb, G3IRM
2 Briarwood Ave
Bury St Edmunds
Suffolk IP33 3QF
England



Without Tears™

Learn DSP and Put your Knowledge to work Immediately!



Coming to a Place Near You

Ft. Lauderdale
Albuquerque
Washington D.C.
Chicago, Dallas
San Jose, Long Beach

Call Z Domain Technologies, Inc.
(800)-967-5034
(404)-664-6738

Call 9-5 EST. Ask for brochure.

Our 2-Day Advanced Course is ready. Call for info

By Taking This 3-Day Course You will really learn DSP. Guaranteed!

Digital Communications

Harold E. Price, NK6K

Best Laid Plans...

I had planned to make the source for the FEC system discussed in the previous column available in this column. Events have conspired to keep that from happening—the first Christmas where my daughter was old enough to get into it (recommended), a kidney stone (not recommended), and work made for an interesting December. Not to worry, though, the mailbag brought some interesting comments on the subject of FEC.

FEC

In the December column, I said "Tests made by Jon Bloom at the ARRL lab show that coding adds at least 4 dB to the performance of the system in a white noise environment." This precipitated several email messages, parts of which are discussed below.

From Kevin J. Rowett, N6RCE (krowett@cisco.com):

Harold, I just received the December QEX *and read your column covering FEC experiments. On a theoretical level, I believe it has a flaw. You state that testing shows "...coding adds at least 4 dB to the performance of the system...". This implies the information carrying capacity of the system channel has improved by 4 dB by using the FEC scheme previously described.*

While I've not reviewed the described FEC scheme in detail, I don't believe it has the ability to increase the information carrying capacity of the channel. Yes, the FEC scheme does allow the radio receiver to operate with a lower S/N ratio, but, FEC adds more bits to the channel, thereby decreasing the digital bit throughput by 4 dB.

FEC doesn't provide a free lunch. You can't get more information capacity out of an existing channel by using FEC. Does FEC have a place in [Amateur Radio]? Yes, in cases where the receiver sensitivity can't be improved, or the transmit power can't be raised (as on a satellite transponder), or better antennas won't help. Maybe it even makes sense in fixed land-based packet radio. The alternative is to use better antenna systems. Anything but omni-to-omni would help. However, where to use FEC doesn't change my comments regarding the stated system improvement of 4dB. The information capacity of the channel didn't improve!

When speaking of the performance, I was speaking of the "system" performance, meaning the collection of hardware and software under test. We did the test by determining the point at which the channel was useless (or severely degraded) for standard AX.25 with the point where the FEC-enhanced system just started to miss frames. The difference in S/N between these two points was 4 dB. This means that, with the FEC scheme described in the June 1993 article, you can use a link that is 4 dB "worse" and still transfer data.

The FEC system does not increase the ultimate information carrying capacity (look up "Shannon bound" in any communications text), but it does reduce the distance between the performance of the system described and the Shannon bound.

Glenn Elmore, N6GN, (glenne@hpsadl3.sr.hp.com) followed with:

I wrote to Phil [Karn] about a related situation with the "40-dB improvement" on radar. I think I agree with Kevin. The channel capacity, as defined by Shannon, isn't changed by what type of modulation/demod or FEC you use. By declaring a channel "useless," you've thrown out the baby with the bath water because of artificial constraints you've placed on the system.

One bad bit has caused you to dump an entire frame of framelength – 1 bits of good data. This does indeed look like a step function.

Due to threshold effect on the detector, on some systems you could argue that 1 dB of increased antenna gain increased the channel capacity by 40 dB (or infinitely) because it moved the BER across a fictitious boundary. Look what happens when you improve many TVRO systems by a couple of dB.

Having said the above I hasten to add that I think that FEC is essential to higher-speed digital radio-packet systems. I'm not sure of the precise number, but I suspect that trying to design L1 hardware to guarantee much more than 10^{-2} or 10^{-3} BER is a bad idea. FEC fills the gap between that and reliable frames to the software. This BER target is after you've done all the right things with antennas, channel equalization through DSP or SS, radio design etc. And I would dearly love to have a low-cost hardware FEC solution to drop in to my 1245-1280 MHz SS radios. Please let me know if someone finds one.

We're talking about the real-world current ham-radio implementations, which do have the "artificial limits" discussed by Glenn. We are using HDLC frames for both AX.25 and TCP/IP links. This causes an entire block of data to be rejected if one bit is bad. This is a legacy passed down by the early VADCG TNC, and is an unfortunate side effect of its ham radio origins—we're cheap. Hams use FM radios (cheap), simple HDLC chips (cheap), and the vast majority still use Bell 202 modem tones (also cheap). All of this causes throughput to rapidly degrade as the S/N passes various thresholds. Amateur packet started on local VHF links where the average S/N was far above the threshold so the disadvantages of the one-bad-bit syndrome was masked by other concerns, such as the hidden terminal problem.

Kevin added: *My concern with the article was the representation of the improvement as a "system" improvement. I fear the casual reader might think FEC cures channel fading, and why don't we do FEC on everything. Without the practical disclaimers noted, we also*

5949 Pudding Stone Lane
Bethel Park, PA 15102
email: nk6k@amsat.org (Internet)
71635,1174 (CompuServe)

open ourselves up to criticism on the theoretical level. The channel baud performance didn't really improve, our use of it did with the FEC scheme.

I completely agree HDLC, with it's all-or-none bit scheme, is a bad choice for channels that suffer from short fades. Some form of FEC is better than raw HDLC!

Glenn had similar comments: *Yes, I think we agree. The only problem is that the readership may not understand this context.*

I think that's what needs to be made clear. Otherwise I'm afraid that FEC might be jumped on as the next panacea, out of perspective with the other issues. "N dB improvement" is relative to the current mangled way of doing things. N depends on which particular mangle you're comparing it with. That's an argument for keeping Shannon as an L1 target/reference, and then building to highlight the other real-world constraints and needs which must be met in a better manner in order to build an effective amateur network.

Somehow we need to better present the multiple issues which exist, L1-L7, so that potential contributors can get sufficient overview to effectively contribute in their particular areas of expertise. It's a big task and figuring out how to get and keep everyone on the same page is really tough.

I agree with both comments, and have therefore devoted much of this column to highlight the issue. Thanks, Kevin and Glenn.

In the meantime, Phil Karn, KA9Q, is carrying on with his FEC work. This includes the 1970's concept of "punctured codes," where you encode the frame with FEC, and then intentionally remove certain bits from the output stream. The receiver knows the pattern of the removals and can use the rest of the FEC information to reconstruct the missing bits. This is a way of turning a rate 1/2 code (twice as many output bits as input bits) into something more efficient (for example a rate 7/8 code, eight output bits for seven input bits). Here you trade error correcting capability for better channel utilization, but require more work in the FEC decoding, since there are always "errors" to be corrected. Recall that Phil's goal is to counter radar induced errors, meaning the underlying channel S/N is good, but bits are removed by radar. Only a relativily small number of real bits are damaged by radar, allowing for a high number of software-removed bits.

As the cost of CPU power continues to decline, old algorithms are worth a new look.

Archie

I also had several comments on the topic of audio spectrum analyzers. One was from Jim Sanford on how to find the FFT142 program and others on the Internet. The easiest way is to use an archie server. Archie (from "archive") maintains a data base of file names from public archive sites. It will report on the locations of files. Many sites will have a local archie client program. You simply type "archie," followed by the name of the program you want. On the system I use, typing "archie fft142" gives me a list of 30+ locations for the fft142 program. If you don't have an archie client, you can connect to a site that provides the service. One such is archie.sura.net. Telnet to this address and log on as "archie." You will first get a list of other archie server sites that may be closer to you. You can then type "prog fft142" to get the file's locations.

If you have Internet access but are an occasional user, I recommend going to the local book store (Walden's, B. Dalton, etc.) and buying one of the many Internet beginner's books available. I've looked at several, they all carry the same basic information on email, ftp, gopher, archie, veronica, and other tools for the Internet browser.

More audio analyzers

Lon Ahlen, WZ9X,
(add@ahlen.cc.purdue.edu) writes:

*While "playing" with a DSP design kit, I found another resource useful for plotting and display—*PC-DSP *by Oktay Alkin [published by Prentice Hall]. It included software for simulation of digital filters and other useful tools. Also,* EXCEL *includes the FOURIER function that performs a Fourier transform. I also used this for some "pretty" reports.*

Any ideas on how one can find more time to play around with this stuff?

Not more time maybe, but to help justify your time, write up your excursions on the digital frontier for *QEX*! □□

AMATEUR TELEVISION

SEE THE SPACE SHUTTLE VIDEO

Many ATV repeaters and individuals are retransmitting Space Shuttle Video & Audio from their TVRO's tuned to Satcom F2-R transponder 13 or weather radar during significant storms, as well as home camcorder video. If it is being done in your area on 70 CM - check page 461 in the 93-94 ARRL Repeater Directory or call us, ATV repeaters are springing up all over - all you need is one of the TVC-4G ATV 420-450 MHz downconveters, add any TV set to ch 2, 3 or 4 and a 70 CM antenna. We also have downconverters, antennas, transmitters and amplifiers for the 400, 900 and 1200 MHz bands. In fact we are your one stop for all your ATV needs and info.

Hams, call now for our complete ATV catalogue!

We ship most items within 24 hours after you call.

(818) 447-4565 m-f 8am-5:30pm pst. **Visa, MC, COD**

P.C. ELECTRONICS **Tom (W6ORG)**

2522 Paxson Ln Arcadia CA 91007 **Maryann (WB6YSS)**