Wade G. Holcomb,
185 Lind
New Haven, Connecticut 065

TRY NMR WITH YOUR OLD CW RIG

Using amateur radio equipment to perform nuclear magnetic resonance experiments

---

Want to try something new and different with your old CW rig? Consider building your own experimental nuclear magnetic resonance (NMR) instrument. With it, you can experience the thrill of sending and receiving radio signals to the protons of hydrogen atoms. As a matter of fact, it's entirely possible to duplicate discoveries made shortly after World War II with that old CW rig of yours, plus a surplus magnet similar to those that formed part of a radar magnetron. Of course, some readjustment will be necessary to get your old rig tuned to the correct frequency. You'll also need an oscilloscope and an automatic keying circuit.

For those who enjoy construction and troubleshooting, this experiment could be the basis of a science fair project using dated ham rig components. Special interests in RF circuits or computer software are very useful in building

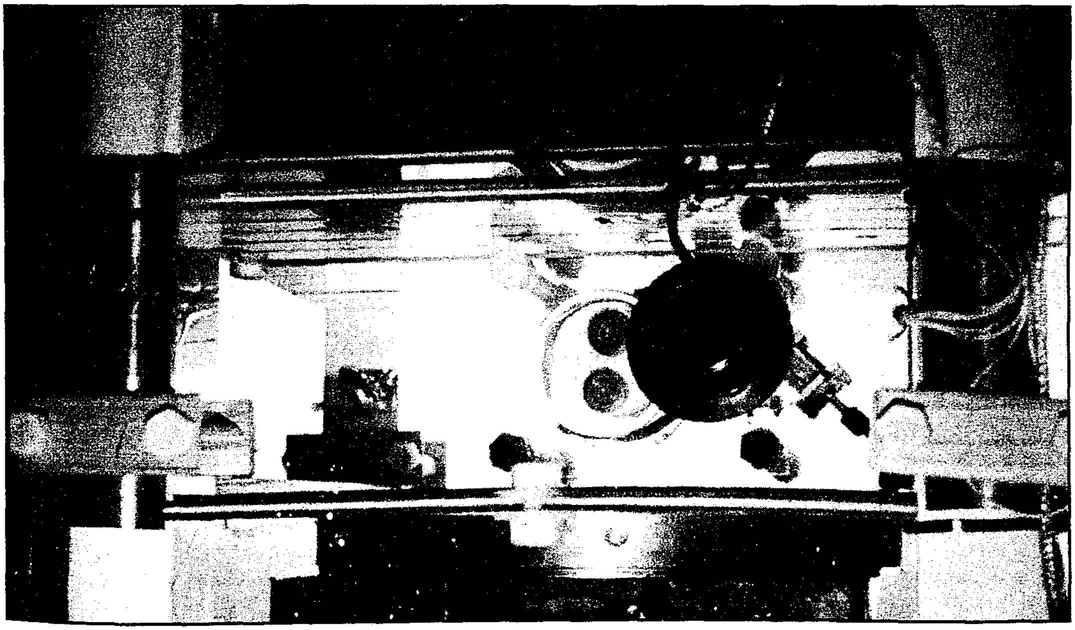


Photo A. Magnet with RF tank coil with two tubes of salad oil. Four steel support columns also serve as the return magnetic field circuit. The field is about 731 Gauss.

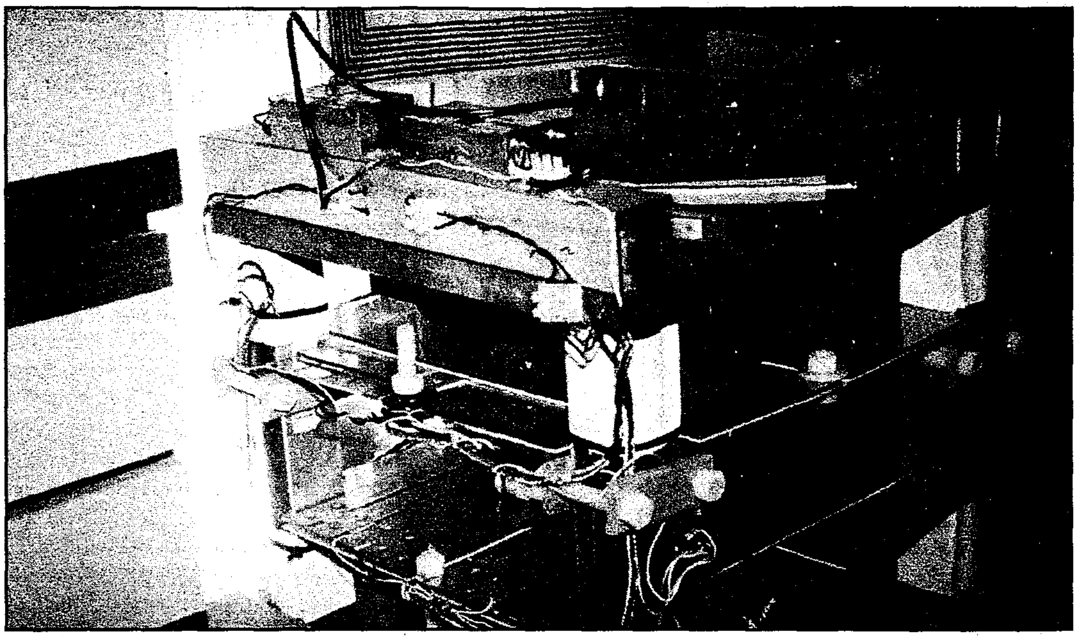


Photo B. The four-poster magnet is 18 inches on each side. A bottle of salad oil is inserted inside a 3.11 tank circuit. Credit cards can be erased if one is not careful.

your own amateur NMR system. **Figure 1** shows a functional block diagram of the major components required to perform amateur NMR.

What is nuclear magnetic resonance?

The hydrogen atom contains one proton at its center. Nuclear magnetic resonance (NMR) and magnetic resonance imaging (MRI) techniques make use of two magnetic fields—a fixed field and a variable radio frequency (RF) field—in a manner that lets an observer make physical measurements based on the proton's reaction to these fields. This method allows one to study the properties of many common substances using components familiar to radio amateurs.

While information on NMR is mostly accessible to those with training in one of the physical sciences, **Reference 1** offers detailed explanations of the fundamentals of NMR using a descriptive, mostly nonmathematical approach. The rapid development of medical MRI systems required that a trained support force be available. This book is often used by institutions to teach support personnel, and is one of several books written to fill this need.

Many atomic nuclei have "spin" and charge. Spin is the atomic equivalent of angular momentum in everyday life. According to quantum theory, a nucleus with spin can only take certain energy levels in a magnetic field. We can visualize the nucleus spinning like a bar magnetic on its axis, producing an associated magnetic field. It is the interaction of this field with external fields that separates nuclear energy levels and allows NMR to occur. The magnetic moment (current times enclosed area) is sometimes called a nuclear magneton. The hydrogen atom has a 2.79 nuclear magneton value.

A small bottle of salad oil contains a large number of possible radio signal sources (about 6 x 10E+22 per cubic milliliter). **Photo A** shows two tubes of salad oil inside a tank circuit between the poles of my magnet. In my magnetic field, only about one atom per million atoms is a potential contributor, on a chance basis, to a detectable RF signal following an RF pulse. A huge number of such atoms results in a detectable signal. The strength of the detected signal can be as much as 5 μV.

The duration of the RF keying pulse and its power level must be determined by experimentation to find the correct amount of energy to "flip" protons. Best results are obtained when the flip is 90 degrees from the static field. For instance, it's possible to have too great a pulse duration or power level, which might result in flipping the protons 450 degrees, a complete circle plus 90 degrees. The detectable signal would be similar to the correct amount!

Finding a magnet

Magnets are still available from surplus catalogs. When choosing a magnet, remember that the RF signal frequency's purity is a function of the field's uniformity. The magnet's uniformity is equal in importance to its field strength in procuring good results. Obtaining a uniform field is a never-ending goal for NMR and MRI

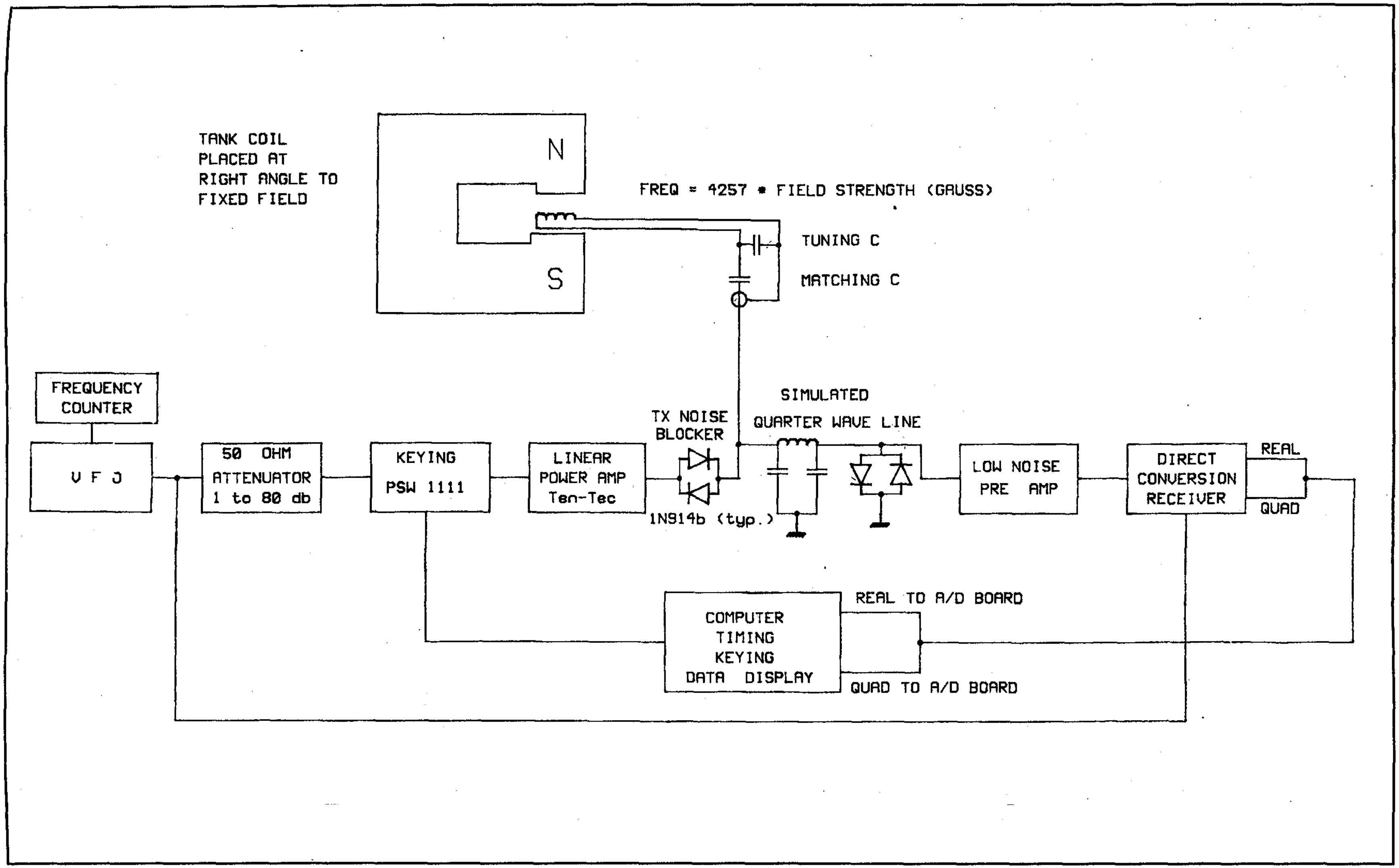


Figure 1. Functional block amateur NMR system.

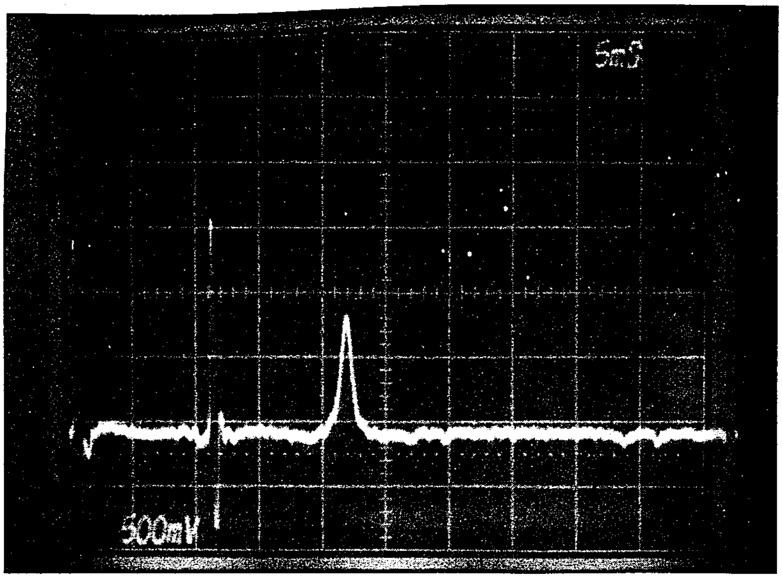


Photo C. A dot/dash RF pulse to the tank coil holding an oil sample in a magnetic field sends back an RF Hahn echo. This is one of the first subjects a new NMR student finds out about (see references).

workers. A tolerance of 5 to 10 parts per million over a volume the size of a golf ball would make a very useful amateur magnet. A change of 1 gauss will mean a change of 4257 Hz in the observed frequency. Moving a metal chair near the magnet can distort the magnetic field and detune your system.

It's even possible to make tests using the Earth's magnetic field at a frequency about 2000 Hz, using audio in place of RF equipment. Perform these tests in your backyard, away from cars or other large metal objects. Several papers appeared during the 1950s showing excellent results in measuring small variations in the Earth's magnetic field.[2]

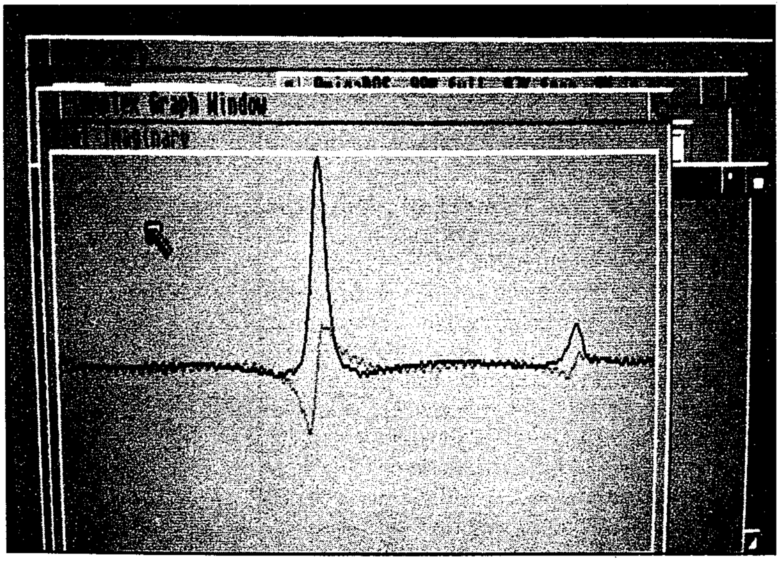


Photo D. Amiga screen shows the real and quadrature of the Hahn echo held RAM memory, this display is the average of 16 echoes. A dual A/D converter board suitable for stereo music will do this nicely.

Simple NMR experiments

The vertical field strength of my 500 pound magnet (see **Photo B**) is about 731 gauss, approximately 1400 times the Earth's magnetic field at my QTH. This magnet is quite temperature sensitive, almost 1 gauss/degree C. I usually have to readjust my master oscillator to find the hydrogen proton frequency if the room temperature changes. My magnet's field strength increases in cold weather.

Once I find the resonant proton frequency, I measure it within one cycle using a frequency counter. This frequency allows a very accurate method of determining the magnetic field strength. The relationship of frequency to magnetic field strength is given by Larmor's constant:

f—magnetic field in gauss x 4257

In my magnet, the NMR frequency is 3.11 MHz, for a field strength of 0.0731T, (The ST unit of Tesla, T, equals 10,000 gauss.) This is near the amateur 80-meter CW band.

My RF tank circuit looks like an 80-meter final tank coil (see **Photo B**). It's driven by short duration RF pulses at 3.11 MHz. When the RF field is applied, the protons spinning in the plane of the static field rotate out of the plane of the field. When the RF field is turned off, the protons return to the plane of the static field, with two degrees of rotational freedom.

The protons' spins, after the RF pulse is turned off, go through a spiral trajectory—like an orange being peeled from one end to the other—emitting a weak RF signal into the resonant tuned tank circuit. The detected RF signal takes the form of a damped sine wave. This damped wave is called a free induction decay (FID), which can last several seconds in a very uniform field, or perhaps only a few milliseconds in a non-uniform field. I sometimes judge the best spot in my magnet by positioning my sample for the longest FID.

This recovery is described by two time constants, T1 and T2, which can be measured later if the data is stored in computer memory. These two time constants, longitudinal (T1) and transverse (T2), describe these return spins to the static field, and can indicate the effect of nearby atomic neighbors on the observed hydrogen protons. For instance in pure water, the two time constants are equal to each other, but this isn't so in salad oil or other complex compounds.

System requirements

The amateur radio requirements needed to

↓ WIDTH ≈ 200 HZ

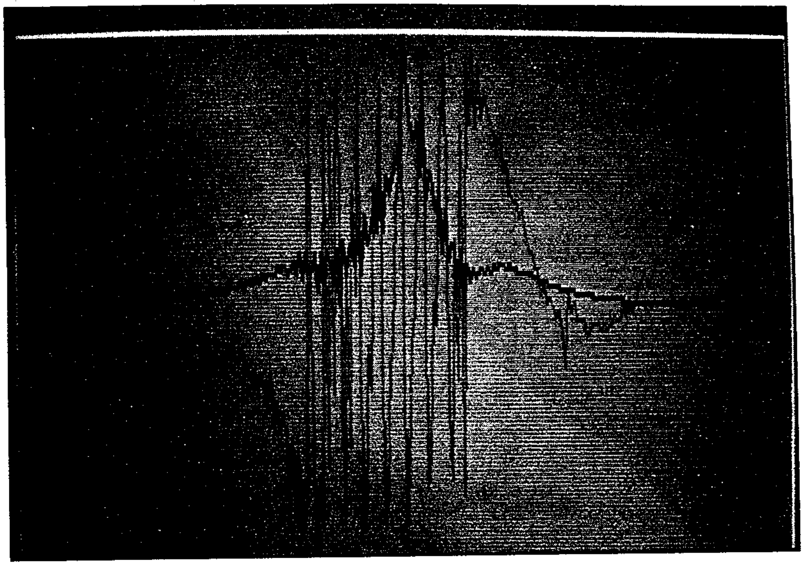


Photo E. Frequency spectrum of Hahn echo shown in *Photo D*, found by using computer software. Baseline is 10 kHz wide. Width at the 50 percent amplitude point is about 200 Hz and may be used to judge magnetic field uniformity. Phase spectrum is shown in background.

↑ USING DAVID REDDY'S PROGRAMS

bounce an RF signal off the earth-moon-earth (EME) are equivalent to those required for listening to the proton's spin (see **Figure 1**). As you know, these are a transmitter, receiver, antenna, keyer, a low-noise receiver front end, a T/R system, and a display. The keyer in my system is a computer interface board and software. I use a direct conversion receiver.

I use a computer with a timer board to generate a dot and dash pattern to key the transmitter with the two required pulses—a 90-degree dot followed by a 180-degree dash. The dot lasts 100 µS and the dash 200 µS in a typical pattern, with a 25 mS spacing. This is repeated after a 500-mS delay. Several different timing patterns are required to determine the proton spin time constants (T1 and T2). You could try it with a hand key, but you wouldn't get the accuracy you need.

The Hahn echo,[3] in **Photos C** and **D**, appearing at 25 mS from my "dot" 90-degree pulse, is captured with a computer analog-to-digital board and stored in computer memory, much as one digitizes a note of music. Later, I use computer software to determine the frequency spectrum (**Photo E**) of the stored echo by Fast Fourier Transform (FFT). The spectrum line width helps me determine the magnetic field uniformity at the position of my sample.

History

I.I. Rabi was known to have been a radio amateur, and was photographed at the controls of this "wireless telegraph" station as a teenager, around 1912.[4] He's given credit for the general concepts of using two magnetic fields to overcome the field created by the atom's rotating electron, which shields the atomic nucleus. He was awarded an unshared Nobel prize in 1944 for this work, while doing radar development for the war effort. More Nobel awards were presented to others for carrying out advances on this method in the months following the end of World War II using circuits developed by the wartime radar laboratories.[5–7] No complete study has been published covering the scientific history of the development of NMR and MRI.

Work in progress

At present, I'm measuring time constants and doing spectrum analysis of Hahn echoes to measure field purity. This should be easy for amateurs to repeat using almost any computer. I did my first Fast Fourier Transform on an Apple II+ based on an article in *BYTE* for viewing music spectrum. This required writing a 6502 machine language FFT routine. This allowed the Apple to become my first audio spectrum display about 10 years ago. I hope to obtain my first 2-dimensional MRI picture, perhaps an image of a sectional slice through an orange, soon.

I'll have to develop computer software and gradient amplifiers to drive the gradient coils shown in **Photo A** before this is possible. Complex patterns of gradients and RF pulses are needed to acquire a 2-D image plane, which must then be "decoded" using 2-dimensional spectrum analysis. With the help of Dave Reddy, N1RBJ, I've developed computer software that will perform a double-precision 128 x 128 2-D FFT on a generic 486DX 66-MHz PC clone in about 4 seconds—much faster than the expensive array processors used for these kinds of reconstructions in the recent past.

We've tested this software by reconstructing raw data of a water-bottle phantom originally acquired on a Yale University experimental NMR system. **Photo F** shows the raw data, which looks like ripples spreading in water, and **Photo G** depicts the finished magnitude and phase images. Note that the finished images are inverted, and the air bubble at the top of the bottle with its meniscus is shown at the bottom.

Summary

If you're interested in transmitters, receivers, or computer software, you'll find the effort required to capture the radio signals emitted by the proton's spin a challenge. Everything I've done can be recreated using common amateur

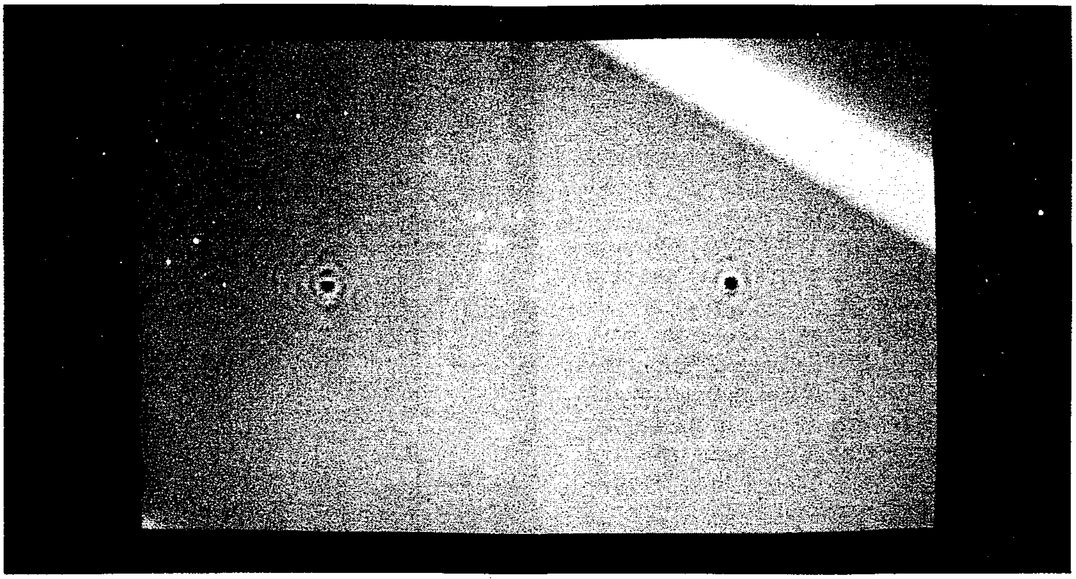


Photo F. Raster display of two 64K arrays showing RF data received from an oil sample. MRI images look like holograms before the 2-D FFT data reduction. This represents a 128 x 128 x 12 bit array.

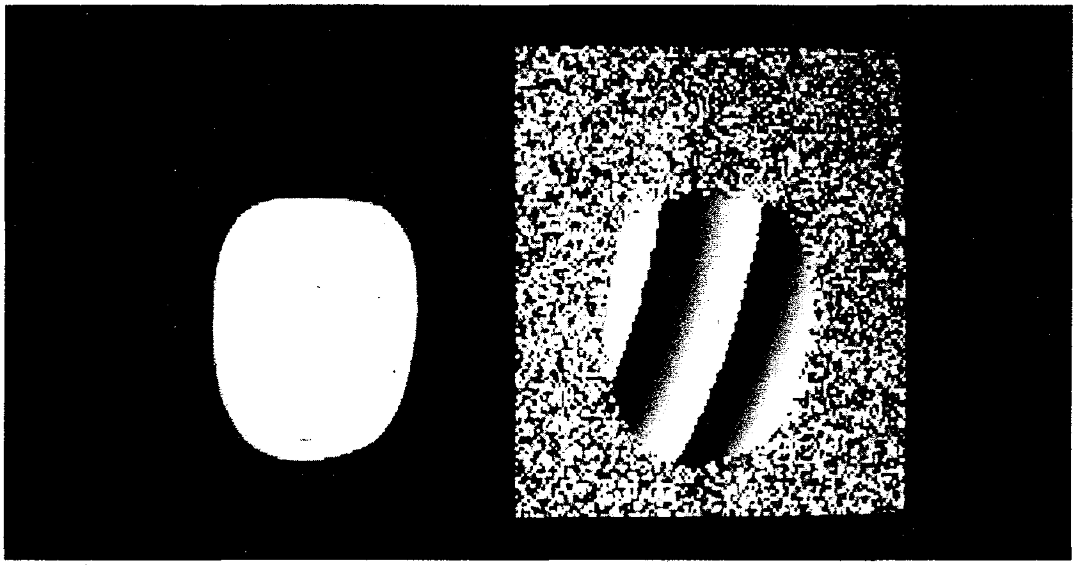


Photo G. After a 2-D FFT computer analysis (*Photo F*) shows a cross-sectional slice through the oil sample bottle. These two images now occupy the same memory space as the images in *Photo F*. Process requires 4 seconds on a 486DX 66-MHz computer.

parts, a magnet, and some patience. Amateurs with RF circuit and computer experience are well-equipped to learn about NMR. I had to learn many new terms—like Larmor's Constant, FFT, FID, T1, T2, and many others—before I was comfortable with this new field that uses RF and computer equipment to perform tasks which would have been material for science fiction stories not too many years ago. ■

REFERENCES

1. H.J. Smith and F.N. Ranallo, *A Non-Mathematical Approach to Basic MRI*, Medical Physics Pub. Corp., 1989.
2. G.S. Waters, "A Measurement of the Earth's Magnetic Field by Nuclear Induction," *Nature*, 73:691, 1955.
3. E.L. Hahn, *Physics Review 270: 80*, 1950.
4. J.S. Rigden, *Rabi Scientist and Citizen*, Basic Books, Inc., 1989.
5. C.L. Stong, "How amateurs can build a simple magnetic resonance spectrometer, *Scientific American 200:171–178*, April 1959.
6. G.E. Pake, "Magnetic Resonance," *Scientific American*, pages 58-66, August 1958.
7. I.L. Pykett, "NMR Imaging in Medicine," *Scientific American 246:78-88*, May 1982.

Fast Fourier for the 6800

Richard H Lord
Bennett Rd
Durham NH 03824

If you're involved with music or speech processing applications with your computer, you've probably wished you could look at the frequency spectrum of your sampled signals. This may not be as difficult as you might guess, because here is a simple, straightforward fast Fourier transform (FFT) subroutine that can do the trick in just a few seconds.

A Microhistory of the Fast Fourier Transform

The analysis of waveforms for harmonic content has a long and fascinating history. Bernoulli and Euler developed the mathematics of the transform while experimenting with musical strings in 1728, nearly a hundred years before Jean Baptiste Fourier gave his name to the equations. Interest in prediction of the tides led Lord Kelvin to build a mechanical harmonic synthesizer that inspired the construction of increasingly complex mechanical harmonic analyzing machines. This trend culminated in the Mader-Ott machine of 1931, which is on display at the Smithsonian Institute in Washington DC.

With the growth of the telephone and the communication industry came sampling theory and the *discrete Fourier transform.* At first, discrete Fourier transforms were hand calculated and tabular forms called "schedules" were soon employed to speed the process. With the development of digital computers in the 1940s this task became somewhat easier to perform. The number of calculations required still made the concept of real time discrete Fourier transforms unlikely even on the ever faster new computers.

Then in the 1960s a number of matrix theory mathematicians, including J W Cooley and J W Tukey, went back to the "schedules" and discovered that a great many of the terms were redundant and could be factored out. The procedure they evolved became known as the *fast Fourier transform*, which reduces the number of calculations to the point that special hardware can be built to perform the transform in real time and display the frequency spectrum continuously on a video display.

The Basic Concepts

A number of books have been published describing the mathematics of the fast Fourier transform in some detail. A few of these contain sample programs in FORTRAN, ALGOL, or BASIC. However, the use of a high level language to perform this computation not only costs a great deal in speed and efficiency, but also obscures the simple binary processes that characterize the algorithm. Since high level languages do not usually support bit manipulation, these processes can become almost as time consuming as the arithmetic.

Clearly, assembly language programming of the fast Fourier transform offers many advantages, but the literature seldom provides any examples of assembly level code to illustrate how the equations are implemented. Thus the program described in this article may well be the reinvention of someone else's "wheel"

The details of the inner workings of the fast Fourier transforms are left to the technical references, but the basic concepts are not difficult to grasp. The transform involves complex products which behave in the manner of the coordinates of a rotating vector. When this vector is at angles which are multiples of 90 degrees, the sine and cosine terms of the equations become +1, 0, or −1. Since terms containing these values do not require computed multiplication, the arithmetic becomes very simple. Other terms cancel each other out in order to simplify the equations at other angles. By factoring these terms out of the transform, many unnecessary calculations may be eliminated.

The input data may be thought of as elements of an input matrix which will be multiplied by a transform matrix. The product is a matrix containing the transformed data. The redundant elements may be factored out of the transform matrix, converting it to the product of a number of simpler transforms. For an input array of 256 points, a discrete Fourier transform would require 256 by 256 complex products or 262,144 binary multiplications. The fast Fourier transform reduces this to eight simpler trans-

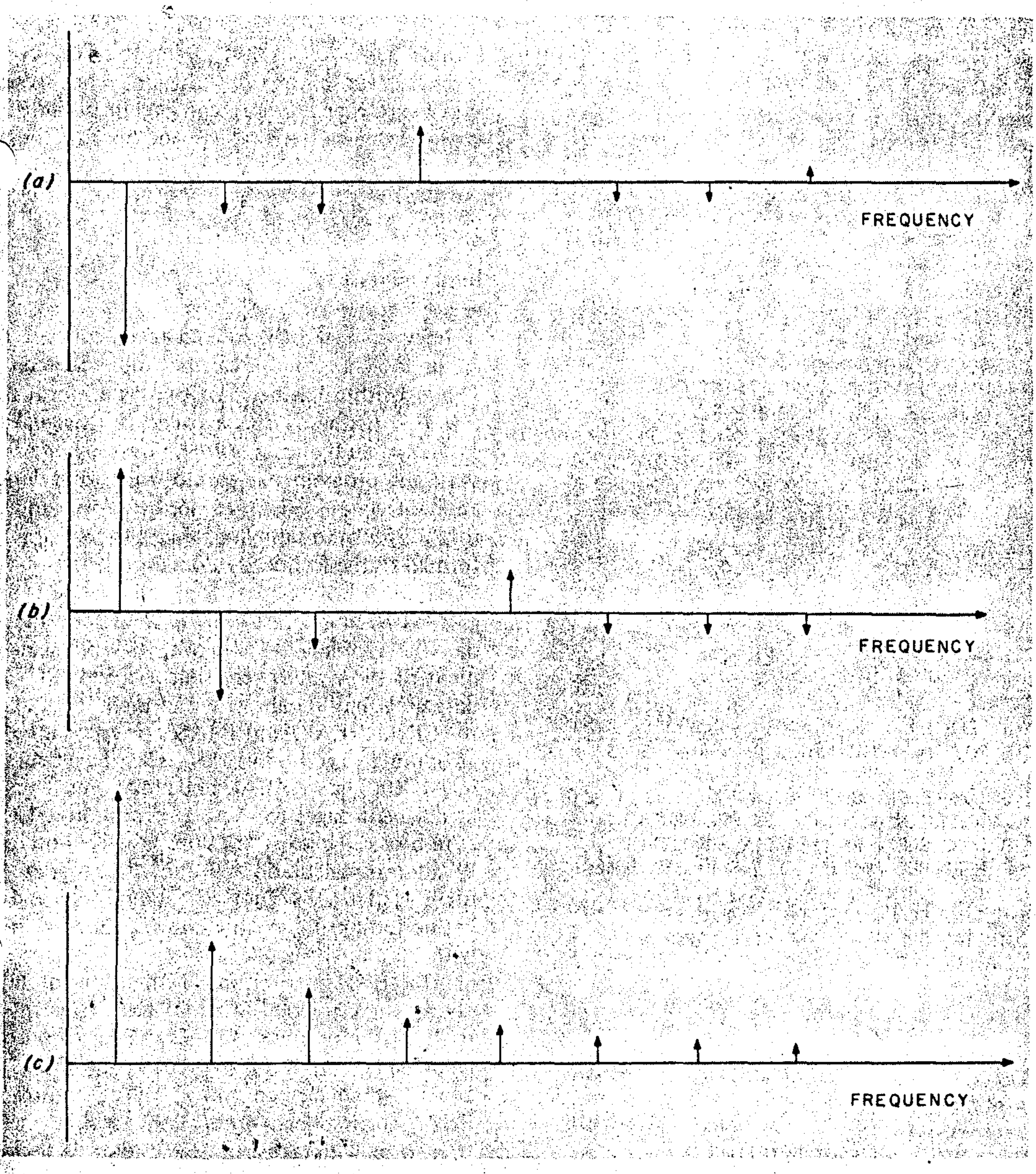


Figure 1: Fast Fourier transform of a square wave using the author's technique. The real (or sine) part of the transform is shown in (a). The imaginary (or cosine) part of the transform is shown in (b). The resulting transform is at (c). The resulting transform values are normally found by taking the square root of the sum of the squares of the cosine and sine elements. In order to save computational time, however, the author takes the sum of the absolute values of the terms, which introduces slight errors into the relative magnitudes of the components.

forms and ultimately requires 8 by 2 by 256 complex products, or 16,384 binary multiplications (1/16 the number of previous multiplications). Even greater savings are realized as the number of points increases.

Each of the simplified transforms operates on the data in pairs of complex points. The real and imaginary parts of a pair are transformed and the new values placed back in the array so that the transform is performed "in place." The algorithm then moves on to the next pair until all pairs have been transformed. The process is repeated for each of the eight stages of our 256 point transform, but on each pass the distance between pairs is changed.

On the first pass, adjacent points are paired. After completing a pair the algorithm skips down to the next. In a sense, the data has been split into 128 adjacent 2 point transforms. These 128 groups are known as cells. On each subsequent pass the distance between elements of the pair is doubled. In the second pass there are 64 cells, each four elements wide. On the final pass there is only one cell containing all 256 elements.

This process of forming pairs and cells causes the elements of the array to become scrambled. On the final pass the data is completely mixed up and must be sorted out before it can be used. The way it is scrambled is very interesting, though. If each element is assigned a binary number that represents its location in the array, the scrambled data makes it appear that the computer has read this binary address backwards. It is as if the binary word were swapped end for end so the most significant bit (MSB) appears where the least significant bit (LSB) should be.

This rearrangement of the data may be corrected by swapping each data point with its bit reverse addressed mate. The procedure

Listing 1: Routine in 6800 assembly language to perform a 256 point fast Fourier transform.

```
                     NAM    FFT#2
                     OPT    O,S,NOGEN
          **********************
          **  FAST FOURIER    **
          **    TRANSFORM     **
          **   SUBROUTINE     **
          **********************
          **   BY R.H.LORD    **
          **  21 APRIL, 1978  **
          **********************
          **
          ** THIS SUBROUTINE PERFORMS A 256 POINT FFT
          ** ON THE DATA IN THE INPUT DATA TABLE.
          ** INPUT DATA IS ASSUMED TO BE TWO'S COMPLEMENT.
          ** THE SUBROUTINE GENERATES A COSINE (REAL) AND SINE
          ** (IMAGINARY) DATA TABLE AT "REAL" AND "IMAG"
          ** THE RESULTANT TRANSFORM DATA IS 128 POINTS
          ** SYMMETRICALLY REFLECTED ABOUT THE CENTER OF
          ** THE 256 POINT TABLE.
          **
          ** THE SUBROUTINE ASSUMES THAT THE INPUT DATA
          ** IS ALL REAL AND THEREFORE DOES NOT MANIPULATE
          ** THE IMAGINARY PORTION UNTIL AFTER THE FIRST
          ** PASS.
          **
          ** ALL DATA AREAS MUST BE ON PAGE BOUNDARIES (XX00)
          ** SINCE THE ROUTINE MANIPULATES ONLY THE LSB'S.
          **
          ** THE TWO'S COMPLEMENT MULTIPLICATION IS KEPT AS A
          ** SEPARATE SUBROUTINE.  IT MAY BE PERFORMED WITH
          ** A CONVENTIONAL SOFTWARE MULTIPLY SUBROUTINE
          ** OR WITH A HARDWARE MULTIPLIER FOR HIGHER SPEED.
          **
          ** THE SUBROUTINE SCALES THE DATA WHENEVER
          ** IT ANTICIPATES OVERFLOW. THE SCALE FACTOR
          ** COUNT IS AVAILABLE IN "SCLFCT".
          **
          **
          **
          **
          **********************
          **    DATA AREAS    **
          **********************
     0800 INPUT  EQU    $0800    INPUT DATA TABLE
     0500 REALT  EQU    $0500    "REAL" DATA TABLE
     0600 IMAGT  EQU    $0600    "IMAG" DATA TABLE
     0400 SINET  EQU    $0400    SINE LOOKUP TABLE
          **********************
0020                 ORG    $0020
          **********************
          ** BASE PAGE PTRS  **
          **********************
0020 0002 RLPT1  RMB    2        "REAL" DATA POINTERS
0022 0002 RLPT2  RMB    2
0024 0002 IMPT1  RMB    2        "IMAG." DATA POINTERS
0026 0002 IMPT2  RMB    2
0028 0002 SINPT  RMB    2        SINE TABLE POINTER
002A 0001 CELNUM RMB    1        CELLS FOR THIS PASS
002B 0001 CELCT  RMB    1        CELL COUNTER FOR PASS
002C 0001 PAIRNM RMB    1        PAIRS/CELL
002D 0001 CELDIS RMB    1        CELL OFFSET(DISTANCE)
002E 0001 DELTA  RMB    1        ANGLE INCREMENT
002F 0001 SCLFCT RMB    1        SCALE FACTOR CTR.
0030 0001 COSA   RMB    1        TEMPORARY COSINE
0031 0001 SINA   RMB    1        TEMPORARY SINE
0032 0001 TREAL  RMB    1        TEMP.  REAL DATA
0033 0001 TIMAG  RMB    1        TEMP.  IMAG DATA
0034 0001 MSBY   RMB    1        MULTIPLY MSB
0035 0001 LSBY   RMB    1        MULTIPLY LSB
0036 0004 MPA    RMB    4        SOFTWARE MPY ACCUM.
          *****************************
```

is called "bit swapping" and may be performed either at the end of the fast Fourier transform or before it is begun. The pretransform swap is more convenient because less points need be swapped and because the vector rotation within each cell is simpler. In the posttransform version the vector angles would also have to be bit swapped.

Implementation

Now that we have looked at the concept, let us look at how it can be implemented. The algorithm has been written as a subroutine (see listing 1) to be called by a signal gathering and display program. It assumes that this program has stored some time dependent data in 2's complement form and that a 256 byte sample of this is to be transformed to the frequency domain.

The fast Fourier transform subroutine begins with an address lookup table for the data areas. This table makes the reassignment of these areas very simple. The INPUT data area may be anywhere in memory, but the SINE, REAL, and IMAG arrays must be at address page boundaries (ie: at hexadecimal XX00), and REAL and IMAG must be in adjacent pages forming a continuous 512 byte block. These restrictions greatly simplify address calculation within the program. SINE is the address of a 256 byte sine and cosine lookup table which must be loaded in with the transform subroutine.

The first instruction of the subroutine clears the variable SCLFCT which keeps track of the number of times the data has to be scaled to prevent overflow. The IMAG array is then cleared and at MOVE the INPUT data is copied into REAL, where the transform will take place. The data is then prescrambled to put it in bit reverse order for the transform process. The bit reversed address is calculated by rotating the least significant bit of the address into the carry and rotating the reversed address out in the opposite direction. The new address is compared with the first address to prevent swapping the data back to the original order, then the two array elements are exchanged.

Once the swapping is complete, the data is ready to be transformed. The fast Fourier transform is performed in eight separate passes; before each pass begins, the data is tested by SCALE to prevent any overflow. For the first pass there are 128 cells formed by adjacent pairs of data. In this pass the vector angle steps in multiples of 180 degrees. This means that all the sine terms are 0 and the cosine terms are either +1 or −1. Also there is no data yet in the IMAG array. The general equations thus become greatly simplified and the pass is reduced to addition and subtraction among elements of the

Listing 1, continued:

```
             **  START OF TRANSFORM    **
             ****************************
0200                ORG    $0200
0200 20 08          BRA    START     JUMP AROUND PARAMETERS
             ****************************
             **  ADDRESS LOOK-UP TABLE  **
             **    FOR DATA AREAS       **
             ****************************
0202 0800    INPD   FDB    INPUT     SET UP DATA AREAS
0204 0500    REAL   FDB    REALT
0206 0600    IMAG   FDB    IMAGT
0208 0400    SINE   FDB    SINET
             ****************************
             **
020A 7F 002F START  CLR    SCLFCT    NOTHING SCALED YET
             **
             ****************************
             **    INPUT DATA SET-UP    **
             ****************************
020D FE 0206 CLEAR  LDX    IMAG      CLEAR OUT IMAG.
0210 5F             CLR B            SET UP COUNTER
0211 6F 00   CLR1   CLR    0,X       CLEAR MEMORY
0213 08             INX
0214 5A             DEC B
0215 26 FA          BNE    CLR1
0217 FE 0202 MOVE   LDX    INPD      SET UP POINTERS
021A DF 20          STX    RLPT1
021C FE 0204        LDX    REAL
021F DF 22          STX    RLPT2
0221 DE 20   MOV1   LDX    RLPT1     MOVE INPUT DATA
0223 A6 00          LDA A  0,X       TO "REAL" ARRAY
0225 08             INX
0226 DF 20          STX    RLPT1
0228 DE 22          LDX    RLPT2
022A A7 00          STA A  0,X
022C 7C 0023        INC    RLPT2+1
022F 26 F0          BNE    MOV1      TEST PAGE OVERFLOW
             ****************************
             **  PRE-TRANSFORM BIT SWAP **
             ****************************
0231 FE 0204        LDX    REAL      SET UP DATA POINTERS
0234 DF 20          STX    RLPT1
0236 DF 22          STX    RLPT2
0238 C6 08   BITREV LDA B  #8        SET BIT COUNTER
023A 96 21          LDA A  RLPT1+1   GET POINTER 1
023C 46      BRV1   ROR A            REVERSE BIT ORDER
023D 79 0023        ROL    RLPT2+1   FOR SECOND POINTER
0240 5A             DEC B            COUNT BITS
0241 26 F9          BNE    BRV1
0243 96 23          LDA A  RLPT2+1   GET REVERSED BYTE
0245 91 21          CMP A  RLPT1+1   COMPARE WITH #1
0247 25 0E          BCS    SWP1      BRANCH IF ALREADY SWAPPED
0249 DE 20   SWAP   LDX    RLPT1     GET POINTER 1
024B A6 00          LDA A  0,X       GET VAL 1
024D DE 22          LDX    RLPT2     GET POINTER 2
024F E6 00          LDA B  0,X       GET VAL 2
0251 A7 00          STA A  0,X       REPLACE WITH VAL 1
0253 DE 20          LDX    RLPT1     GET FIRST POINTER
0255 E7 00          STA B  0,X       COMPLETE SWAP
0257 7C 0021 SWP1   INC    RLPT1+1   DO NEXT POINT PAIR
025A 26 DC          BNE    BITREV    UNLESS ALL ARE DONE
             *************************************
             **      FFT  FIRST PASS            **
             *************************************
             **  SINCE IN PASS 1 ALL ANGLES     **
             **    ARE MULTIPLES OF 180 DEG.    **
             ** THERE ARE NO PRODUCT TERMS.     **
             **    AND NO IMAGINARY TERMS YET   **
             **  HENCE A FAST VERSION OF PASS 1 **
             *************************************
025C BD 0333 PASS1  JSR    SCALE     SCALE IF ANY OVER-RANGE DATA
```

REAL array. Considerable time is saved by making this pass separate and bypassing the unneeded table lookup and multiply routines.

Once this pass is completed, the arithmetic gets much more complex. The remaining seven passes are performed by a general fast Fourier transform algorithm. It begins at FPASS by setting up 64 cells of four elements with the pairs separated by two units. The vector angle is set to increment by 90 degrees by setting DELTA to 64. At NPASS the pointers are set up for the first cell and the pass then begins with a sine and cosine table lookup. The complex data pair is then processed using the standard fast Fourier transform equations:

$$\begin{aligned} TR &= RN\,COS(w) + IN\,SIN(w) \\ TI &= IN\,COS(w) - RN\,SIN(w) \end{aligned}$$

$$\begin{aligned} RM' &= RM + TR \qquad & RN' &= RM - TR \\ IM' &= IM + TI \qquad & IN' &= IM - TI \end{aligned}$$

After each pair has been transformed the angle is incremented by DELTA and the next pair processed. When all pairs in a cell have been transformed the routine moves down to the next cell and returns to NCELL to continue the process. When the last cell has been done, CELCT becomes 0 and the pass is complete.

At the end of each pass the number of cells and the angle increment are divided in half and the pair separation and number of pairs per cell are doubled. The whole process is then repeated by branching to NPASS until the end of the last pass when the number of cells becomes 0. The routine then branches to DONE and returns to the calling program.

The SCALE subroutine is used to anticipate and prevent overflow of the 8 bit data. It is called before each pass and begins by testing the value of each data point. If any point exceeds the range of -64 to +64 the subroutine branches to SCL4 where the entire array is scaled down by a factor of 2. The variable SCLFCT is incremented to indicate the total number of times the data has been scaled.

The multiply routine has been placed at the end of the program to make substitution of other versions easy. The original program was written for a hardware multiplier similar to the device described by Bryant and Swasdee in April 1978 BYTE, page 28. To eliminate the need for such exotic hardware, a software multiply routine has been substituted with some increase in transform time. After the multiplication is completed

Listing 1, continued:

```
025F FE 0204        LDX    REAL     SET UP POINTERS
0262 DF 20          STX    RLPT1
0264 DE 20   PA1    LDX    RLPT1    GET POINTER
0266 A6 00          LDA A  0,X      GET RM
0268 E6 01          LDA B  1,X      AND RN
026A 36             PSH A           SAVE RM
026B 1B             ABA             RM′=RM+RN
026C A7 00          STA A  0,X      STORE NEW RM′
026E 32             PUL A           GET OLD RM
026F 10             SBA             RN′=RM-RN
0270 A7 01          STA A  1,X      STORE RN′
0272 7C 0021        INC    RLPT1+1  MOVE TO NEXT PAIR
0275 7C 0021        INC    RLPT1+1
0278 26 EA          BNE    PA1      KEEP GOING TILL DONE
             ********************************
             **  COMPUTATION OF FFT        **
             **       PASS 2 THRU N        **
             ********************************
027A 86 40   FPASS  LDA A  #64      SET UP PARAMETERS
027C 97 2A          STA A  CELNUM   FOR CELL COUNT
027E 97 2E          STA A  DELTA    AND ANGLE
0280 86 02          LDA A  #2       AND FOR
0282 97 2C          STA A  PAIRNM   PAIRS/CELL
0284 97 2D          STA A  CELDIS   DISTANCE BETWEEN PAIRS
0286 BD 0333 NPASS  JSR    SCALE    KEEP DATA IN RANGE
0289 96 2A          LDA A  CELNUM   GET NUMBER OF CELLS
028B 97 2B          STA A  CELCT    PUT IN COUNTER
028D FE 0204        LDX    REAL     SET UP POINTERS
0290 DF 20          STX    RLPT1
0292 DF 22          STX    RLPT2
0294 FE 0206        LDX    IMAG
0297 DF 24          STX    IMPT1
0299 DF 26          STX    IMPT2
029B FE 0208 NCELL  LDX    SINE
029E DF 28          STX    SINPT
02A0 D6 2C          LDA B  PAIRNM   GET PAIRS/CELL CTR.
02A2 96 21   NC1    LDA A  RLPT1+1  GET POINTER 1 LSBY
02A4 9B 2D          ADD A  CELDIS   ADD PAIR OFFSET
02A6 97 23          STA A  RLPT2+1  SET BOTH POINTER 2′S
02A8 97 27          STA A  IMPT2+1
02AA 37             PSH B           SAVE PAIR CTR
02AB DE 28          LDX    SINPT    SET UP SINE LOOKUP
02AD A6 00          LDA A  0,X      GET COSINE OF ANGLE
02AF 97 30          STA A  COSA     SAVE ON BASE PAGE
02B1 A6 40          LDA A  64,X     GET SINE
02B3 97 31          STA A  SINA     AND SAVE IT
02B5 DE 22          LDX    RLPT2    GET "REAL" POINTER 2
02B7 A6 00          LDA A  0,X      GET RN
02B9 36             PSH A           SAVE IT
02BA D6 30          LDA B  COSA     GET COSINE
02BC BD 036A        JSR    MPY      MAKE RN*COS(A)
02BF 97 32          STA A  TREAL    SAVE IT
02C1 32             PUL A           RESTORE RN
02C2 D6 31          LDA B  SINA     GET SINE
02C4 BD 036A        JSR    MPY      RN*SIN(A)
02C7 97 33          STA A  TIMAG
02C9 DE 26          LDX    IMPT2    GET IMAG. POINTER 2
02CB A6 00          LDA A  0,X      GET IN
02CD 36             PSH A           SAVE IT
02CE D6 31          LDA B  SINA     GET SINE
02D0 BD 036A        JSR    MPY      IN*SIN(A)
02D3 9B 32          ADD A  TREAL    TR=RN*COS+IN*SIN
02D5 97 32          STA A  TREAL
02D7 32             PUL A           RESTORE IN
02D8 D6 30          LDA B  COSA     GET COSINE
02DA BD 036A        JSR    MPY      IN*COS(A)
02DD 90 33          SUB A  TIMAG    TI=IN*COS-RN*SIN
02DF 97 33          STA A  TIMAG
02E1 DE 20          LDX    RLPT1
02E3 A6 00          LDA A  0,X      GET RM
02E5 16             TAB             SAVE IT
```

the data must be scaled up by a factor of 2. This is because the sine and cosine terms represent fractional binary values. The least significant bit is shifted in from the lower byte to preserve accuracy.

Analyzing the Results

After working with all this mathematics and software, what do you end up with? We started with a 256 point time domain sample in REAL. The fast Fourier transform converts this to a frequency domain sample corresponding to the spectrum of the input. The first element of each array represents the DC component of the input. The next element represents the sine wave with period equal to the duration of the input sample. Each remaining element depicts a multiple of this frequency until the middle of the array is reached, representing 128 cycles per period. The remainder of the array is symmetrical to the first 128 points.

Each element in the REAL and IMAG arrays represents information about one frequency component of the input sample. But why do we end up with two arrays, and what do the cosine terms of REAL and the sine terms of IMAG really mean to us? Usually this information is described in terms of amplitude and phase of the component, and often the phase information is of little interest. The cosine and sine terms represent the X and Y components of a vector with length and angle equal to the amplitude and phase terms that we are after. All we have to do is find the length of the vector from the square root of the sum of squares of the cosine and sine terms.

The only problem is that this calculation requires almost as much time as the transform, due to the square root. If we bypass the root and display the sum of squares (the power spectrum) we miss most of the detail of the lesser components. I have found that the highly unmathematical solution of displaying the sum of the absolute values is fairly satisfactory, although it introduces some error in the relative amplitude of peaks. This value is then sent to a digital to analog converter for display on an oscilloscope.

Putting the Fast Fourier Transform to Work

This program has a number of interesting applications for speech recognition, image processing, and the synthesis of musical instruments. A recent issue of *The Computer Music Journal* even describes a program for transcribing recordings back into sheet music (see bibliography, page 118).

MAXIMIZE
YOUR MICRO!
8080/Z80 System Software

Purchasing a microcomputer system, even at today's low prices, is a significant investment. And to utilize that investment to its full extent requires a solid base in system software. Don't just accept what comes with your hardware... there's a better alternative!

OPUS/ONE: Business-oriented, block-structured high-level language. Includes such capabilities as extended arithmetic precision (up to 55 digits), multi-character variable names, and easy to use string operations. Includes a built-in DOS with random access files.
OPUS/ONE $99.00

OPUS/TWO: Extends the capabilities of OPUS/ONE with such features as error trapping, machine code and OPUS subroutine calls, overlays, and more disc file commands.
OPUS/TWO $195.00

FORTE: An OPUS run-only system monitor, perfect for end-user installations. Runs all OPUS Language programs.
FORTE $60.00

S.O.S.: (Single-user Operating System) A full-function DOS which *includes* OPUS/THREE, a text editor (TEXTED), an assembler (ASSEMBL), and a package of useful system utility programs.
S.O.S. $385.00

TEMPOS: The ultimate microcomputer system software package. A multi-user/multi-tasking DOS which will handle up to 7 interrupt-driven terminals simultaneously, in a true time-sharing environment. Includes OPUS/THREE, TEXTED, ASSEMBL, and many utility programs.
TEMPOS $785.00

All packages are upward-compatible. That is, programs and data developed under OPUS/ONE may be run at any higher level, up to and including TEMPOS.

Standard device drivers are available for many common peripherals; all packages include System Generation capability, allowing the user to *interactively* add drivers for any I/O device, including disc drives.

Like to know more? Circle the inquiry number below or contact your dealer for your free copy of our system software brochure! For complete information, order your user's manual now, and we'll apply the price toward purchase of the software. Please add $1.50 per manual (set) for shipping/handling within the U.S. (Master Charge and VISA accepted).
OPUS User's Manual $12.50
S.O.S. Set (inc. OPUS Manual) ..... [illegible]
TEMPOS Set (inc. OPUS Manual) ..... [illegible]

1642 S. Parker Road, Suite 300
[illegible] Colorado 80231

Listing 1, continued:

```
00215 02E6 9B 32          ADD A  TREAL     RM'=RM+TR
00216 02E8 A7 00          STA A  0,X
00217 02EA DE 22          LDX    RLPT2
00218 02EC D0 32          SUB B  TREAL     RN'=RM-TR
00219 02EE E7 00          STA B  0,X
00220 02F0 DE 24          LDX    IMPT1
00221 02F2 A6 00          LDA A  0,X       GET IM
00222 02F4 16             TAB              SAVE IT
00223 02F5 9B 33          ADD A  TIMAG     IM'=IM+TI
00224 02F7 A7 00          STA A  0,X
00225 02F9 DE 26          LDX    IMPT2
00226 02FB D0 33          SUB B  TIMAG     IN'=IM-TI
00227 02FD E7 00          STA B  0,X
00228 02FF 96 29          LDA A  SINPT+1   INCREMENT ANGLE
00229 0301 9B 2E          ADD A  DELTA
00230 0303 97 29          STA A  SINPT+1
00231 0305 7C 0021        INC    RLPT1+1   INCREMENT POINTERS
00232 0308 7C 0025        INC    IMPT1+1
00233 030B 33             PUL B            GET PAIR COUNTER
00234 030C 5A             DEC B            DECREMENT
00235 030D 26 93          BNE    NC1       DO NEXT PAIR
00236 030F 96 21          LDA A  RLPT1+1   GET POINTERS
00237 0311 9B 2D          ADD A  CELDIS    ADD CELL OFFSET
00238 0313 97 21          STA A  RLPT1+1
00239 0315 97 25          STA A  IMPT1+1
00240 0317 7A 002B        DEC    CELCT     DECR. CELL COUNTER
00241 031A 27 03          BEQ    NP1       NEXT PASS?
00242 031C 7E 029B        JMP    NCELL     NO, DO NEXT CELL
00243                **
00244                ** CHANGE PARAMETERS FOR NEXT PASS **
00245                **
00246 031F 74 002A NP1    LSR    CELNUM    HALF AS MANY CELLS
00247 0322 27 0C          BEQ    DONE      NO MORE CELLS
00248 0324 78 002C        ASL    PAIRNM    TWICE AS MANY PAIRS
00249 0327 78 002D        ASL    CELDIS    TWICE AS FAR APART
00250 032A 74 002E        LSR    DELTA     HALF THE ANGLE
00251 032D 7E 0286        JMP    NPASS     DO NEXT PASS
00252                *********************************
00253                **  END OF FFT ROUTINE        **
00254                *********************************
00255                **
00256 0330 39      DONE   RTS              EXIT FFT SUBROUTINE
00257 0331 0002           RMB    2         ROOM FOR JUMP EXIT
00258                **
00259                ******************************
00260                **  OVER-RANGE DATA SCALE  **
00261                ******************************
00262 0333 FE 0204 SCALE  LDX    REAL      SET UP DATA POINTER
00263 0336 5F             CLR B            SET UP PAIR CTR
00264 0337 37      SCL1   PSH B            SAVE PAIR CTR
00265 0338 C6 02          LDA B  #2        SET UP PAIR
00266 033A A6 00   SCL2   LDA A  0,X       GET DATA
00267 033C 08             INX              BUMP POINTER
00268 033D 81 C0          CMP A  #$C0      TEST LOWER LIMIT
00269 033F 22 [illegible] BHI    SCL3      SKIP TO NEXT POINT
00270 0341 81 40          CMP A  #$40      TEST UPPER LIMIT
00271 0343 24 08          BCC    SCL4      SCALE IF OUT OF RANGE
00272 0345 5A      SCL3   DEC B            TEST NEXT POINT
00273 0346 26 F2          BNE    SCL2
[illegible]
```

To get meaningful information from the transform, the input data must be sampled judiciously. While this program in theory is capable of analyzing 128 harmonics of a given sample, this is only true when the input represents exactly one complete cycle of the waveform being analyzed. Most data just doesn't come packaged that way.

To accurately measure the pitch of a sound you must sample many cycles. To analyze harmonics you want to sample few. The best result for real data will always be a compromise between range (bandwidth) and resolution. Both can be increased only by analyzing more points, which takes more time.

After experimenting with one sample at a time you will probably want to try continuous analysis. The input data pointer at hexadecimal address 0202 can be moved through an input buffer by the program that calls the transform. At roughly three seconds per transform, the data cannot suitably be analyzed in real time. A sample of a few seconds of data can be continuously analyzed and the changes slowly displayed. This is probably most easily accomplished by transferring the "sum of absolute value" data to a display buffer which is then scanned by an interrupt driven display program.

Bigger, Better, and Faster

Like most software, this program exists to be rewritten. No attempt was made to optimize execution speed. Preliminary experiments with an MMI 67658 hardware multiplier took slightly under one second. This relatively minor improvement was probably due to the time wasted in moving the data in and out of the multiplier. Perhaps it can be streamlined to the extent that a continuous display can be created. I plan to try a version for the 6502 microprocessor with hope of adding still more speed.

The algorithm is simple enough so that conversion should be easy. Enterprising 8080 and Z-80 enthusiasts shouldn't have too much trouble adapting the principles to their computers, either. Conversion to double precision or 512 to 1024 points should also be possible, although the present addressing scheme would have to be abandoned.

I hope this program will provide you with a tool that will be a lot of fun to play with. Please write and tell me what uses you find for it and any improvements you would like to suggest.

Continued on page 118

Listing 1, continued:

```
035C 44             LSR A            DIVIDE IT BY 2
035D 80 40          SUB A  #$40      MAKE IT 2'S COMP.
035F A7 00          STA A  0,X
0361 08             INX              BUMP POINTER
0362 5A             DEC B            NEXT POINT
0363 26 F3          BNE    SCL6
0365 33             PUL B
0366 5A             DEC B            NEXT PAIR
0367 26 EC          BNE    SCL5
0369 39             RTS              RETURN
             ********************************
             **  2'S COMP. MULTIPLY SUBR.  **
             ********************************
036A 97 37   MPY    STA A  MPA+1     STORE MULTIPLIER
036C D7 39          STA B  MPA+3     AND MULTIPLICAND
036E 4F             CLR A
036F 97 36          STA A  MPA       CLEAR MSB'S
0371 97 38          STA A  MPA+2
0373 97 34          STA A  MSBY      CLEAR PRODUCT
0375 97 35          STA A  LSBY
0377 5D             TST B
0378 2C 03          BGE    MPY1      NEGATIVE MULTIPLICAND ?
037A 73 0038        COM    MPA+2     EXTEND NEG TO MSB
037D 7D 0037 MPY1   TST    MPA+1
0380 2C 03          BGE    MPY2      NEG MULTIPLIER ?
0382 73 0036        COM    MPA       EXTEND NEG TO MSB
0385 C6 0F   MPY2   LDA B  #15       SET UP COUNTER
0387 77 0036 MPY3   ASR    MPA       SHIFT X RIGHT
038A 76 0037        ROR    MPA+1
038D 24 0C          BCC    MPY4      BIT WAS ZERO
038F 96 39          LDA A  MPA+3     ADD Y TO PRODUCT
0391 9B 35          ADD A  LSBY
0393 97 35          STA A  LSBY
0395 96 38          LDA A  MPA+2     MSB'S
0397 99 34          ADC A  MSBY
0399 97 34          STA A  MSBY
039B 78 0039 MPY4   ASL    MPA+3     SHIFT Y LEFT
039E 79 0038        ROL    MPA+2
03A1 5A             DEC B
03A2 26 E3          BNE    MPY3
             **
             ** SCALE IT UP **
             **
03A4 96 34          LDA A  MSBY
03A6 79 0035        ROL    LSBY
03A9 49             ROL A
             **
             ** RETURN WITH PRODUCT IN A
             **
03AA 39             RTS
             ********************************
             **     END OF FFT PROGRAM     **
             ********************************
                    END

    INPUT   0800   REALT   0500   IMAGT   0600   SINET   0400
    RLPT1   0020   RLPT2   0022   IMPT1   0024   IMPT2   0026
    SINPT   0028   CELNUM  002A   CELCT   002B   PAIRNM  002C
    CELDIS  002D   DELTA   002E   SCLFCT  002F   COSA    0030
    SINA    0031   TREAL   0032   TIMAG   0033   MSBY    0034
    LSBY    0035   MPA     0036   INPD    0202   REAL    0204
    IMAG    0206   SINE    0208   START   020A   CLEAR   020D
    CLR1    0211   MOVE    0217   MOV1    0221   BITREV  0238
    BRV1    023C   SWAP    0249   SWP1    0257   PASS1   025C
    PA1     0264   FPASS   027A   NPASS   0286   NCELL   029B
    NC1     02A2   NP1     031F   DONE    0330   SCALE   0333
    SCL1    0337   SCL2    033A   SCL3    0345   SCL4    034D
    SCL5    0355   SCL6    0358   MPY     036A   MPY1    037D
    MPY2    0385   MPY3    0387   MPY4    039B
TOTAL ERRORS 00000
```

Continued from page 117

BIBLIOGRAPHY

1. Brigham, E Oran, *The Fast Fourier Transform,* Prentice-Hall, Englewood Cliffs NJ, 1974.
2. Bryant, J, and Swasdee, M, "How to Multiply in a Wet Climate," BYTE, volume 3, number 4, April 1978, page 28.
3. Cooper, James W, *The Minicomputer in the Laboratory,* John Wiley and Sons Inc, New York, 1977.
4. Moorer, J, "On the Transcription of Musical Sound by Computer," *Computer Music Journal,* volume 1, number 4, November 1977, page 32.
5. Stearns, Samuel D, *Digital Signal Analysis,* Hayden Book Co Inc, Rochelle Park NJ, 1975.

Listing 2: The object code listing in hexadecimal format of the assembly language program given in listing 1. This listing can be used to manually enter the program or as a confirmation copy for the PAPERBYTE^{tm} bar code representation given in figure 2. The format used for this listing is a 2 byte address field, followed by up to 16 bytes of data, with a 1 byte check digit at the end of each line. Note that the data in hexadecimal locations 0400 to 04FF constitute the sine and cosine lookup table which must be loaded with the transform subroutine.

```
0200 20 08 08 00 05 00 06 00 04 00 7F 00 2F FE 02 06   F3
0210 5F 6F 00 08 5A 26 FA FE 02 02 DF 20 FE 02 04 DF   34
0220 22 DE 20 A6 00 08 DF 20 DE 22 A7 00 7C 00 23 26   39
0230 F0 FE 02 04 DF 20 DF 22 C6 08 96 21 46 79 00 23   5B
0240 5A 26 F9 96 23 91 21 25 0E DE 20 A6 00 DE 22 E6   A1
0250 00 A7 00 DE 20 E7 00 7C 00 21 26 DC BD 03 33 FE   1C
0260 02 04 DF 20 DE 20 A6 00 E6 01 36 1B A7 00 32 10   CA
0270 A7 01 7C 00 21 7C 00 21 26 EA 86 40 97 2A 97 2E   3E
0280 86 02 97 2C 97 2D BD 03 33 96 2A 97 2B FE 02 04   88
0290 DF 20 DF 22 FE 02 06 DF 24 DF 26 FE 02 08 DF 28   1D
02A0 D6 2C 96 21 9B 2D 97 23 97 27 37 DE 28 A6 00 97   73
02B0 30 A6 40 97 31 DE 22 A6 00 36 D6 30 BD 03 6A 97   81
02C0 32 32 D6 31 BD 03 6A 97 33 DE 26 A6 00 36 D6 31   46
02D0 BD 03 6A 9B 32 97 32 32 D6 30 BD 03 6A 90 33 97   7C
02E0 33 DE 20 A6 00 16 9B 32 A7 00 DE 22 D0 32 E7 00   4A
02F0 DE 24 A6 00 16 9B 33 A7 00 DE 26 D0 33 E7 00 96   B7
0300 29 9B 2E 97 29 7C 00 21 7C 00 25 33 5A 26 93 96   CC
0310 21 9B 2D 97 21 97 25 7A 00 2B 27 03 7E 02 9B 74   BB
0320 00 2A 27 0C 78 00 2C 78 00 2D 74 00 2E 7E 02 86   4E
0330 39 00 00 FE 02 04 5F 37 C6 02 A6 00 08 81 C0 22   AC
0340 04 81 40 24 08 5A 26 F2 33 5A 26 EB 39 33 7C 00   E9
0350 2F FE 02 04 5F 37 C6 02 A6 00 8B 80 44 80 40 A7   ED
0360 00 08 5A 26 F3 33 5A 26 EC 39 97 37 D7 39 4F 97   17
0370 36 97 38 97 34 97 35 5D 2C 03 73 00 38 7D 00 37   87
0380 2C 03 73 00 36 C6 0F 77 00 36 76 00 37 24 0C 96   CD
0390 39 9B 35 97 35 96 38 99 34 97 34 78 00 39 79 00   65
03A0 38 5A 26 E3 96 34 79 00 35 49 39                  95

0400 7F 7F 7F 7F 7F 7F 7E 7E 7D 7D 7C 7B 7A 79 78 77   C9
0410 76 75 73 72 71 6F 6D 6C 6A 68 66 65 63 61 5E 5C   A4
0420 5A 58 56 53 51 4E 4C 49 47 44 41 3F 3C 39 36 33   78
0430 31 2E 2B 28 25 22 1F 1C 19 16 12 0F 0C 09 06 03   A2
0440 00 FD FA F7 F4 F1 EE EA E7 E4 E1 DE DB D8 D5 D2   8F
0450 CF CD CA C7 C4 C1 BF BC B9 B7 B4 B2 AF AD AA A8   B1
0460 A6 A4 A2 9F 9D 9B 9A 98 96 94 93 91 8F 8E 8D 8B   78
0470 8A 89 88 87 86 85 84 83 83 82 82 81 81 81 81 81   40
0480 81 81 81 81 81 81 82 82 83 83 84 85 86 87 88 89   37
0490 8A 8B 8D 8E 8F 91 93 94 96 98 9A 9B 9D 9F A2 A4   5C
04A0 A6 A8 AA AD AF B2 B4 B7 B9 BC BF C1 C4 C7 CA CD   88
04B0 CF D2 D5 D8 DB DE E1 E4 E7 EA EE F1 F4 F7 FA FD   5E
04C0 00 03 06 09 0C 0F 12 16 19 1C 1F 22 25 28 2B 2E   71
04D0 31 33 36 39 3C 3F 41 44 47 49 4C 4E 51 53 56 58   4F
04E0 5A 5C 5E 61 63 65 66 68 6A 6C 6D 6F 71 72 73 75   88
04F0 76 77 78 79 7A 7B 7C 7D 7D 7E 7E 7F 7F 7F 7F 7F   C0
```

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4
0 1 3 4 6 7 9 A C D F 0 2 3 5 6 8 9 0 1 2 4 5 7 8 A B D E F
0 A 2 B 3 C 4 B 3 B 3 B 3 C 4 C 4 C 0 6 E 6 C 3 C 4 A 2 A F

Figure 2: PAPERBYTE^tm bar code version of listing 2. For details on how to read bar codes, see Bar Code Loader, *a PAPERBYTE^tm book by Ken Budnick.*

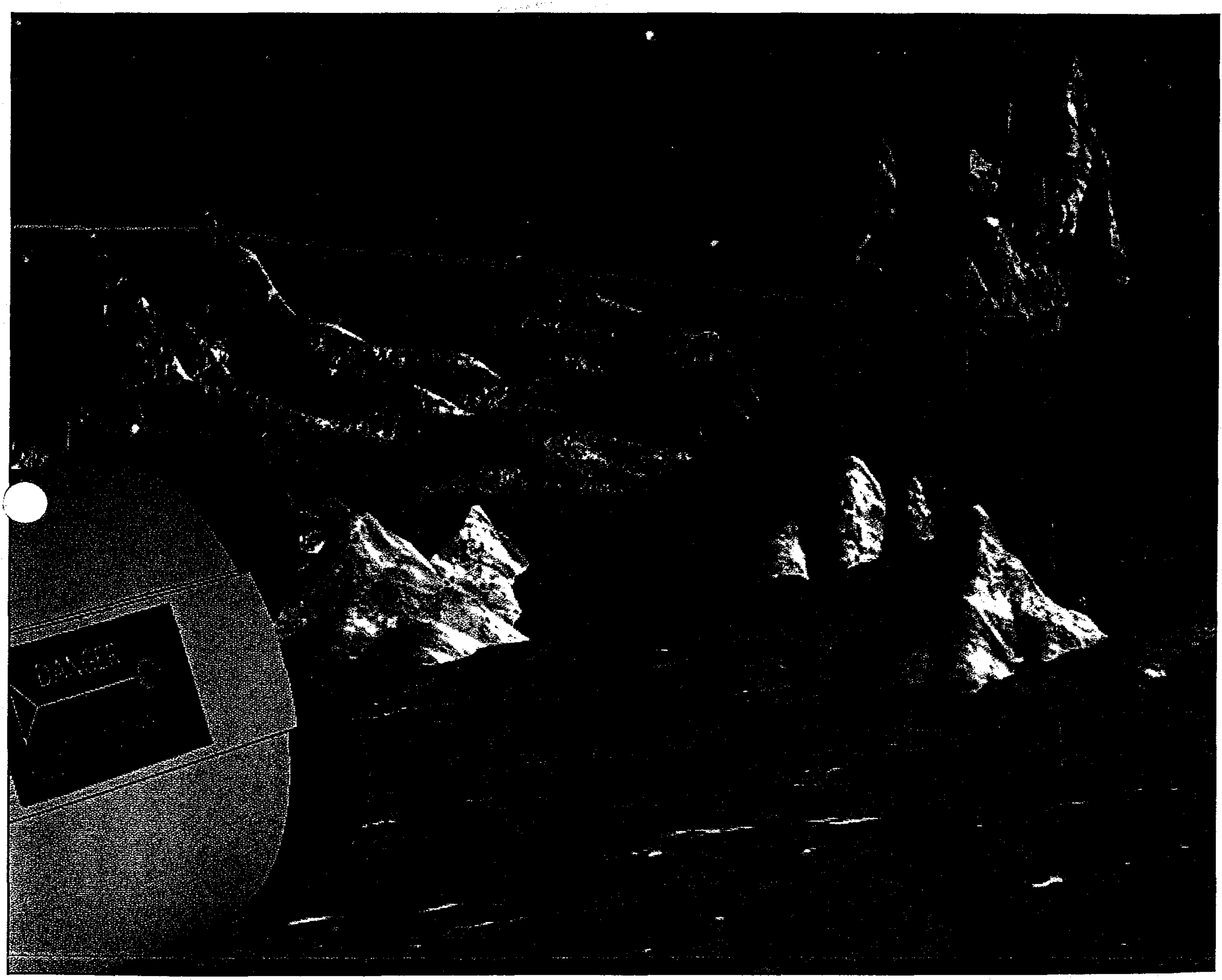
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 ■

COMMUNICATIONS QUARTERLY

THE JOURNAL OF COMMUNICATIONS TECHNOLOGY

Winter 1996

$9.95



- Optical Communications
- Oscillators with Low Phase Noise and Power Consumption
- Try NMR with Your Old CW Rig
- NEC-WIN BASIC for Windows—A Review
- *Ulysses* Verifies the Shape of the Interplanetary Magnetic Field
- Simple and Inexpensive High-Efficiency Power Amp
- The MRF-255 RF Power Field-Effect Transistor
- Fractal Loops and the Small Loop Approximation
- Factors in HF-ARQ System Throughput
- Instruments for Antenna Development and Maintenance: Part 4

ISSN 1053-9433