**Milt Cram, W8NUE and George Heron, N2APB**

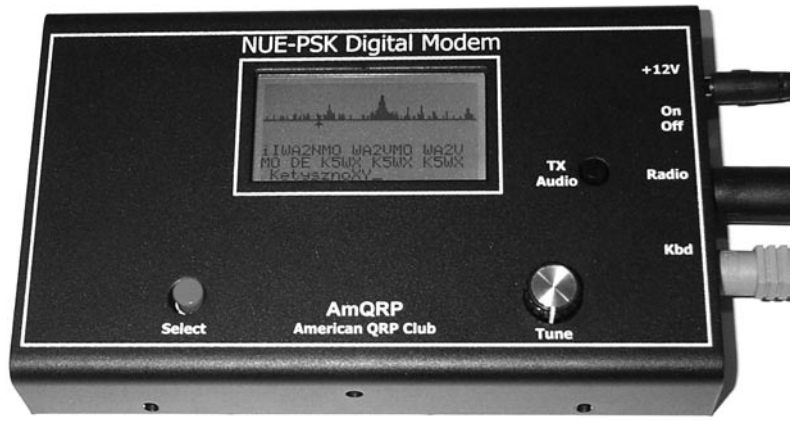
9807 Vista View Dr, Austin, TX 78750, w8nue@arrl.net • 2419 Feathermae Ct, Forest Hill, MD 21050, n2apb@amsat.org

# NUE-PSK Digital Modem

*Enables PSK31 field operation… without using a PC!*

PSK31 is one of the latest communications modes to capture the interest of hams worldwide. Its inherent ability to dig out low, nearly inaudible signals is ideally suited for low power, QRP, enthusiasts. The PSK31 digital modem engine, however, requires intense digital signal processing (DSP) that is only commonly available in PC sound cards. Thus, the PSK operator desiring portability for field operation is locked into using a laptop computer as a controller, which results in a cumbersome station. But there's hope!

This article presents the design and construction of a stand-alone, battery-operated digital modem using a Microchip dsPIC microcontroller. The project includes a graphic display for transmit and receive text data, as well as a band spectrum and tuning indicator. Using GPL open source software, the modem can be homebrewed for less than $80. When coupled with an SSB-capable transceiver or with a popular PSK-xx transceiver board from Small Wonder Labs, you too can have an effective portable PSK31 station.




## Background

PSK31 was introduced in 1998 to the ham technical community at large in RSGB's *RadCom* magazine.[1] Hams could get on the air with this digital mode using a dedicated (expensive) DSP card, a crude DOS control program for entering/displaying messages, and interface cables for connection to the station SSB transceiver. Later, a brilliant PC program was developed (*DigiPan*) that used a panoramic graphical display to show all signals occurring within a band segment, and print received messages on the PC screen.[2] This was an astonishing improvement in the user interface for PSK31! Later in 2001, Dave Benson, K1SWL, designed a single board PSK31 transceiver kit (PSK-20) that required no physical tuning, and when used with *DigiPan* running on a PC, it made a quite compact PSK31 station.[3]

Even with these clever hardware and software designs, however, there *still* was room for improvement. The sound card in a laptop or PC is still needed for the intense demodulation requirements of the PSK algorithm. If you were to use a modern laptop for that computing power, taking an expensive and delicate computer into the field is a hair-raising experience. It is difficult to see the subtle spectral lines or the screen text data when viewing a laptop LCD display in the bright sunlight of a mountaintop QSO. Then, only *if* your laptop battery lasts long enough to enjoy the fun of operating PSK out in the open, and *if* you can see the laptop display in the bright sunlight, and *if* you feel like lugging that expensive laptop out into the harsh elements, you could indeed operate PSK31 in the field — but what an ordeal!

## PSK MODULATION-DEMODULATION OVERVIEW

We will not go into great depth concerning the theory and operation of PSK. In this paper we'll first overview the PSK31 encoding scheme, followed by the more demanding decoding scheme.

Note that while the NUE-PSK project focuses on the generation and decoding of PSK31, it is generally known that PSK31 is merely one of many modulation techniques within the "phase shift keying" family of communication techniques. PSK31 operates at 31.25 bits/second, while other speeds may be achieved using slight algorithmic variations. PSK is perhaps more accurately termed BPSK, for bi-phase shift keying, whereby two distinct phase states separated by 180° are used to convey the information. Four states may also be encoded/decoded, as is done with QPSK (quad-phase shift keying), in order to provide higher speeds and the ability for better error correction methods.

We will primarily describe the topic of PSK31, yet understand that some of these other modes can also be achieved with the same hardware and software used in NUE-PSK.

### Modulation (PSK31 Encoding)

The PSK31 modulation algorithm is quite straightforward and could even be implemented on a conventional PIC-like device (one without a DSP core). This was done in several projects over the years within the QRP community; see, for example, the PSK31 Beacon project from the NJQRP Club.

Summary of the encoding steps:

1) Varicode encoding of the input text character stream coming from the keyboard to create an optimized bit-representation of the text;

2) BPSK serialization of the Varicode character to create the proper sequence of phase changes in the waveform based on the bits in the Varicode;

3) Form the wave shape from the com-

[1]Notes appear on page 12.

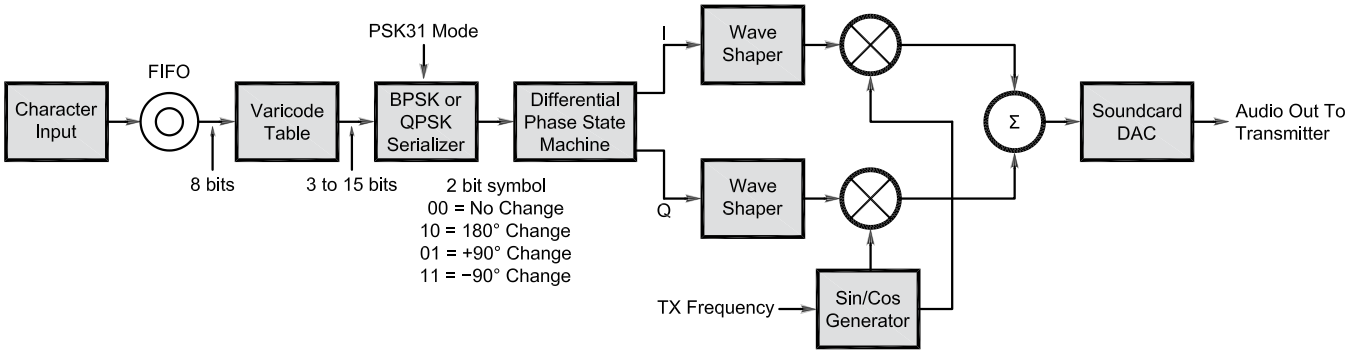



**Figure 1 — PSK modulation block diagram.**

bination of phase changes coming from the serializer, being careful to reduce the power level to zero when the 90° phase changes occur, thus reducing the bandwidth of the transmitted PSK signal.

These steps are all performed by a dsPIC processor, per the functional block diagram shown in Figure 1. As ASCII characters are produced by a keyboard, they are first converted to Varicode encoded characters using a lookup table. A string of binary bits, the length of which is variable (hence "Varicode"), is generated from the table. The strings of bits are then used to drive a differential phase state machine, which uses predefined tables to modulate the amplitude of the quadrature outputs (sine and cosine waveforms) of a numerically-controlled oscillator (NCO). The sine and cosine codes are derived from a lookup table to produce the NCO carrier.

The two quadrature oscillator signals are multiplied by amplitude functions, as determined by the phase state machine, and the resulting channels of data are added to produce a digital version of either a BPSK or QPSK signal. Although a simpler scheme could be used for BPSK alone, this method has the advantage that it can also generate QPSK. This digital stream of data is then sent to a digital to analog converter (DAC) to produce an audio carrier with BPSK/QPSK modulation. The output of the DAC is sent to the transceiver audio input for conversion to RF.

### Demodulation (PSK Decoding)

Whereas the encoding process described above is pretty straightforward, the PSK *decoding algorithm* is significantly more complex and computationally demanding. This may be why there have been so few homebrew standalone PSK demodulator projects in the ham community. The PC sound card is clearly the easiest way to provide the intense DSP processing needed for decoding PSK; hence PC-based PSK31 programs abound.

This is where the stand-alone (PC-less) NUE-PSK project excels — it is able to independently handle the complex PSK decoding algorithm in real time, thus providing the first truly portable digital modem for hobby use.

Follow Figure 2, the PSK demodulation block diagram, as we walk through the decoding steps.

Summary of the decoding steps:

1) Receiver audio is sampled at 8 kHz, creating a digital floating point representation of the audio stream.

2) Data is fed into a 512 point Fast Fourier Transform (FFT) for display, tuning and visual signal monitoring purposes.

3) The audio floating point data stream is converted to a baseband signal centered on the operating frequency. The NCO generates sine and cosine signals and multiplies them with the audio stream to produce I (in phase) and Q (quadrature phase) data streams.

4) The I and Q data streams are decimated by 16 to reduce the sample rate to 16 times the signal BW. The final sampling rate then is 8000 / 16 = 500 Hz. [In digital-signal-processing speak, to decimate a signal by some number, *n,* you keep every *nth* sample, throwing away all of the other samples.—*Ed.*]

5) A 65-tap "matched bit" finite impulse response (FIR) filter is applied to produce a magnitude response for best signal to noise ratio (SNR) for data extraction, and to minimize inter-symbol interference (ISI) presented in the signal path and in the receive filter.

6) AFC is performed to lock on the incoming signal frequency by using another FIR with BW = 31.25 Hz.

7) AGC is accomplished by computing the average signal magnitude from the I and Q data streams. Infinite impulse response (IIR) filters are selected to provide either slow decay or fast attack settings.

8) Frequency error detection is done by scanning the FFT data within the capture range while looking for the nearest peak. Also, a wide range AFC algorithm is performed by calculating the slope and moving the NCO to place the peak at the center.

9) Symbol synchronization is done by finding the center of each symbol in order to sample at the optimum time. There are 16 samples per symbol at 500 Hz intervals, so each sample energy is IIR-filtered and stored in an array. The array elements with the most energy are selected as the center of the data symbol at each symbol period of 32 ms.

10) Squelching is done by taking the histogram of incoming signals and considering the spread (difference angle, or arctangent of Q / I between each sample) around 0° and 180° as a measure of signal quality. The narrower the spread, the stronger and more coherent the signal.

11) Symbol decoding is the last step, whereby we convert the I and Q signals back to two possible symbols by using the difference angle (<90° = 1, >90° = 0). The resultant symbols are shifted into a register until the inter-character mark (2 or more zeros) is found. The shift register is then used as an index into a reverse Varicode table containing the originally transmitted characters.

These eleven algorithm steps can be followed in the block diagram of the demodulation process.

## The Path to a Design

After operating with the limitations of using a laptop in the field, we decided that we wanted a PSK station that did not require the use of a PC in any form. We wanted something that would be very portable and compatible with QRP operations, providing many hours of operation from batteries. The project described in this article is a result of this desire — but it took a little time for advancing technology to pave the road.

The initial efforts to develop a "portable PSK" controller began about eight years ago with a reproduction of the original G3PLX approach described in *RadCom*, but with a more current DSP card providing the horse-

power for the PSK31 engine. The design also included a novel Morse user interface and tight coupling to the PSK-20 transceiver. The project was documented in the QRP literature and was presented at ham conferences, but ultimately it was too complex and fragile for wide-scale use.[4] See Figure 3.

The next approach we considered was based on the use of low power DDS (direct digital synthesis) chips for generating audio tones with the proper phase modulation. A multiplying DAC was used for modulating and shaping the amplitude of the tones, and a microcontroller was used to demodulate the PSK and display the resulting characters. Analog filters were used for filtering the PSK signal ahead of digital processing in the microcontroller. The analog filters, however, proved to be too bulky and difficult to design when trying to use standard-value components. Such filters also cannot provide the same level of performance as can be obtained with digital filters. Eventually this approach was also abandoned.

**Table 1**
**NUE-PSK Digital Modem Parts List**

| *Designator* | *QTY* | *Description* | *Source* | *P/N* |
|---|---|---|---|---|
| C1, C2, C3, C7, C9, C11 C13, C17, C18, C19, C21 C22, C23, C24, C25 | 15 | Capacitor, 0.1 µF, 1206 SMT | Digi-Key | PCC1883CT-ND |
| C4, C5, C9, C10, C12, C17 | 6 | Capacitor, 1 µF, 16 V, SMT | Digi-Key | PCE3045CT-ND |
| C6 | 1 | Capacitor, 10 µF, 25 V, SMT | Digi-Key | PCE3118TR-ND |
| C15, C16 | 2 | Capacitor, 20 pF, 1206 SMT | Digi-Key | 311-1153-1-ND |
| D1, D2, D3 | 0 | Diode, Schottky 1N5817, DO-41 | Digi-Key | 1N5817DICT-ND |
| D4, D5 | 2 | Diode, Schottky MA2SE01, SMT | Digi-Key | MA2SE0100LCT-ND |
| ENC-1 | 1 | Rotary encoder | Mouser | 688-EC12E2420802 |
| J1 | 1 | Coaxial dc power connector, 2.1 mm | Mouser | 163-5004-E |
| J2 | 1 | 6-pin Mini-DIN | Mouser | 161-2206 |
| J3 | 1 | 8-pin Mini-DIN | Mouser | 161-2208 |
| J4 | 1 | Pinheader, female, 1 × 2 | Mouser | 517-870-01-03 |
| J5 | 1 | IC socket, 16-pin DIP | Mouser | 575-199316 |
| J6, J7 | 2 | 9 V battery clip | All Electronics | BST-3 |
| LCD | 1 | LCD, CFAG12864, 128 x 64, graphics | Crystalfontz | CFAG12864BTFHV |
| P1 | 1 | Pinheader, 1 × 2, 0.1” | Mouser | 517-834-01-36 |
| P3 | 1 | Pinheader, 2 × 3, 0.1” | Mouser | 517-834-01-36 |
| P4 | 1 | Pinheader, 1 × 4, 90° | Mouser | 517-5111TG |
| P5 | 1 | Pinheader, 1 × 2, 0.1”, 90° | Mouser | 517-5111TG |
| P8 | 1 | 8-pin Mini-DIN plug | Mouser | 171-2608 |
| PB1 | 1 | Pushbutton, DPST, momentary | New ark | 19C6398 |
| PB1-cap | 1 | Pushbutton cap | New ark | 18M6492 |
| Piezo | 1 | Piezo buzzer | Digi-Key | 433-1023-ND |
| Q1, Q2, Q3 | 3 | Transistor, NFET, 2N7000 | Digi-Key | 497-3110-ND |
| R1, R2, R9, R12 | 4 | Resistor, 1 kΩ, 1206 SMT | Digi-Key | RHM1.00KFCT-ND |
| R4 | 1 | Resistor, 10 kΩ, 1206 SMT, 1% | Mouser | 71-CRCW1206-10K |
| R7, R8, R10, R11 | 4 | Resistor, 10 kΩ, 1206 SMT | Digi-Key | 311-10KECT-ND |
| R13 | 1 | Mini-potentiometer, 1 kΩ | Mouser | 317-2080F-1K |
| R3 | 1 | Resistor, 47 Ω,1/2 W axial | Mouser | 293-47-RC |
| R14 | 1 | Trim pot, 10 kΩ | Mouser | 652-3306W-1-103 |
| R15, R16 | 2 | Resistor, 6.8 kΩ, 1206 SMT | Digi-Key | 311-6.8KECT-ND |
| R5 | 1 | Resistor, 2.0 kΩ, 1206 SMT, 1% | Mouser | 71-CRCW1206-2K |
| S1 | 1 | Switch, SPDT, slide, PCB mount, 90° | Digi-Key | EG1917-ND |
| SH-1 | 1 | Pinheader, 1 × 2 shunt | Mouser | 517-951-00 |
| U1 | 1 | IC, Microchip DSC, 64-pin QFP, dsPIC33FJ128MC706 | Mouser | 579-33FJ128MC706IPT |
| U2, U3 | 2 | IC, Octal Level Shifting Buffer, TXB0108 (TSSOP-20) | Mouser | 595-TB0108PWR |
| U4 | 1 | IC, Microchip EEPROM, 24AA256 (8SOIC) | Digi-Key | 24AA256-I/SN-ND |
| U5 | 1 | IC, Freescale microcontroller, MC68HC908QY4, 16-DIP | Digi-Key | MC68HC908QY4VPE-ND |
| U6 | 1 | IC, Dual-DAC, MCP4922, 14SOIC | Digi-Key | MCP4922-E/SL-ND |
| U7 | 1 | IC, Programmable Gain Amplifier, MCP6S21, 8SOIC | Digi-Key | MCP6S21-I/SN-ND |
| U8 | 1 | IC, Op Amp, MCP601, 8SOIC | Digi-Key | MCP601-I/SN-ND |
| U9 | 1 | Voltage regulator, 5 V switching, PT78ST105H, 5 V | Digi-Key | PT78ST105H-ND |
| U10 | 1 | Voltage regulator, 3.3 V, LP2950 (TO-92) | Digi-Key | LP2950CZ-3.3-ND |
| X1 | 1 | Crystal, 10 MHz, 20 pF (FOXSLF/100-20) | Digi-Key | 631-1101-ND |
| W1 | 1 | Flex cable assembly, 1 × 20 | Newark | FSN-21A-20 |
| | 1 | Cable assembly, 3-wire (battery clips) | | |
| Hardware | | | | |
| | 8 | Machine screw, pan slotted, #2-56 × 0.25” | Mouser | 5721-440-1/4-SS |
| | 8 | Machine screw, pan slotted, #4-40 × 0.25” | Mouser | 5721-256-1/4-SS |
| | 4 | Spacer, hex tapped, #2, 0.375” (LCD) | | |
| | 4 | Spacer, nylon, hex tapped, 4-40 × 0.25” (PCB) | Mouser | 561-L4.25 |
| | 1 | Knob | Mouser | 506-PKG50B1/4 |

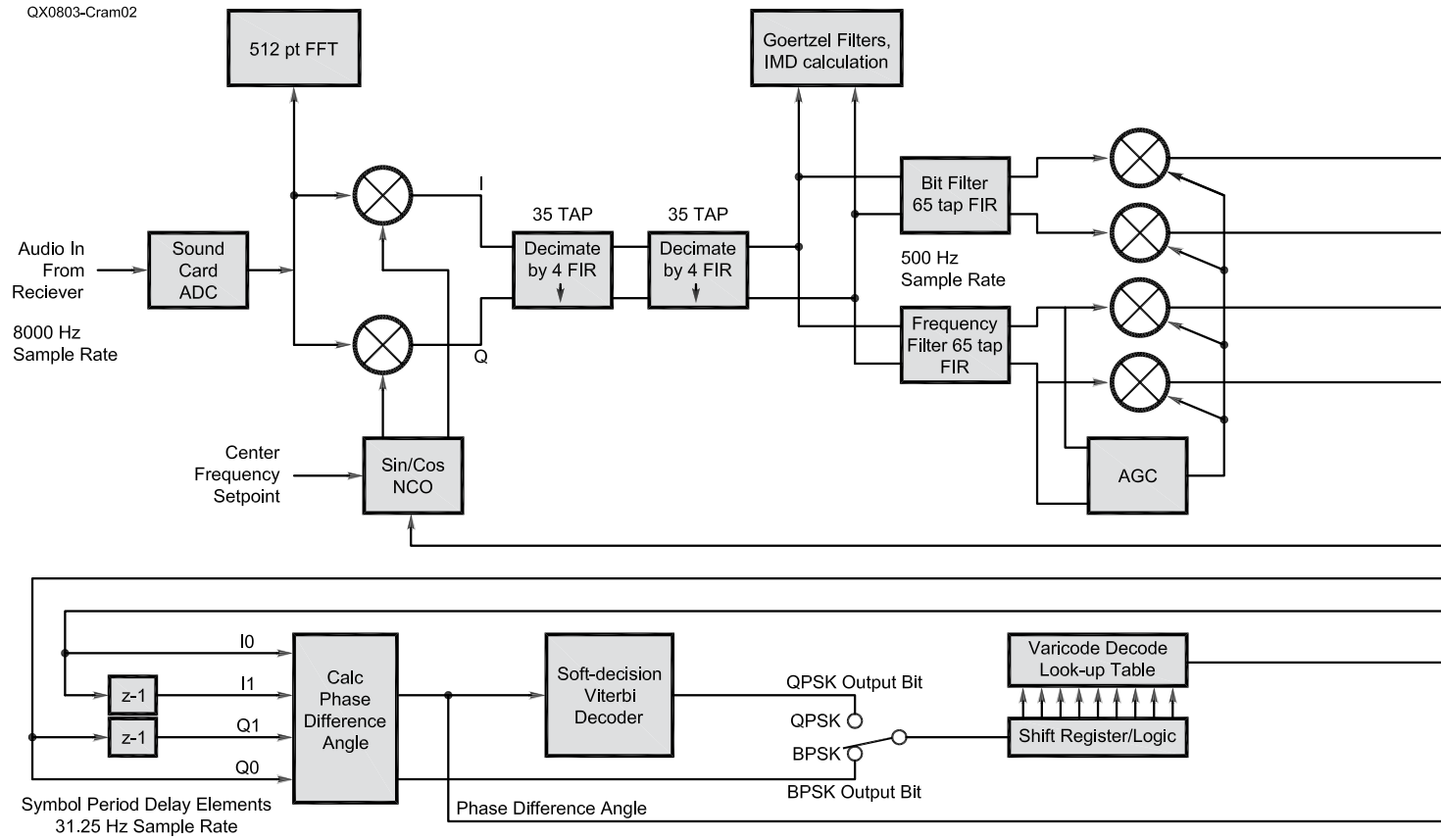



**Figure 2 — PSK demodulation block diagram.**

### Success At Last

The approach that ultimately proved workable in every regard was one in which all processing is accomplished within a single microcontroller — one that is capable of performing the digital signal processing "number crunching" as well as handling all control chores. The newly-released dsPIC33 microcontroller from Microchip is a delightfully powerful combination of a conventional control processor with a DSP core for intense digital signal processing.[5] Available in a small package with lots of I/O for controlling peripherals, this was just what the doctor ordered.

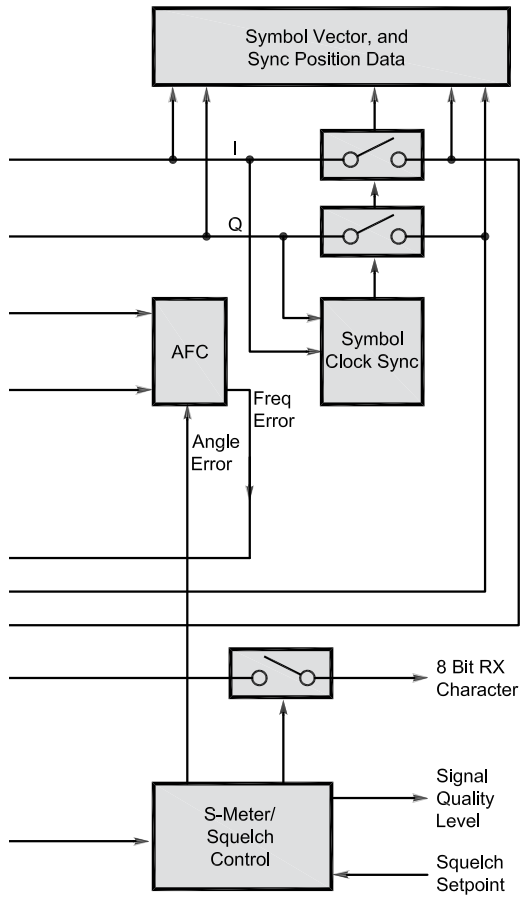
It was perhaps fortuitous that others in our QRP clubs were having similar fantasies at about the same time. K5JHF was exploring the dsPIC chip family and decided they would make a good basis for a number of projects of interest to the group. He kick-started things with the design of a dsPIC33 project board, including such peripherals as a programmable gain amplifier (PGA), digital to analog converter (DAC), EEPROM memory, liquid crystal display (LCD), a quadrature rotary encoder and interfaces for a programmer and a keyboard. This was enough to give birth to what we now call the NUE-PSK digital modem.

## NUE-PSK Hardware Overview

As illustrated in the schematic diagram of Figure 4, U1 — a dsPIC33F is at the heart of the project design. This highly-integrated dsPIC33F device employs a powerful 16-bit architecture that seamlessly integrates the control features of a Microcontroller (MCU) with the computational capabilities of a DSP IC. The resulting functionality is ideal for applications that rely on high-speed, repetitive computations, as well as control — just perfect for our PSK31 digital modem project! Table 1 is the complete parts list for the NUE-PSK modem.

**Figure 3 — We built this portable PSK unit around 2000. It was too complex and expensive, with separate boards for DSP and control processing. It did include a novel CW user interface.**

The dsPIC33F central processing unit (CPU) has extensive mathematical processing capability with its DSP engine, dual 40-bit accumulators, hardware support for division operations, barrel shifter, 17 × 17 multiplier, large array of 16-bit working registers and a wide variety of data addressing modes. Flexible and deterministic interrupt handling, coupled with a powerful array of peripherals, renders the dsPIC33F devices suitable for control applications. Reliable, field programmable flash program memory ensures scalability of applications that use the dsPIC33F family of devices. The specific device we used contains 128 KB of program flash memory, 16 KB of RAM, nine 16-bit timers, 16 general-purpose I/O pins, a pulse




## Buying or Building Your Own NUE-PSK

Assembled and tested NUE-PSK modems can be purchased from the American QRP Club at **www.amqrp.org/kits/nue-psk31/**. The cost is $199 for US and Canadian shipment; $219 for overseas orders. Accessories are also available. You can order online, or send a check or money order payable to the American QRP Club c/o George Heron, 2419 Feather Mae Ct, Forest Hill, MD 21050. Full and partial kit versions will be available later this year. Check the American QRP Club Web page for the latest updates.

If you prefer to source your own parts and build from scratch, see Figure 1. The NUE-PSK software is available for free downloading on the NUE-PSK Web page.

Whether you decide to homebrew the modem, or perhaps get the partial kit and assemble it yourself, don't be afraid of soldering the surface mount ICs used in this project. Here's a technique that works great even for the 64-pin dsPIC chip. Using a magnifying lamp, position the IC on the pads and tack solder two corner leads to hold the package in place. Liberally solder all the leads to the pads without any concern for shorts between the leads. Next, use some desoldering braid (like SolderWick) to remove all excess solder along the rows of leads. Don't worry about overheating the IC package — it's tough. After all that excess solder is sucked up, you're left with the cleanest looking connections that could ever be achieved by hand soldering!

width modulation port, a port designed for reading quadrature encoders, two 16-channel ADCs, two UARTS, two SPI ports, two I2C ports, and comes in a 64-pin quad surface mount flat pack package. Whew, this sure is a powerful chip.

The initial prototype used the dsPIC to capture and decode signals from the PS2 Keyboard. This worked fine, except that on rare occasions, the dsPIC appeared to reset itself. This had the unfortunate effect of losing current operating information such as the frequency, call sign, and other. After reviewing all information regarding the PS2 keyboard, we didn't like the way we were capturing scan codes from the keyboard. Data was being sent synchronously from the keyboard to the dsPIC, using a clock of only roughly known frequency (~10-20 kHz). Each clock pulse caused an interrupt in the dsPIC, which then sampled the data stream. With the keyboard protocol, selected by IBM many years ago, each scan code is sent using 11 clock pulses. In addition, each keystroke press and release results in three or more scan codes being generated. Consequently, each keystroke generated a minimum of 33 interrupts.

Apparently too much time was being wasted just processing keyboard interrupts, and that was the likely cause for the occasional dsPIC resets. To solve this problem, we decided to use another small microcontroller to do most of the work handling the keyboard data. This second microcontroller, U5 (Freescale 68MC908QY4) simply responds to the clock from the keyboard and gathers the bits received into a complete scan code (11 interrupts). Once a scan code is completed, the 'QY4 generates a strobe pulse to the dsPIC. Again, an interrupt in the dsPIC causes the dsPIC to capture an entire scan code on a set of port pins, and place it in a buffer, or merely sets a flag if the scan code is not a usable character. The ultimate effect of this division of responsibilities is that the dsPIC now responds to only 1/11th of the number of keyboard interrupts that were present in the first attempt.

Two LCD displays were initially chosen for the PSK interface. A character LCD was used for displaying received decoded text and as a monitor for text being placed in the transmit queue. Text is displayed when in transmit and as macros are being played back. The cursor changed from steady to flashing when in transmit. Setup Menus were also displayed on the text display. A 144 × 32 pixel graphics LCD was then used to display the FFT-generated spectrum of the audio passband. The lowest six rows of the display were used for the tuning cursor. Since a 512-point FFT is used with an 8 kHz sampling rate, we have 256 points for a 4 kHz passband. We chose to display only the frequency range from 500 Hz to 2500 Hz, using 128 columns of the display. (The last 16 of the 144 horizontal pixels in each row were not used.) The data and control lines for each display were buffered by level translators U2 and U3, required to match the voltage levels between the 3.3 V dsPIC and the 5 V displays.

Since our original prototypes were built, we decided that we could possibly save some cost and simplify packaging by using a single graphics display for both text and spectral display. A 128 × 64 pixel display was chosen. The display drivers were combined into one, and modified to handle display of text buffers and an FFT of the input signal (spectral display), along with a "cursor" for tuning. Text is displayed on the bottom half of the display, using 5 × 8 pixel characters with 4 lines of display. The top 32 pixels are used for the spectral display, and the tuning cursor. In addition, the display incorporates a backlight that can be turned on or off by means of either a hot key or from a menu selection. FET transistor Q2 buffers the control line going to the backlight pin on the LCD.

The EEPROM, U4 (24AA256), provides local storage for the macro and user-set variables entered during modem operation. This memory device is controlled by one of the I2C ports on the dsPIC.

A computer-adjustable gain stage, the programmable gain amplifier (U7, MCP6S21), brings the low level audio input stream coming from the SSB receiver to the analog-to-digital converter on the MCU. Amplifier U8 (MCP601) presents precisely one-half the Vdd voltage to the ac reference input of U7.

Processed digital transmit audio tones are converted to a continuous analog stream by D-to-A converter U6 (MCP4922). The audio level control R4 sets the appropriate modulation level to the input of the SSB transmitter, which is generally a one-time setting for the transmitter being used. To produce a bipolar ac signal, a numeric constant equal to one

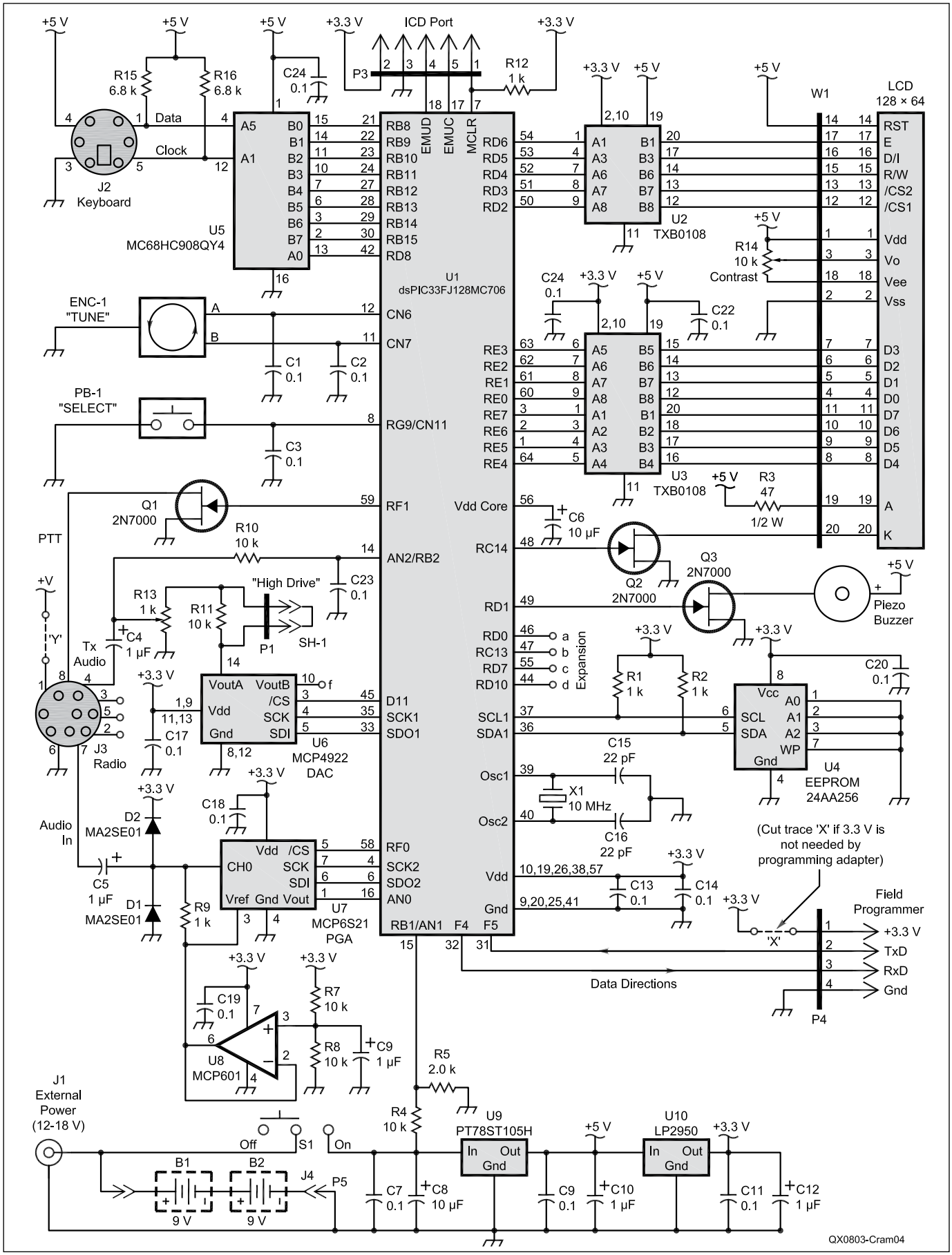



**Figure 4 — The NUE-PSK schematic diagram.**

half of the full scale output is added to the data stream generated by the dsPIC. Since the output is capacitively coupled, the dc term represented by the half scale constant is removed. The full analog signal is presented to the audio level control, however, and one of the dsPIC ADC inputs is used to measure the dc voltage on the wiper of the level control. This allows the dsPIC to determine the position of the wiper and display that information on the LCD, as desired (a menu option). This facilitates setup with different rigs, once the correct setting is determined for each rig. The wiper of the control is ac coupled to the rig audio input.

FET transistor Q1 (2N7000) buffers the push-to-talk (PTT) control line sent to the transceiver, used to put the radio into transmit mode.

A piezo buzzer is provided to deliver audible feedback for Tuning, menu selection and for future features. FET transistor Q3 buffers the control line to the buzzer.

The audio input, output and PTT control lines are brought off the pc board using an 8-pin mini-DIN connector, J3. This approach minimizes the number of connectors and cables normally used to connect a digital mode controller to an HF rig, as sometimes these cables can get mixed up and messy at the operating station. Further, when the NUE-PSK modem is used with a dedicated HF rig – say a Yaesu FT-817/857/897 or a PSK-20 transceiver card — the other end of the cable may also be consolidated to a single multi-pin plug, providing a neat and elegant interconnect with the radio.

For the design of the power supply, we chose to use a switching regulator (U9) instead of the more conventional 7805 linear regulator to get 5 V on the board. This solution requires a lower operating current from the supply because of the greater efficiency achieved by the switching "buck" regulator. A linear regulator merely dissipates the power difference between input and output in the form of heat. Thus, even though the dsPIC draws approximately 100 mA, the modem now only requires about 60 mA from the supply during normal operation, and portable power is easily provided by conventional alkaline batteries. Figure 7 shows the current requirement as a function of supply voltage.

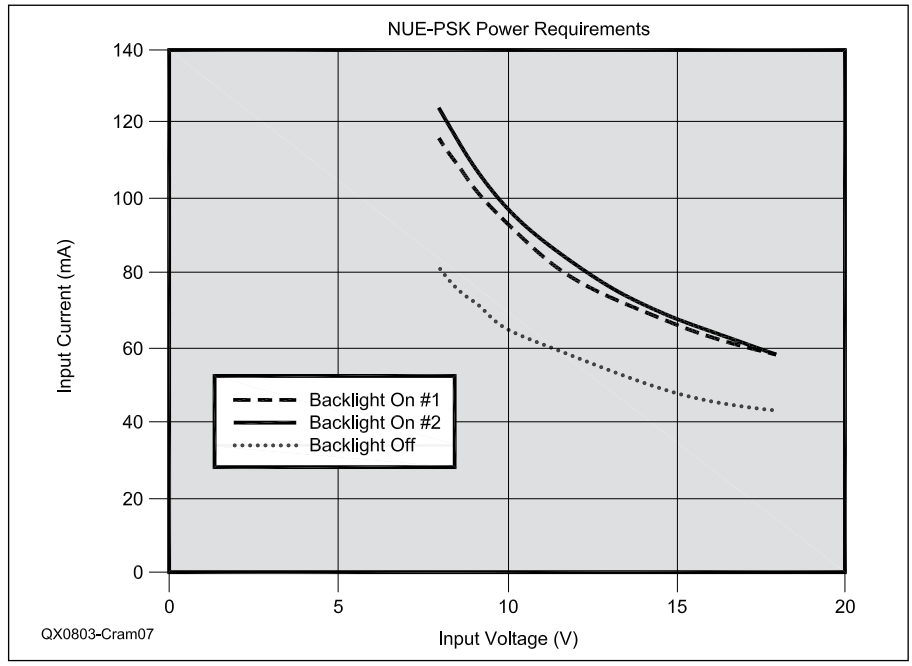



**Figure 7 — Power requirements for the NUE-PSK modem. Measurements illustrate the dramatic benefits of using the switching "buck" regulator. Regulator efficiency increases as higher supply voltages are used. The top curves show the input current requirement when running with the display backlight on, while the lower curve shows 15 mA less current when the backlight is off.**


**Figure 5 — The two-LCD Prototype used a graphic LCD for the spectrum display (top) and a character LCD for receive and transmit text characters (bottom).**

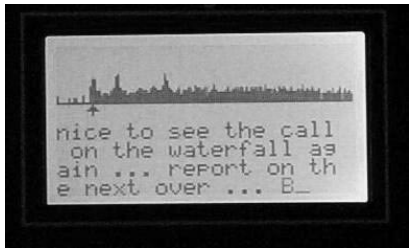



**Figure 6 — The newer single graphic LCD shows both spectrum and receive or transmit characters. The backlight affords great night time visibility and costs only 20 mA in additional current demand.**

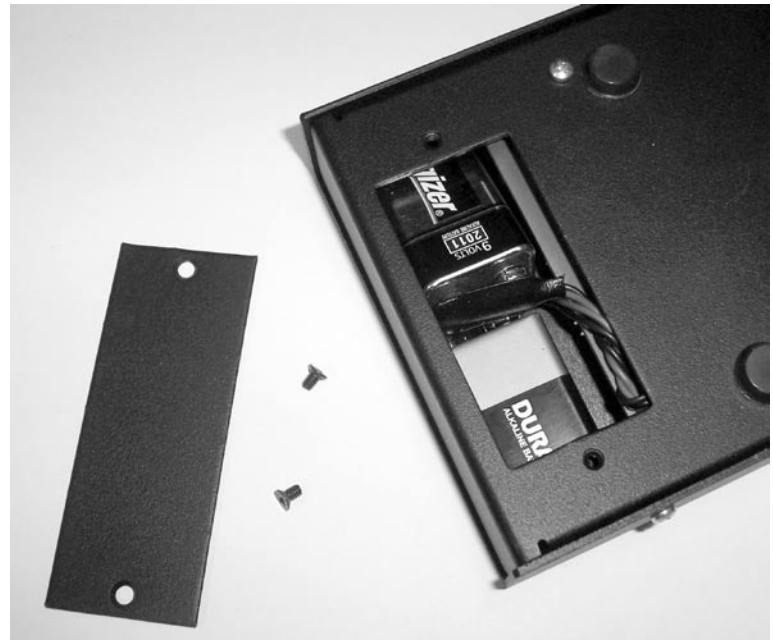


**Figure 8 — The two 9 V alkaline batteries nestle tightly against the circuit board in the case compartment. When installed, the screw-on cover holds them firmly in place.**

A small drawback of using the switching regulator is that a 9 V minimum input is required to maintain regulation; so battery operation is achieved by using two standard 9 V batteries in series to provide a nominal 18 V input to the modem. See Figure 8. Of course the digital modem may instead be externally powered by applying 12 V through J1. When external power is applied, the internal battery connector should be disconnected, or the batteries should be removed.

The NUE-PSK project is assembled using a single 4 × 5 inch pc board — quite an improvement over the Portable PSK projects done previously, as well as over the prototype hardware for this current design. The pc board holds all components — the LCD, rotary encoder, power connector and radio interface connectors — and may be assembled into your favorite homebrewed enclosure, or in the clam shell aluminum enclosure made available when the kit is purchased from the American QRP club.[6] This enclosure also has a conveniently-accessed compartment on the back side that houses the 9 V batteries. See Figure 9.

**Figure 9 — This photo shows the NUE-PSK assembly. A 4 × 5 inch circuit board fits neatly into the enclosure, holding all components. (Individual wires are shown connecting the display in this prototype unit.) The battery "door" in the back of the case is visible along the left edge of the photo. Two 9 V batteries fit into the space between the circuit board and the case.**

### Hardware Evolution

Before ending up with a neat and compact circuit board, the NUE-PSK design started out as a rather large and sprawling prototype hardware layout. This is normally the case with complex projects, because it allows the designers to try out different approaches and components, while also allowing them to easily monitor and debug the design.

The prototype design was built using a proto-board purchased at Fry's Electronics. It has plated-through holes on 0.1 inch centers to facilitate mounting through-hole components. The surface mount dsPIC microcontroller is mounted on a "SchmartBoard," also obtained from Fry's.[7] This particular board is designed to permit attaching 32 to 100 pin SMT devices, and has 0.65 mm lead separation (pitch). Schmart Boards are available in several pitches and pin count configurations to accommodate prototyping of a range of SMT controllers. Header pins and sockets are used to connect the board to the main prototype board. Point-to-point circuit connections were accomplished using 30-gauge Kynar wire, and a hand-stripping tool was used to strip the ends prior to soldering to the socket/connector pins. Thus the prototype was rather easily assembled and the result was relatively solid when complete.

## Development Tools and Getting Started in Software

While Microchip is well-known in the ham community, few of us had experience using this new family of PIC chips. Microchip apparently foresaw this situation and they have provided an amazing number of application notes, specifications and guidance for designers to use in quickly coming up to speed.

Further, even the best chip on earth would be crippled without a good set of software development tools; but Microchip again came to the rescue with a C compiler and an extensive DSP library that proved invaluable to us in developing the project. Both of these were available for free, so what more could we ask!

To program the dsPIC, we discovered that the inexpensive (~$25) PICkit2 programmer from Microchip is entirely adequate for the job. In-circuit debugging is not readily achieved with the free versions of the tools, but we seemed to do alright regardless.

The final essential aspect in enabling this project was a design reference for the PSK31 modem algorithm, provided by Moe Wheatley, AE4JY. His *PSKcore* documentation and C++ source code was professionally done and placed into the public domain, so it was available for our use.[8] We concluded that it would be a straightforward conversion to C language so we could use our free compiler and have it work on the dsPIC33, and we relied heavily on it.

### Software Overview

Although Wheatley's code was written in C++, and was developed for use on a PC, it was not too difficult to convert it for compilation under C, for which there is a free compiler from Microchip. As part of our QRP group project, John Fisher, K5JHF, provided much of the initial software for the project. His code includes most of the initialization code, a keyboard handler, a basic LCD driver, I2C and SPI drivers, an interface for EEPROM storage, and ADC and DAC interfaces. Milt, W8NUE, developed the remaining code fairly easily, even though his programming experience has been mostly in BASIC and Visual Basic, with some FORTRAN.

### PSK31 Decoder Processing

The receiver audio from an SSB transceiver is supplied to the NUE-PSK circuits. Before processing by the dsPIC, it is passed to the PGA, whose gain is controlled by the dsPIC via a serial peripheral interface (SPI) connection. The output of the PGA is then sampled by an internal 12-bit ADC on the dsPIC.

Timer 1 of the dsPIC provides all of the critical timing. The timer is set for interrupts every 125 microseconds, corresponding to a sample rate of 8000 samples per second. In receive (demodulation), ADC samples are captured into a 2048 word buffer. Once the buffer is half full, a flag is set to inform the system that data is available for processing. Only half of the buffer is processed at a time. This ping-pong buffering technique allows continuous data processing to be accomplished while the other half is being filled in real time.

The "main" routine of the program is an endless loop in which a number of flags are tested and, if found to be set when queried, they are used to trigger execution of various functions. For example, if the ProcPSK flag is checked and found to be set, a block of data is then processed. Each sample in the buffer is multiplied by a quadrature NCO, producing I and Q signals. Each of these is then passed two times through 35-tap decimate-by-4 FIR filters. This provides I and Q signals that are now sampled at 500 samples per second. (If in PSK63 mode, the second filter bank will decimate-by-2, providing 1000 samples per second.) While the block of 1024 samples is being processed, the second half of the buffer is being filled with new samples under control of the Timer 1 interrupts. Processing then ping-pongs between the two halves of the buffer. Using this technique we never

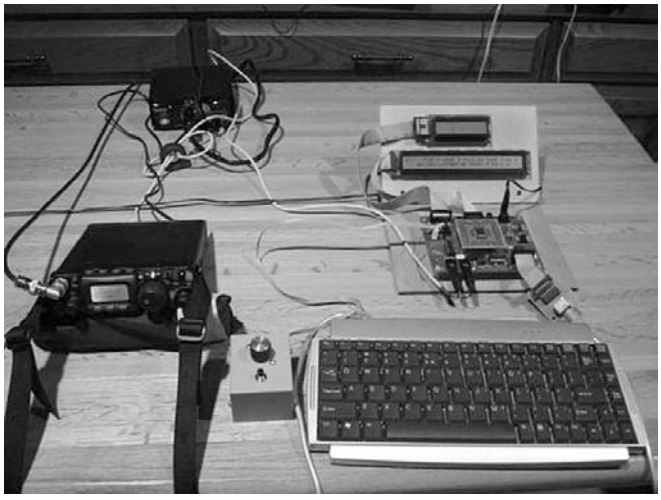


**Figure 10 — NUE-PSK Prototype System. Clockwise from upper right: NUE-PSK displays and prototype hardware, standard PS2 keyboard, FT-817 transceiver, and power supply.**

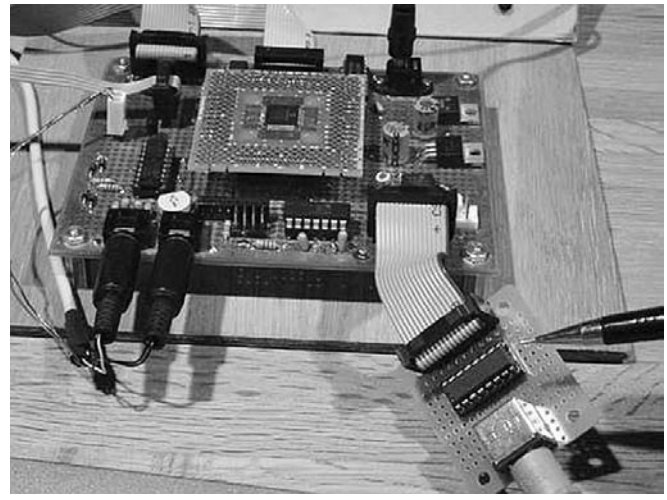


**Figure 11 — This close-up of the NUE-PSK prototype shows the multiple cabling and programmer connection (lower right), which allows convenient access to the electronics during design shake down.**

write new data to the part of the buffer that is being processed.

The next step is to split each of the I and Q channels into two paths. One is for the processing of the bits and one path is for processing of frequency data, producing four channels of data. Each of these channels is filtered by a 65-tap FIR. The I and Q bit channels should be optimized to minimize intersymbol interference, while the I and Q frequency channels should be optimized for fast response of the automatic frequency control (AFC) loop. All of the FIR filters have responses as specified by AE4JY. Instead of using the *PSKcore* filtering code, we are using FIR filters from the Microchip DSP library, as these software filters are designed to take into account the special hardware features of the dsPIC. The results can be shown to be the same, however. That is, they have identical frequency responses.

The bit channels are processed as described in the *PSKcore* specification to determine the proper time for determination of the phase changes that are employed in PSK. Since the bit rate of PSK31 is 31.25 Hz, each bit extends for 32 milliseconds in time. We have a sample rate of 500 Hz at this stage of processing, so there are 16 samples for each bit. The point in time for proper synchronization of the phase detection process is based on an analysis of the average energy in each of the 16 samples when averaged over several bits. Without going into the mathematical details, suffice it to say that the maximum energy always occurs at the boundary between successive bits. This fact is used to establish synchronization in the bit detector.

We used the free *WinFIRDesigner* software, with parameters obtained from the AE4JY code to calculate the FIR filter coefficients. As noted above, the frequency responses obtained with these coefficients are identical to those published by AE4JY.

A processing block takes the four filtered signals, and proceeds to:

1) obtain a digital AGC control;

2) calculate frequency errors;

3) correct the numerically controlled local oscillator;

4) determine bit boundaries;

5) determine whether a 1 or a 0 is being received;

6) collect the decoded 1s and 0s into a Varicode pattern;

7) convert the Varicode pattern into ASCII characters; and finally

8) display the resulting characters.

The *PSKcore* routines were used to perform AGC, bit synchronization, character decoding, and so on. In addition, we added code that will perform a 512 point FFT on the samples (8 kHz sampling rate) that are provided to the FIR filters. The processed FFT is then converted to magnitude, and then to a logarithmic scale. The 500-to-2500 Hz portion of the spectrum is displayed on the upper half of a $128 \times 64$ pixel graphics LCD. This display is essential for tuning. More about this later.

The final demodulator processing output is a decoded ASCII character. These decoded characters are displayed on the lower half of the $128 \times 64$ graphics display, as four lines of 20 characters each. The display driver includes a line buffer so that once a line of characters is filled, it is scrolled up and new characters are inserted at the beginning of the second line. This approach was chosen so that printed characters remain fixed for easy reading, as opposed to all characters being in constant motion (scrolled horizontally) once a line is filled.

### PSK31 Encoder Processing

As mentioned earlier, the encoding process is considerably less-intense as compared to the decoder operations. ASCII characters are accepted from the keyboard, converted to Varicode characters, and the binary string represented by the Varicode is used to modulate the phase and amplitude of an audio carrier — the PSK audio signal.

Although *PSKcore* code creates a block of data to be sent to the PC soundcard, we chose to generate a single sample of output signal for each and every 125 microsecond timer interrupt. This minimizes data memory requirements. The method of generating the desired phase and amplitude modulation is that developed by AE4JY with the exception that the tables used reside in program memory instead of data memory. The use of these tables eliminates the time-consuming calculation of sine and cosine signal components. The choice of placing these tables in program memory was made because we had plenty of program memory with the dsPIC, but not a lot of spare data memory. The calculated data samples are then scaled for output to a 12-bit DAC. The DAC output, after capacitive coupling, is then routed to the audio input of an SSB transceiver.

As each interrupt occurs, the code steps through the tables, providing modulation values for the I and Q signals, resulting in either BPSK or QPSK modulation. The modulated I and Q signals are added together prior to the DAC.

## Using the NUE-PSK Digital Modem

Install two standard 9 V alkaline batteries in the battery compartment, or connect a 9 to 18 V dc supply to the coaxial power connector (2.1 mm) on the right end of the modem.

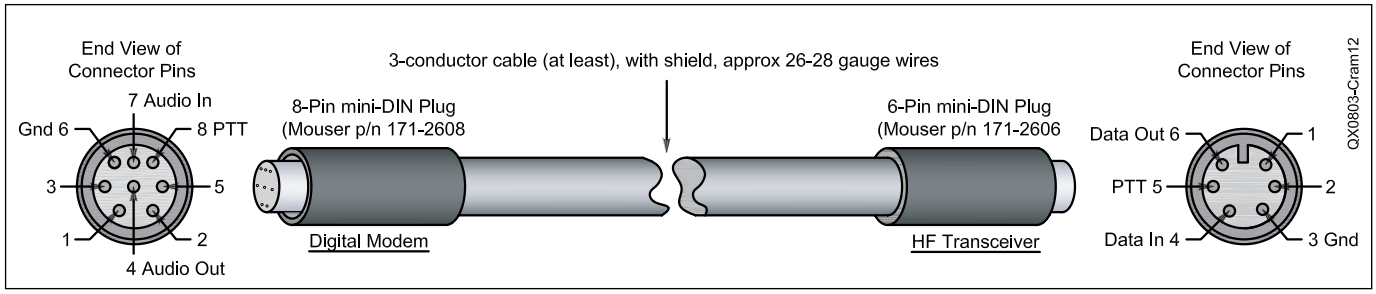



**Figure 12 — Connections between NUE-PSK digital modem and a typical HF transceiver. (The wiring diagram shows the connections for a Yaesu FT-817 radio.)**

### Signal Connections

Install a connector, or connectors, to the end of the cable that has an 8-pin mini-DIN connector. Most modern HF rigs have a mini-DIN Data or AUX connector, which provides for PTT, fixed level audio from the receiver (independent of the volume control on the rig), and a line-level (approx 100 mV RMS) audio input to the transmitter. On the Yaesu FT-817/857/897 this connector is a 6-pin mini-DIN. On many Kenwood HF rigs there are 6-pin and 13 pin mini-DIN connectors that may be used. See Figure 12 for wiring details.

### Keyboard

The modem requires an AT/PS2 style keyboard for character entry. The keyboard also provides for entry and playback of macros. Use the 6-pin mini-DIN connector on the end of the modem to connect to the keyboard.

### Operation

Once you have the cable between the modem and the rig connected, keyboard attached, and power available, you are ready to operate PSK. But first, some additional setup may also be desired, as described next.

Turn on the modem. If the cable between the rig and modem is wired correctly, you should see evidence of signals and/or noise on the top half of the display (the spectrum area). Tune your rig to one of the PSK sub-bands. These are typically 70 to 74 kHz above the lower band edge on 40 and 20 meters. If there is PSK activity on the band, you should see peaks on the graphic display. The horizontal location of the peaks corresponds to the audio frequency of each signal relative to the tuned frequency of the rig. For example, if the rig is tuned to 14070 kHz, the display shows audio frequencies from 500 Hz to 2500 Hz, or actual RF frequencies from 14070.5 to 14072.5 kHz.

Now for the fun — tuning! Turn the encoder clockwise, or counterclockwise, to move the cursor to a higher, or lower frequency. (The cursor is the small triangular icon just below the spectrum display.) The audio frequency is displayed when turning the encoder. Try to align the cursor with one of the peaks on the display. Don't worry if it is not exactly aligned. Once close to the peak, stop turning the encoder. The modem now attempts to "lock" onto the signal and fine-tune the frequency if needed. If the modem is able to lock onto a PSK signal, it will very shortly begin decoding the signal, and then display characters on the screen. The time it takes for decoded characters to appear depends on the ability of the modem to estimate the center frequency of the incoming signal, and the signal to noise ratio. Tuning can also be done with the arrow keys on the keyboard. The right and left arrow keys provide finer tuning, while the up and down arrow keys provide faster tuning. The tuning rate of the encoder on the modem can also be selected from a menu setting. Note: When tuning in receive mode, the spectral display is frozen—this is intentional.

Now, on to set-up for transmission. Connect your rig to a dummy load.

Since PSK signals generated by the modem contain simultaneous multiple frequencies (over a very narrow bandwidth), it is imperative that the audio output from the modem not overdrive the input to the rig, or very poor signal quality will result. To facilitate setting the audio drive to the rig, a potentiometer on the modem may be used to adjust the level. In addition, the modem includes provision for "measuring" the position of the potentiometer, so that it can be easily reset to the same setting in the future. More on this later.

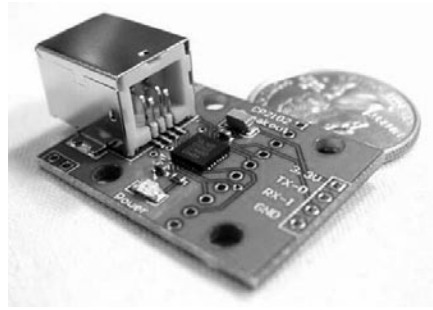


**Figure 13 — A USB-to-TTL interface adapter from SparkFun will allow your computer USB port to connect to the modem for programming updates to the software.**

We have found that the best way to set up for PSK operation is to initially set the transceiver for normal SSB operation, including whatever power setting you usually employ. For example, if you have a 100 W PEP rig, set it up for 100 W on SSB.

Switch to Digital mode (if your rig provides that option, otherwise retain the SSB mode).

Then press F8 on the keyboard. This places the modem in the TUNE state, which is denoted by "TUNE" at the top left of the display. The modem is now generating a continuous tone, which is fed to the audio input of the rig. The PTT signal from the modem should also cause the transceiver to switch to transmit. At this point, the potentiometer on the modem (just to the right of the display) can be adjusted to set the power level of the transceiver. A transmit power of 15 to 40% of the rig's rated power is recommended. (In other words, 15 to 40 W with a 100 W rig). Keeping the power at this level does two things. First, it minimizes distortion due to clipping. Second, it avoids excessive heating in the rig finals, since PSK is a 100% duty cycle mode. A power meter is very handy for making this setting. Once the potentiometer has been set, press F8 again to return to receive mode.

You should now be ready for transmitting PSK.

Pressing F12 will place the modem in transmit mode, but with a PSK idle tone being generated (unlike the CW tone in TUNE). If you are ready to give it a try, press F12. At this point, anything that you type on the keyboard will be converted into Varicode characters and transmitted using PSK modulation. Pressing F12 again, will toggle back to receive. When in transmit mode, "TX" will appear at the top left of the display.

### Macros

If you want to set up macros (pre-recorded strings of characters for subsequent playback) before proceeding, now is a good time to do it.

For those already familiar with PSK operations, macro setup is very similar to many of

the popular PSK programs. There are a few differences though. Some of the typing will be "blind" — not all of the input characters will be echoed to the display.

Before you begin to operate, you should record your call sign in the modem's EEPROM. While in receive mode, type your call sign and then press Ctrl+M.

Macro recording is initiated by pressing Ctrl plus the function key that you want to be associated with your macro. For example, to use F1 for calling CQ, press Ctrl + F1. Then begin typing " cq cq cq de." Now enter Alt+M, press the space bar, enter Alt+M again, press the space bar again, enter Alt+M again, press the space bar, enter "K" and finally enter Ctrl+Q. (Omit the quotes during the typing). Now press F9 to store the macro. When this macro is played during transmission, by pressing function key F1, it will call CQ three times followed by your call sign 3 times, followed by "K," and then the modem will revert to receive. In this procedure, entering Alt+M informs the modem that you want to insert your call sign into the transmit buffer. Entering Ctrl+Q, inserts a special character, which the modem recognizes as "quit transmitting and revert to receive." Each macro can contain up to 255 characters.

You can also record the "other station's" call sign in RAM (not in nonvolatile EEPROM) by pressing Ctrl+T after first typing their call sign on the keyboard. To insert the other station's call sign into a macro, simply use Alt+T in the macro. Then, when you play the macro, the other station's call sign will be inserted into the macro. This way, whenever you enter a new call sign using Ctrl+T, you do not need to re-record the macro to use the new call sign.

### Menus

Configuration of the modem is done through a menu system. For example, you can select between PSK, QPSK, and QPSK reversed. You can also change the software squelch setting, the gain of the programmable gain amplifier (PGA), turn CW Identification on or off, turn the display backlight on or off, change the tuning "increment," monitor battery voltage, or monitor the setting of the TX audio potentiometer. Other items may be added to the menu at a later time.

The menu system has two means of access. If you wish to access the menus using the keyboard, simply press F10 on the keyboard. Next enter a number on the keyboard corresponding to the submenu that you wish to access. Once this selection is made, choices for the submenu will be displayed. Another numeric entry will denote your selection. With the keyboard menu system, entering the submenu choice on the keyboard will cause an exit from the configuration menu.

The second method of menu access is through the "Select" button on the menu and the rotary encoder. Pressing and holding the Select button for more than ½ second will activate the menu system. When initially activated, the display will show "Configure" on one line, followed by "Exit" on the line below. If you wish to abort configuration, simply tap the Select button at this time. If, on the other hand, you wish to configure one of the modem settings, simply rotate the encoder clockwise, or counter clockwise, to cycle through the top level menu selection. Once you see an item that you wish to change, tap the Select button again. This will then allow you to cycle through a list of choices (again by rotating the encoder). When the choice you wish to make appears on the display, tap the Select button again. This will record your choice, and the menu will revert to the top level, showing "Exit" as the default choice. You can now make additional changes, or tap the Select button again to exit the Configuration menu.

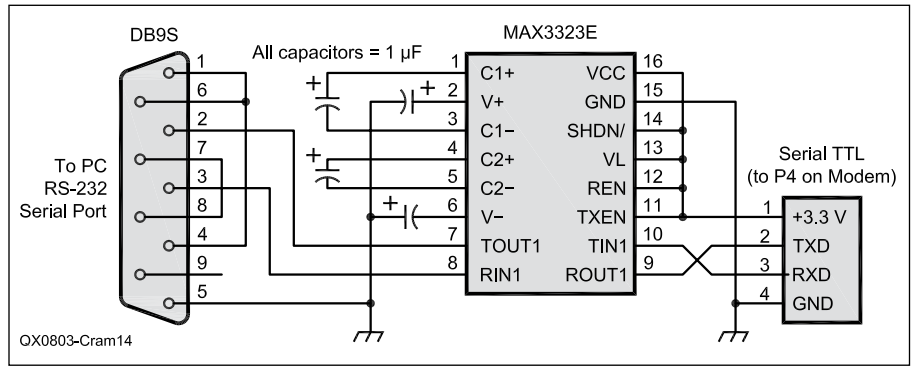



**Figure 14 — This schematic diagram shows an easy-to-build RS-232 interface that you can use between your computer serial port and the serial TTL input on the NUE-PSK modem.**

### Hot Keys

A number of "Hot Keys" have been defined for use with the modem:

**F1 to F7**: Play Macros.

**Ctrl-Fn:** Record Macros — Enter keystrokes. When finished, Press F9.

**Alt-Fn**: Delete Macro associated with Fn.

**F8**: Toggle TUNE mode. May be accessed only in RX or TX (Not in Setup, or Macro Recording).

**F11**: Display the first few bytes stored in EEPROM.

**F12**: Toggle between RX and TX (again, not in Setup or Macro Recording).

**F10**: Display the main Setup Screen. (Accessible only in RX mode).

**#**: A numeric selection from the Main Menu, selects a submenu, which is then displayed. Another numeric selection activates your selected parameter.

**Ctrl-M**: Save keyboard entries into a fixed location in EEPROM (for recording "my call sign," for use in Macros).

**Ctrl-T**: Save keyboard entries into a RAM location (for recording "their call sign" — also for use in Macros).

**Alt-M**: Insert "my call sign" into a Macro.

**Alt-T**: Insert "their call sign" into a Macro.

**Ctrl-F**: Save the current frequency into EEPROM so that it can be restored at the next power-up.

**Alt-F**: Retrieve the saved frequency and make it the current frequency.

**Ctrl-Tab**: Display the current (audio) frequency.

**Ctrl-A**: Enable AFC.

**Alt-A**: Disable AFC.

**PgUp**: Increase PGA gain.

**PgDn**: Decrease PGA gain.

**Ctrl-L**: Clears the text area of the LCD.

**Ctrl-K**: Clears the keyboard buffer (while receiving, keystrokes are not displayed — this allows clearing the buffer, so that call signs may be entered, or re-entered in case you think that you have entered the wrong call sign).

**Ctrl-B**: Clears the internal buffers.

**Ctrl-Q** Inserts a TX-OFF control character in the TX buffer, or Macro.

**Ctrl-O** Toggles the display backlight on and off.

Here is a useful combination of macros:

**F1**: CQ

**F2**: Call "them" twice w/ toggle

**F3**: Call "them" once w/o toggle

**F4**: BTU

**F5**: 73

**F6**: Brag File

**F7**: Test message

For your personal macros, choose whatever you want. You can create ones for contesting, or just for casual rag-chewing.

## Updating the Modem with Newer Features

Increasingly today, microcontroller projects have an ability to be "field updated" with new features and software updates

made available by the designers. So, instead of needing to send your instrument back for reprogramming to get these new capabilities and bug fixes, you can simply download the latest-and-greatest software from the Internet and send it to the target hardware. The device automatically updates its internal memory with the new program. What a great way to keep your project completely up to date with the latest features!

We have incorporated this field updating capability into the NUE-PSK Digital Modem. You just need to connect your PC serial port to the modem using a simple adapter, and send it the new software obtained from the NUE-PSK Web site whenever new capabilities are made available.

We designed a TTL serial port into the modem, accessible via a 4-pin connector, P4, located inside the battery compartment. By connecting your computer's USB port to an inexpensive USB-to-TTL adapter such as the CP2102 from SparkFun and plugging the CP2101 (or equivalent) into P4, the modem's "Load Software" menu selection will initiate the bootload sequence to "burn" the new software into the modem's controller.[9] Then, once you power-cycle the modem, you'll be running the latest software release containing, for example, a new digital mode, some new I/O capabilities, and so on. This is really quite a convenient and powerful capability for the project.

## Possible Future Enhancements

Updating the graphics LCD to display current spectral information consumes a considerable fraction of the total processing time. If all LCD display routines were to be off-loaded to a small microcontroller, there would be more time available for processing faster digital modes, such as PSK63.

Additional dynamic range would be possible if an external ADC, with 16 to 24 bits, were to be employed. The Austin QRP group is currently evaluating ADCs and Codecs that might be used in this application.

The next logical step in the evolution of portable amateur radio digital communication is decreased size and increased portability. We envision someday — perhaps sooner rather than later — having a completely integrated, handheld digital modem and low-power transceiver.

## Conclusions

"On-Air" experience with the NUE-PSK Digital Modems has clearly demonstrated the effectiveness of the design, and its suitability for portable digital-mode operations, with an attractive, compact, low-power package. It is also a testament to the wonderful design skills of Moe Wheatley and his *PSKcore* software engine, as well as to how evolving technology continuously improves our options in amateur radio.

We very much enjoyed collaborating on the design of this project with several members of our QRP clubs. We are confident that you will enjoy the flexibility and power offered with the NUE-PSK Digital Modem when used on your bench or out under the stars.

## Notes

[1]Peter Martinez, G3PLX, "PSK31: A New Radio-Teletype Mode", *RadCom*, Dec 1998 and Jan 1999. (Reprinted in *QEX*, Jul/Aug 1999, pp 3-9).

[2]DigiPan software, v1.2 is available at members.home.com/hteller/digipan. DigiPan stands for "Digital Panoramic Tuning" and brings the ease and simplicity of panoramic reception and transmission to PSK31 operation. DigiPan provides a panoramic display of the frequency spectrum in the form of an active dial scale extending the full width of the computer screen. Depending upon the transceiver IF bandwidth, it is possible to "see" as many as 40 to 80 PSK31 stations at one time. DigiPan was developed as freeware by Howard (Skip) Teller, KH6TY/4 and Nick Fedoseev, UT2UZ.

[3]PSK-20 Transceiver Kit for PSK31, Small Wonder Labs, Dave Benson, K1SWL (ex-NN1G). E-mail: dave@smallwonderlabs.com, Web site: www.smallwonderlabs.com

[4]George Heron, N2APB, "Portable PSK" project **www.njqrp.org/portablepsk**

[5]Microchip: **www.microchip.com**. Technical documentation for the entire line of Microchip microcontrollers is available. The MPLAB Integrated Development Environment, and Student Edition C compiler are available for free download.

[6]The AmQRP Club is selling the NUE-PSK Digital Modem for $199 (US & Canada) or $219 (DX) as a fully assembled and tested unit. (Price includes shipping. CA residents please add 8.25% sales tax.) Kits will be offered later this year. To order, write a check/MO payable to "AmQRP Club" in US funds, and send to "The American QRP Club, 2419 Feather Mae Ct, Forest Hill, MD 21050 USA". Payment also accepted through PayPal to amqrpkits@amqrp.org. See the NUE-PSK project page for all details, including source code (www.amqrp.org/kits/nue-psk). We expect to offer full kits (all parts plus housing) and partial kits (PCB and preprogrammed microcontrollers) later in the year.

[7]SchmartBoards: **www.schmartboard.com** See part 202-0011-01 (32-100 pin QFP, 0.5 mm).

[8]Moe Wheatley, AE4JY, "*PSKCore,*" **www.qsl.net/ae4jy/**. For his source code, download PSKCoresrc.zip. For the full technical specification, download "*PSKCore* Interface Specification and Technical Description Ver 1.40"

[9]USB "Breakout Board" Interface, SparkFun CP2102, **www.sparkfun.com/commerce/product_info.php?products_id=198**

### Other Useful PSK31 Technology References:


Don Urbytes, W8LGV, "A PSK31 Tuning Aid," *QST*, Dec 1999, pp 35-37.

Steve Ford, WB8IMY, "PSK31 — Has RTTY's Replacement Arrived?," *QST*, May 1999, pp 41-44.

Steve Ford, WB8IMY, "PSK31 2000," *QST* May, 2000, p 42.

Howard "Skip" Teller, KH6TY, and Dave Benson, NN1G, "A Panoramic Transceiving System for PSK31," *QST*, June 2000, pp 31-37.

Dave Benson, K1SWL (ex-NN1G), "The NJ Warbler — a PSK-80 Single Board Transceiver for PSK31," *QRP Homebrewer*, Summer 2000, pp 15-21.

Johan Forrer, KC7WW, "Using the Motorola DSP56002EVM for Amateur Radio DSP Projects," *QEX*, Aug 1995, pp 14-20.

ARRL Web site collection of PSK31 articles, links, literature and products: **www.arrl.org/tis/info/psk31.html**

The "Official" Homepage of PSK31 is at **aintel.bi.ehu.es/psk31.html**



*Milt Cram, W8NUE, was first licensed in 1953 as WN8NUE and has held several calls (minus the "N") with an Amateur Extra class license. He is a longtime homebrewer and member of the Austin QRP Club, enjoying operating low power and the digital modes on HF. Milt holds BEE, MS and PhD degrees in electrical engineering from Georgia Tech and comes from a family of hams (dad, Ernie, W8JKX (SK), great uncle, Oz, W1JUJ (SK), and son, Marc KC5RWZ). Milt is now retired, after serving many years in electronic design and management.*

*George Heron, N2APB, has been a software developer and technology manager in the northeastern US for more than 30 years, working in later years in the field of information security. He is the chief scientist for McAfee, helping to develop new security products and technologies to protect home and corporate users. A ham since 1968, he is an avid homebrewer in RF and digital circuits, with a special interest in DSP and microcontroller applications to QRP, and has co-developed the Micro908 Antenna Analyzer. He leads the New Jersey QRP and the American QRP clubs, and has previously edited/published* QRP Homebrewer *magazine and* Homebrewer Magazine.


QEX