An Inexpensive Terminal Node Controller for Packet Radio

An inexpensive processor chip forms the basis for a simple-to-build TNC.

By Bob Ball, WB8WGA

Most hams who have tuned around on the 2-meter band at one time or another have heard the infamous *blurrp* of packets being sent back and forth. Sometimes it is an informal QSO, sometimes an Amateur Packet Position beacon, or maybe someone sending info on a hot DX station. Packet radio has found a wide number of applications since its introduction in the 1980s.

Did you ever want to have an inexpensive way to monitor local packet activity? Or perhaps set up your own digipeater to get a communications link to a particular point? Or possibly, actually do some programming on your own simple Terminal Node Controller (TNC)?

If so, this simple project might be the project you have wanted. The unit can easily be built for under $25. After it is completed, the software supplied will:

- monitor local packet activity
- act as a complete digipeater
- send packet beacons at user defined intervals containing your defined text
- allow you to communicate in a round-table fashion using "converse" mode
- send APRS NMEA position reports when connected to a GPS receiver.

Note that this unit does *not* provide all functionality of the complete commercially available TNC units. It is a simplified version that captures enough functionality to get you on the air and have some fun with packet radio. If you want to expand your programming knowledge and explore AX.25 a bit, the unit described will allow you to modify the software to support just about any application you might have been considering for packet radio. This could include, for example, anything from temperature monitoring, GPS interfaces or remote control applications. The list of possibilities goes on and on. All source code for the unit is provided as a starting point and programming is possible without buying an expensive programmer. We will provide additional information on programming the processor later in this article.

I got started on this project after looking at the functionality of some of the new microprocessor chips. They just keep getting more powerful while the cost keeps dropping. After checking out a few different ones, I became attracted to the *MicroChip* series, in particular, one designated the

23 Ingerson Rd
Jefferson, NH 03583
wb8wga@arrl.net

PIC16F88. This 18-pin dual-inline chip sells for about $3 in single units. It provides several analog to digital (A/D) channels, includes a UART to talk to a terminal, a programmable Flash program memory and a programmable EEPROM memory to store call signs and options. It can execute a program instruction every 200 nsec. Amazing! It is certainly not the Intel 8080 I used back in the early 70s! After looking at the unit I began to wonder if it could form the basis of a simple TNC.

One of the things I discovered is that many of the existing PIC processor based packet TNCs use the MX614P modem chip. Modem chips are great building blocks but do add to power consumption and cost. In addition, they can be difficult to obtain. Most of the existing TNC project designs use processor chips that have been replaced by more powerful and less expensive units.

This article will describe how to use the enhanced internal capability of these new processors to code and decode the digital signals. While the project described here is designed for packet radio, the project board and modemless techniques described should work for just about any digital mode (RTTY, Pactor, etc). So, once you have built this unit for packet use, with a little work and ingenuity, it can be reprogrammed for enhanced packet radio or other amateur digital modes.

As with many ham projects, I was able to build on the published work of many others. In particular, Mike Berg, NØQBH,[1] has done some work on modem-less receiver design using an external comparator. A number of PIC programmers[2] have documented resistor ladder network designs for older technology PIC processors that gen-

[1]Notes appear on page 25.

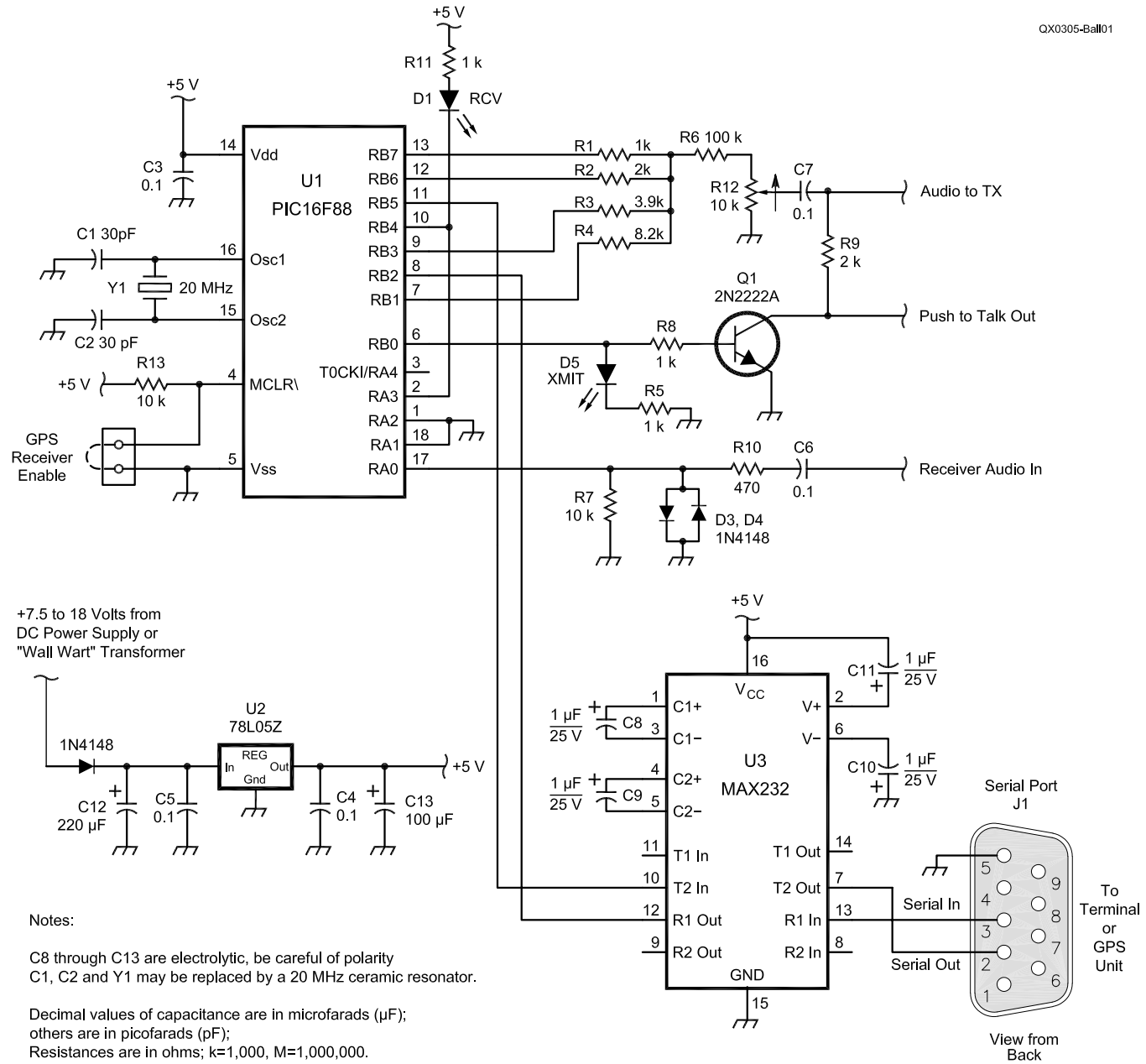

Figure 1—Schematic of the inexpensive TNC.

erate a clean sine wave needed to transmit Audio Frequency Shift Keying (AFSK) tones for transmit. I have combined this existing receive and transmit work, added a command interface which is patterned after the Tucson Amateur Packet Radio (TAPR)[3] terminal node controllers, and made use of the newest PIC technology to produce a simple TNC. It provides the following features:

- A command interface is provided that allows configurable options to be set and stored in EE ram and restored on subsequent resets.
- A monitor feature is included to allow all packets, no packets, or just packets addressed to my station to be displayed on the terminal attached to the serial port.
- Digipeating of packets (up to 255 bytes/packet) is supported.
- Alias capability allows the unit to function as RELAY type digipeater.
- Beacon capability with user set parameters is supported.
- Converse mode operation to transmit text typed from the serial port supporting "round table" chat type operation is provided.
- Support is provided to transmit APRS position beacons when a GPS unit is attached to the serial port.

Circuit Description

A schematic of the TNC is shown in Figure 1. It contains two integrated circuits, one for level conversion of the RS232 signals and the microprocessor. The microprocessor is the 18-pin DIP version of the PIC16F88. The popular MAX232 chip does the RS232 conversion.

Connections to the radio consist of audio input from the speaker jack, audio to the microphone circuit, and a connection to the push to talk circuit.

The serial port is used to connect to a standard terminal program or to a GPS receiver that outputs NMEA sentences. The terminal program is used to display received packets, send text while in the CHAT mode, and do some initial setup of the unit. Jumper J4 is used to specify what is connected on the serial port.

Note that after initial setup is complete and the station parameters are set in EERAM, the unit will run as a super simple remote digipeater without the MAX232 and serial port connections. With this configuration, it becomes a one-chip TNC.

Power to the unit can be anything from 7 to 18 V. Power from the 12 V radio can be used, or an inexpensive *wall wart* will do the job.

Receiving Packets

Packet radio, in common with many digital transmission techniques, uses *Audio Frequency Shift Keying* (AFSK) to send the stream of *ones* and *zeros*. It uses two frequencies, 1200 and 2200 Hz to represent a change in the value of the bit stream. For receiving purposes, this TNC uses a comparator to find zero crossings of the AFSK sine wave. The TNC calculates the interval between zero crossings to determine the frequency present. The software then converts the changes in tones to a data stream. To imagine this, think of a sine wave at either 1200 or 2200 Hz. If we clip the sine wave with a couple of diodes (D1 and D2 in the schematic), we get a square wave. The circuit is shown in Fig 2.

If the resulting square wave is fed to a zero-crossing-detector, a pulse is generated at every crossing. If the processor internally times the interval between these pulses, it can determine the frequency of the sine wave. The F88 processor has timers and an internal comparator circuit so all that decoding work can be done inside the chip. The software inside the PIC can also do some digital filtering and throw out frequencies outside of the passband of the audio. The resulting modem-less design is extremely sensitive and rivals the performance of the commercial modem chips like the MX614. My units routinely decode packets that are less than 1 S unit on my 2-meter radio from stations over 75 miles away.

It should be noted that the above technique could be used for sampling the frequency of any sine wave. It should work for any of the digital modes that use AFSK or for any applications that need to detect an audio tone of a certain frequency.

Transmitting Packets

To transmit, the software needs to generate either a 1200 or 2200 Hz sine wave. The frequency is changed to indicate the transmission of a digital one or zero.

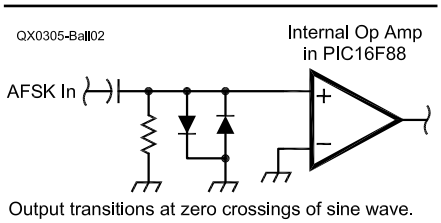

Figure 2—Determining the frequency of a sine wave using a zero crossing detector.

A sine wave is no more than a varying voltage level over time. If the processor can change the voltage level in the right amount and time frame, a sine wave with low distortion can be produced. The TNC must only generate a single tone at a time, although production of multiple simultaneous tones (i.e. DTMF) is possible. MicroChip has documented this technique in one of its application notes if you are interested in more details.

This generation can be done in software by controlling a resistor ladder network that has the correct values to generate a sine wave. If the voltage is changed often enough (say 32 times per audio cycle) at the right value, a sine wave is produced. Figure 3 shows the voltages that are generated at the wiper of R12 when the Ports RB7, RB6, RB5 and RB1 are taken through a binary sequence by the program at the correct frequency. Increasing the number of samples decreases the distortion of the sine wave but increases the real-time requirement. A sine wave adequate for this application can be generated using 32 points.

Software

Approximately 3000 words of program instruction reside on the PIC chip to do the job of taking the received packet, storing it, checking it for validity, sending the info to the serial port, and re-transmitting the packet as necessary. If you are not interested in studying or modifying the provided software, a copy of the working assembled software is provided that can be downloaded into the processor (see next section).

If you are interested in learning about PICs or how packets are formed, etc, you might want to modify the code to meet your needs. While the complete software structure will not be described here, a copy of the commented source code is available[4] and can be used as a starting point for further experimentation. The software is organized by functional processes (ie receive packet, digipeat, command processor, etc) and scheduled in a round robin manner to make modification easy. The terminal interface is implemented as a table driven function, so that additional input commands can be easily added.

The software in this unit uses published snippets of code from many hams who have shared their work on the Web. References and credits are given to all the authors in the source software. Please remember to maintain these credits if you modify the software and pass it on to others.

Construction

Construction and layout of this unit is not critical. One option is to build on a standard perforated board. The unit shown in Figure 4[5] was built on standard RadioShack prototype board with two aluminum angle brackets for supports. As with all CMOS integrated circuits, use caution when inserting the ICs to avoid electro static damage. A good wrist-grounding strap is very useful.

This unit runs at a relatively fast clock speed so bypassing of the 5 V power is important. The 0.1 µF capaci-

Spreadsheet to Calculate Sine Wave Value for TNC Transmitter							
R1	1000						
R2	2000						
R3	3900						
R4	8200						
R6	100000						
R12	10000	This is with wiper at top of pot					
					Ladder	Time	To
	MSB			LSB	Out		Transmitter
Value	V1	V2	V3	V4	Eo	t	Vo
7	0	5	5	5	2.33	1	0.21
6	0	5	5	0	2.00	2	0.18
5		5		5	1.65	3	0.15
4		5			1.32	4	0.12
3			5	5	1.00	5	0.09
2			5		0.68	6	0.06
1				5	0.32	7	0.03
1				5	0.32	8	0.03
1				5	0.32	9	0.03
1				5	0.32	10	0.03
2			5		0.68	11	0.06
3			5	5	1.00	12	0.09
4		5			1.32	13	0.12
5		5		5	1.65	14	0.15
6		5	5		2.00	15	0.18
7		5	5	5	2.33	16	0.21
9	5		0	5	2.97	17	0.27
10	5		5	0	3.33	18	0.30
11	5	0	5	5	3.65	19	0.33
12	5	5		0	3.97	20	0.36
13	5	5	0	5	4.29	21	0.39
14	5	5	5	0	4.65	22	0.42
15	5	5	5	5	4.97	23	0.45
15	5	5	5	5	4.97	24	0.45
15	5	5	5	5	4.97	25	0.45
15	5	5	5	5	4.97	26	0.45
14	5	5	5	0	4.65	27	0.42
13	5	5	0	5	4.29	28	0.39
12	5	5	0	0	3.97	29	0.36
10	5		5		3.33	30	0.30
9	5	0	0	5	2.97	31	0.27
	Note when Vo is Plotted over time, a sine wave is produced						

Figure 3—Sine wave generation on a PIC.

tors on each integrated circuit should be located right at the device with leads of one inch or less. I can't tell you how many times I have spent time debugging a digital circuit problem only to find out I had noise on the 5 V line. Bypass capacitors are cheap and small, use them freely.

If you make your own board for this project, it is always good to do some checkout before you install the integrated circuits. First apply 12 V and check for +5 on pin 14 of the PIC16F88 and pin 16 of the MAX232. If this looks good, install the chips using the wrist strap. Don't forget to program the microprocessor chip (see later section) before installation.

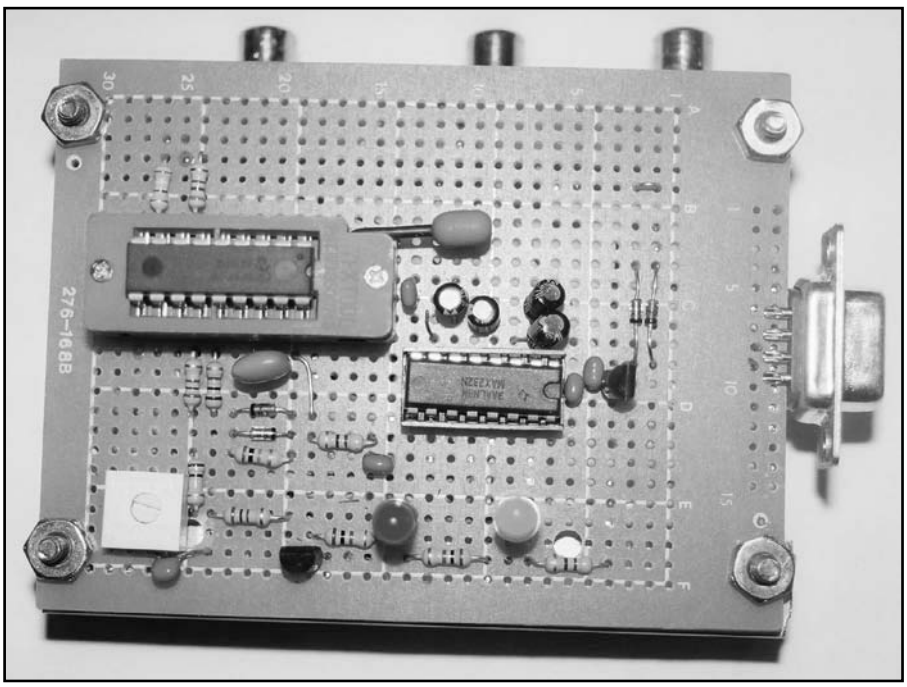

Figure 4—Breadboard style unit.

After building the prototype shown in Figure 4, I discovered another construction option, the popular Olimex prototyping board. This is the fastest way to construct the TNC.

Many projects need the basic processor, crystal, power circuit and serial connection. Olimex has provided a built and tested circuit board with this much of the circuit complete. It also includes an area to build the rest of the circuit. In the case of the TNC, approximately ½ of the components are already on the board.

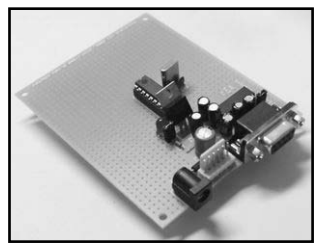

Figure 5—Olimex prototype board.

The purchased board from Olimex contains the processor chip, all serial

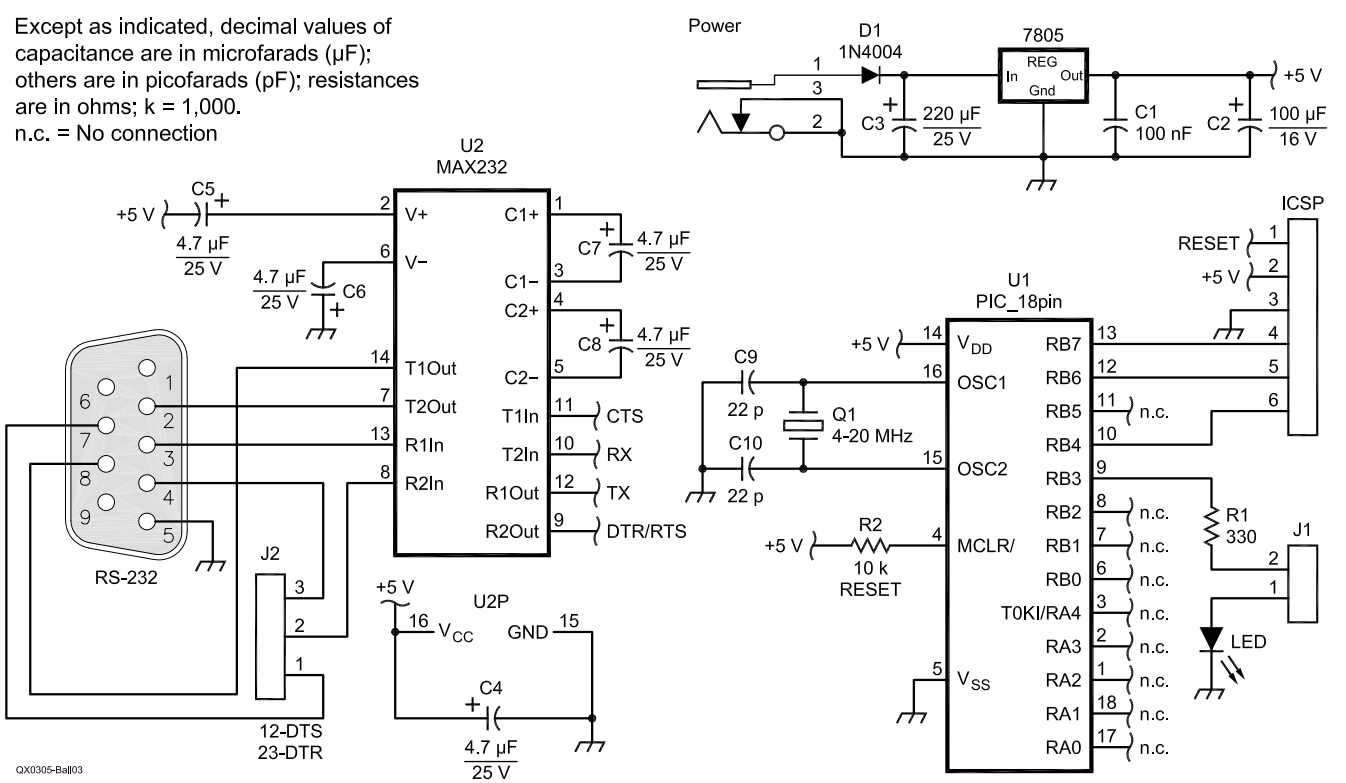

Figure 6—Schematic of the Olimex board.

Detailed Instructions for the Olimex Board

When you receive your Olimex board, it is a good idea to check it out before you add any wiring. I recommend installing the Boot Loader program which is supplied with a small checkout program that will flash the leads on the board.

Initial Checkout

1. Remove the supplied PIC16F88 and program it with the supplied Bloader Program. This should be the only time you will need an external programmer for this project. REMEMBER TO DISABLE THE MASTER CLEAR WHEN THE CHIP IS PROGRAMMED.

2. Reinstall the processor chip. With no serial port connected, power up the unit. The LEDs should alternately flash. If they do not, stop here and get a replacement board from your supplier.

Now go ahead and add the TNC wiring:

Serial connections

1. Remove J1
2. If GPS operation is desired, install wiring for GPS Receive Enable. Do not install jumper until set up is complete (see text).
3. Lead Labeled RX to Processor Pin 11
4. Lead Labeled TX to Processor Pin 8

TNC Receive Circuitry

1. Ground Pins 1 and 18 of Processor
2. Connect Processor Pins 2 and 10 to D1 Cathode
3. Connect R11 to 5V
4. Connect R10,D2, D3, R7 to Pin 17
5. Ground other lead of D2,D3, and R7
6. Connect R10 other end to C6

TX Receive Circuitry

1. Connect one end of R1,R2,R3, and R4 to Processor Pins 13,12,9, & 7 respectively.
2. Connect other end of R1,R2,R3, and R4 to R6
3. Connect other end of R6 to R12
4. Ground other end of R12
5. Connect one end of C7 to R12
6. Connect other end of C7 to output and R8
7. Connect Pin 6 of Processor to D5 Anode and other end of R8
8. Connect other end of D5 cathode to R5
9. Connect other end of D5 to ground
10. Connect R8 other end to Q1 Base
11. Connect Q1 Emitter to Ground
12. Connect Q1 Collector to R9 and PTT out
13 Connect other end of R9 to C7

Wiring is Complete. Now install the TNC software

1. Remove Power from the TNC
2. Connect a serial cable from the serial port to your PC or terminal. Start up the terminal program.
3. Remove J4
4. Install the PC portion of the loader program called Screamer
5. Start up Screamer. Hit the program button and supply the file name for the TNC software. Screamer should say "Waiting for Broadcast"
6. Power up the TNC, Screamer should program the chip and indicate successful completion
7. Exit Screamer program on PC and Start Terminal Program
8. The command interpreter should be running on your TNC. Set your options, connect the radio, and your on the air.

Figure 7—Step-by-step wiring instructions for the Olimex board.

channel components and all power components on a 100×80-mm board. With the addition of a few components, the unit is complete. Connectors for the radio interface can be based on personal preference. I use RCA audio jacks for the audio in and out and a standard phone plug for the push-to-talk connection. This allows use of the readily available audio/video cables for the audio in, transmit audio and PTT. You may want to select these connectors to meet your need.

The Olimex board does not raise the cost of the project significantly. A constructed prototyping board for the PICF88 is available for $12.95 plus shipping directly from Olimex. Other sources of the board[6] provide the board equipped with the PIC16F88 for $18.95 plus shipping. A picture of the Olimex board as it comes out of the shipping box is shown in Figure 5. The schematic of the board is shown in Figure 6. Step-by-step wiring instructions to add the parts to the Olimex board are shown in Figure 7.

A picture of the completed TNC using the Olimex board is shown in Figure 8. Note the case, which is based on a 71 cent plastic school supply container I purchased at my local WalMart store. It does a nice job of keeping the unit clean and dry.

Programming the Unit

Code Assembly

The hex file output for the TNC is provided with the software and may be used directly. If you do not wish to reassemble the software, skip this section.

If, however, you have decided you would like to experiment with the assembly code for this program, the provided source will need to be reassembled. This can be done with the free assembler and simulator provided by MicroChip. First, download and install the free MP Lab development system available from the MicroChip Web site on your PC. Using the instructions provided, assemble the provided TNC assembly file. **In the project options of MPLAB window, set the output format for the hex file to INHX8M.**

Programming the Flash Memory

Now that your code is assembled, there are a couple of options for programming the PIC16F88 chip. If you own a PIC programmer, or have a friend who has one, one method is to just download the provided hex files and you're done. If you choose to do this, the next section on the boot loader can be skipped.

Programming with the Boot Loader

If you intend to experiment with the software or install software updates, the *boot loader* is a nice way to program your unit. Once it is installed, it is much faster and convenient than using an external programming unit. With the boot loader, the chip can be reprogrammed without removing it from the circuit board using the same serial connection that is used during normal operation.

To do this, MicroChip boot loader programmers use the PIC16F88 feature that allows it to write its own instruction words. With this feature, a user can do a one-time program of a small program (called a boot loader) in high flash memory and then use this program to download the user software over the serial link. This means that the user using a regular PC with a serial link to the TNC can load software updates. Several free boot loader programs are available on the Web. I have included a site,[7] Spark Fun Electronics, that provides a freeware version that works well with the TNC. I have also included a copy of the boot loader (with permission) for the PIC16F88 from the Spark Fun Electronics site with the source code.

- Download and install the PC portion of the programmer (called *Screamer*) It is fast!
- Download the F88 version of the bootloader hex file (called Bloader -F88-20Mhz.hex) from the web site.
- Using a PIC programmer, complete the one time program of the F88 chip using the hex *Bloader* code. **When setting up the programmer for this step be sure to disable the Master Clear pin on A5 (MCLR on Pin) in the Config word. This allows PORT A5 to be used as an input lead for the GPS function.**
- Apply power to the TNC. The LEDs should alternatively flash as an indicator that the boot loader program was successfully installed.
- Connect the serial cable between the PC and the TNC. Start the *Screamer* programmer program on the PC. Select the TNC hex file (Modemless TNC V1) and click on **PROGRAM**.
- When the unit indicates *Waiting for Pic Broadcast*, cycle the power (allow 3 or 4 seconds for the capacitors to discharge) on the TNC. *Screamer* will then program the Chip in about 10 seconds.
- Close the *Screamer* program and reopen the terminal program on the PC. The command interpreter should respond to your commands. More info on using *Bloader* is provided in the *readme* file section in the bootloader code.

Your unit is now programmed. The boot loader should remain there for the life of the chip. To reprogram the unit in the future, repeat steps five through seven.

EE Program Memory

Once the program is loaded and

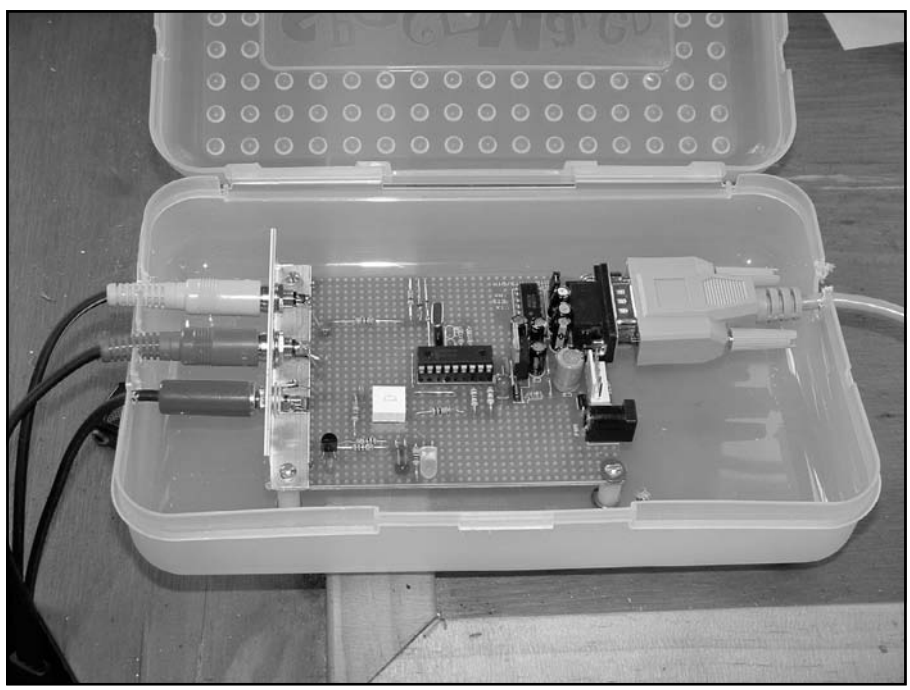

Figure 8—Completed TNC built on the Olimex board.

operational, the unit must be customized for the station using it. Information on the station call sign, beacon info, etc, must be programmed and retained. Like most TNCs, this is done from a terminal. The EE RAM of the PIC is used for this and the information is obtained over the serial port. Then a command (PERM) is executed to write the info into the EE RAM. Once the *perm* command has been executed, the data will last for many years and will withstand program changes and power transitions.

Setting the Options

Station options are set via an ascii terminal on the serial port. I use *Windows Hyperterminal*. It comes with the standard *Windows* software. Other terminal emulators will work as well. Set the terminal up at 9600 baud, 8 bits, no parity, Back Space (BS) for the rubout character, and echo. If the *echo* option is not available on your terminal program, the TNC does provide a command option to allow local echo.

After these options are set, connect a standard serial cable between your terminal and the TNC. After powering up the TNC the following message should be shown.

WB8WGA Modemless TNC V1.08
Type "Help" for Information
cmd:

If this doesn't appear, start checking the wiring around the microprocessor chip and serial connection for errors.

Once you have that working, it is time to receive some packets. Connect an audio cable from your 2-meter speaker audio to the TNC. Open the squelch and set the audio level to the point at which the LED lights. Now close the squelch and the LED should go out. If this step doesn't work, check out the wiring on the incoming audio lead and the LED wiring.

Once this is operational, you should be seeing packets being monitored on your screen.

Before we transmit with your new TNC, it is necessary to set your station parameters and save them in EERAM. The FCC seriously frowns on packets that aren't properly identified. While in command mode, the TNC will accept either upper or lower case characters.

First, you will need to set your call sign using the *mycall* command as follows:

cmd: mycall WB8WGA
OK
cmd:

Note that all callsigns have a 0 to 15 sub id, which is defaulted to zero.

The other parameter that must be set is destination call sign. This is used when the unit transmits beacons, APRS reports, or enters chat mode. This information is set via the *unproto* command. The format for this command is *unproto callsign1 v callsign2,* where callsign1 is the final destination and callsign2(optional) is the digipeater that should be used to get the packet to the destination. A max of 3 digipeaters may be used. An example is shown below:

cmd: unproto n0qbh-15 v wb8wga-10 v relay v wide
OK
cmd:

Transmitting Packets

Now it's time to send out some *blurrps*. Hook up the transmitter connections, audio and push-to-talk. Once connected to your radio, it is necessary to set the transmitter deviation. This is best accomplished with a calibrated deviation meter.

The *calibrate* command is provided to help with this step by providing a solid 1200 or 2200 Hz tone for the adjustment. It keys the transmitter and alters between the two tones each time the space bar is pressed. The opera-

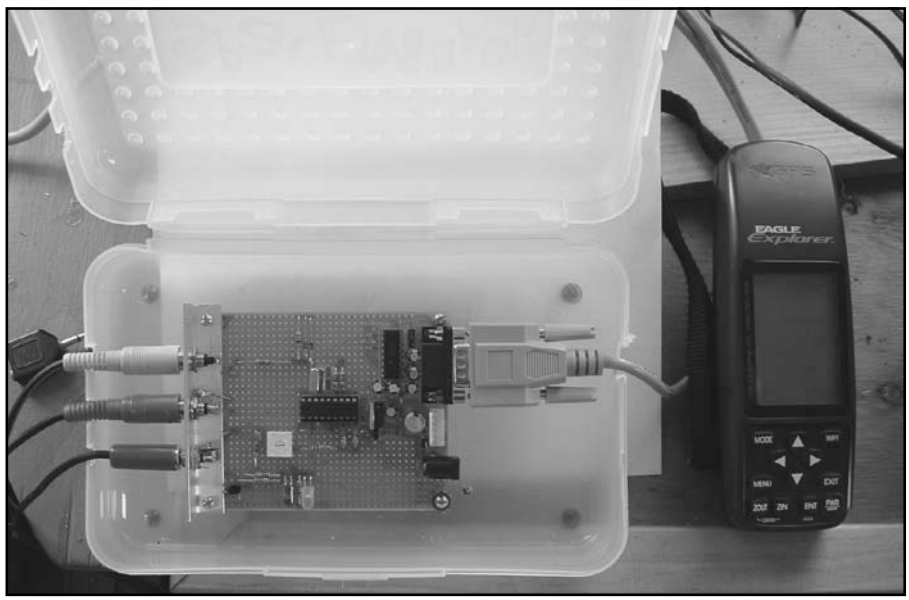

Figure 9—GPS unit connected to the TNC.

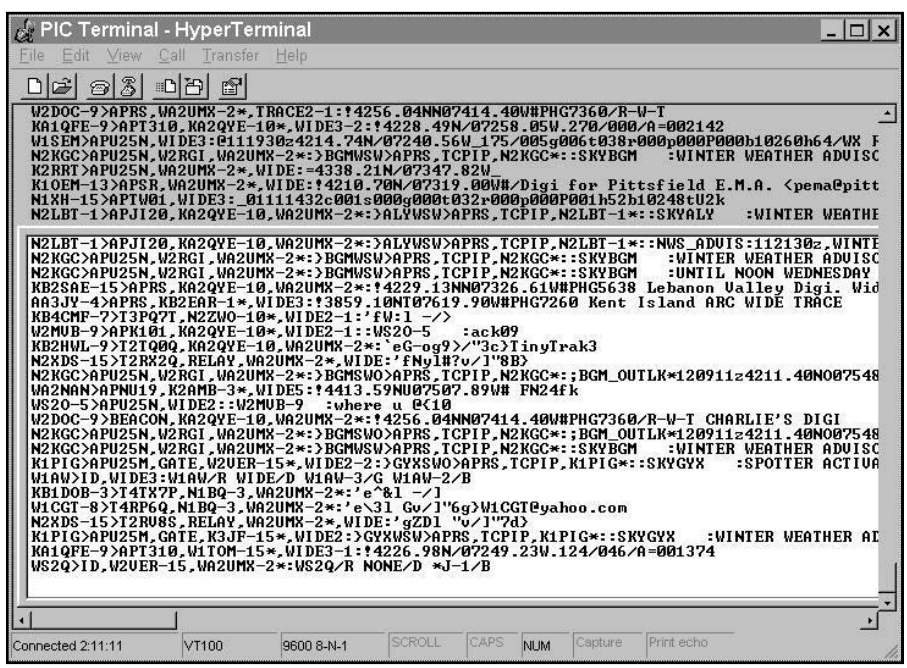

Figure 10—Packet output screen using *Hyperterminal* with monitor enabled.

tion is as follows:

cmd:CALIBRATE
Calibrate Mode. Space Bar to Toggle, Control C to Exit
<ctl-c>
OK
cmd:

You may also need to adjust the number of flags at the beginning of the packet to allow your transmitter to stabilize. This is done using the *txdelay n* command. Each increase in the number *n* adds one more flag (approximately 6 ms). The command syntax is:

cmd: TXDELAY 60
OK
cmd:

Your TNC should now be ready to go on the air. Tune your radio to a local packet frequency.

Beacon Operation

If you want your unit to send out beacons, you must enter some beacon text. This is done by entering:

cmd: btext *This is Bob in Jefferson, NH*
OK
cmd:

Then, the beacon interval can be set. Beacons can be set between 1 and 59 minutes. As an example, a beacon every half hour would be:

cmd: beacon every 30
OK
cmd:

The beacon may be turned off by entering:

cmd: beacon every 0
OK
cmd:

Monitor Operation

Three options are provided for the monitoring of packets. If you do not wish to use the serial interface of the unit (ie a dedicated digipeater), monitoring can be turned off. This can be done by the following command:

cmd: monitor off
OK
cmd:

Monitoring of all packets, independent of their call sign addresses would be selected by:

cmd: monitor all
OK
cmd:

Selective monitoring of packets addressed to *mycall* or *alias* can be accomplished by setting the monitor as follows:

cmd: monitor me
OK
cmd:

A screen capture of actual packet traffic reception as captured by *Hyperterminal* is shown in Figure 10. In this case, the *monitor* option was set to *all*.

If you don't leave a terminal connected to your TNC for monitoring (ie, a standalone remote digipeater application), it is a good idea to turn the monitor off when the EE ram configuration is complete. This will reduce power and real time consumption in the processor. The monitor feature is automatically shut off if the unit is in GPS receiver mode (Jumper J4 installed).

Alias Operation

Alias operation is optional and is used in the receive process. It allows your unit to respond to packets that might not be addressed directly to your call sign. The alias can be any 7-character string, with optional SSID. As an example, setting my alias to *relay* is done as follows:

cmd: myalias relay
OK
cmd:

Digipeating

Digipeating may be turned on and off by the following command:

cmd: digi on
OK
or
cmd: digi off
OK

If digipeating is on, any packet matching *mycall* or *myalias* will be *digipeated* on to the next station specified in the header.

Saving Your Station Parameters

After you have inputted all your parameters, you can display them all to make sure everything is correct. This is done with the *disp* command and should look as follows:

cmd: disp
TXDELAY 60
ECHO on
GPS $GPRMC
TRace off
MONitor on
DIGIpeater on
BEACON on EVERY 20
UNPROTO aprs v relay v wide v wide
MYCALL wb8wga-0
MYALIAS unit1
BTEXT Bob in Jefferson, NH
cmd:

Don't worry about the GPS or TRACE commands for now, they will be described in a later section.

If everything is correct, the parameters can be saved in EE memory with the permanent command as follows:

cmd: perm
OK
cmd:

Trace Function

For software testing or experimentation, a trace command is provided. It gives a hexadecimal display of a packet being formatted for transmit but the actual transmit is not completed. It also provides a hexadecimal display of received packets. Its syntax is:

cmd: TR xmit
OK
cmd:
or
cmd: TR rcv
OK
cmd:

Converse Mode

The TNC will allow the transmission of unnumbered information (UI) frames of anything that is typed from the serial port. This allows for a *chat* mode in which users can converse in a roundtable type operation. This is done by entering converse mode as follows:

cmd: converse
Entering Converse Mode, Hit Control C to Exit
Hello World
<ctl C>
cmd:

While in converse mode, other stations on the frequency that send frames will have their frames displayed as well, allowing for a roundtable type operation.

APRS Operation

Automatic Position Reporting System (APRS) is a fun and useful aspect of packet and Amateur Radio. The APRS system provides a good means to transmit position information, weather information, and messages. This TNC software supports some of the common APRS functions used by many hams today on the 2-meter band (typically 144.39 MHz) so it can get you on APRS very inexpensively. Several good books[8] are available from the league on APRS to get you going.

If you are using your TNC as part of a home QTH setup, you can get your newly built TNC registered in the APRS system, see your reports on the Internet, and at the same time provide a digipeater for local hams. To do this, you simply need to set the *unproto* and beacon text. Also *digi* must be turned on and *myalias* turned on if you wish to be a digipeater. As an example, I use the following command sequence to put one of my units as a digipeater from my home QTH:

cmd: myalias RELAY
OK
cmd:btext !4424.17N/07126.40W# /R Bob in Jefferson,NH

OK
cmd: unproto APRS v RELAY v WIDE v WIDE
OK
cmd:digi on
OK
cmd:

Note that you need to substitute your fixed position longitude and latitude (mine is 44.24.17 N and 71.126.40 W). The # indicator at the end indicates you serve as a digipeater.

Sending National Marine Electronics Association (NMEA) Sentences from a GPS

Another way to send APRS packets is by using your GPS unit to provide the position information. The TNC beacon feature is used to transmit location reports from a GPS unit using the serial port. The GPS unit is connected to the serial port instead of a terminal and NMEA-0183 sentences are transmitted each beacon interval instead of beacon text (btext). A picture of the TNC connected to an EAGLE TNC via the serial port is shown in Figure 9. Note to activate this feature. The GPS receiver enable function must be set **at initialization time**, telling the software to interpret input on the serial port as NMEA sentences rather than TNC commands. To return to regular mode, simply recycle the TNC power.

NMEA Setup for Your GPS

There are many types of ascii sentence structures available from GPS units. The TNC software supports $GPGGA, $GPGLL and $GPRMC sequences. These are the most common but check your GPS manual to ensure your data is compatible with one of the protocols. When the beacon timeout in the TNC has occurred, it will wait and send the next valid NMEA sentence to come from the GPS unit. Thus, your most recent location is transmitted. This sentence type must be set from the command line as follows:

cmd: GPS $GPGGA
OK
cmd:
or
cmd: GPS $GPRMC
OK
cmd:
or
cmd:GPS $GPGLL
OK
cmd:

Also remember that *mycall, unproto, and beacon* parameters must be set before the beacon can be transmitted. A common *unproto* for APRS is:

cmd: unproto APRS v RELAY v WIDE v WIDE
OK
cmd:

Check with other local APRS users to get the best route for your area.

The *beacon every n* should be set to a reasonable value to capture your position changes while not overburdening the APRS system with reports. I use 15 minute positioning reports on my system.

These options can be made permanent by using the *perm* command. After this is completed, the GPS jumper (J4) is inserted and power recycled. The unit will then start sending your position reports.

Note that the unit will continue to *digipeat packets* while in the GPS mode, assuming *digi* is on.

Future Steps

I hope this project will get you interested in packet radio operation and microprocessor usage. The hardware unit is just a starting point and is flexible enough that it can be programmed to do many useful things in your shack. Use of the Olimex experimenters' boards make it easy to get a fast start on a new project. The modemless receive and transmit techniques are good starting points for other digital modes. Use of the boot loader concept provides a lot more flexibility for changes to your ham projects. After you gain an understanding of AX25, what you put in the packets from your TNC is only limited by your imagination. Hope to hear your *blurrrp* on the air soon.

Bob Ball, WB8WGA, is a retired electrical engineer who lives in the White Mountains of northern New Hampshire. His working career spanned 30 years in the Columbus, Ohio area as a technical manager at Bell Laboratories. Bob holds a Bachelor of Science Degree in Electrical Engineering and a Master of Science Degree in Computer Science, both from the Ohio State University. He holds an Extra Class license and has been continuously licensed as an amateur operator for 40 years, first licensed at WA3ANV in 1964. Bob enjoys many facets of Amateur Radio but can most often be found on the bands running 40 and 20 meter CW.

Notes

[1]Mike Berg's work on modemless packet decoders for AX.25 can be viewed at his web site, **www.ringolake.com**.

[2]Many references on using a ladder network for generating sine waves are on the Web. MicroChip (**www.microchip.com**) has an application note and schematics, AN655, giving more information on the technique. Byon Garrabrandt, N6BG, uses this technique in his popular original Tiny Trak unit implemented on the PIC16F84. The schematic can be viewed at Byon's web site at **www.byonics.com/tinytrak/tinytrak.zip**.

[3]Many excellent resources are available at the TAPR Web site (**www.tapr.org**). Published source code contributions have been used in this project as well as information from the on-line TNC2 manual. Good papers exist on their site for understanding the AX.25 protocol. I used the AX.25 protocol specification and an excellent paper by John Hansen, W2FS, on how to implement the protocol. The command sequences are similar to the TAPR TNC1 and TNC2 Units.

[4]Hex and source files for this project are available at the League Web site (**www.arrl.org/qexfiles**).

[5]Photos for this article were provided with permission by Imre Szauter.

[6]Olimex boards with an installed PIC16F88 are available at Spark Fun Electronics (**www.sparkfun.com**) for $18.95 USD. Pictures of the board, software, and schematic were obtained (with permission) from the Web site.

[7]A boot loader for the PIC16F88 is available at no cost at **www.sparkfun.com**. It consists of 2 software parts, the small portion that is programmed on the chip (call Bloader) and the PC portion (call Screamer). For convenience, a copy of the Bloader and Screamer is also provided at the League Web site.

[8]APRS—"Moving Hams on Radio and the Internet," by Stan Horzepa, WA1LOU, provides a good overview of the history and working of APRS. □□