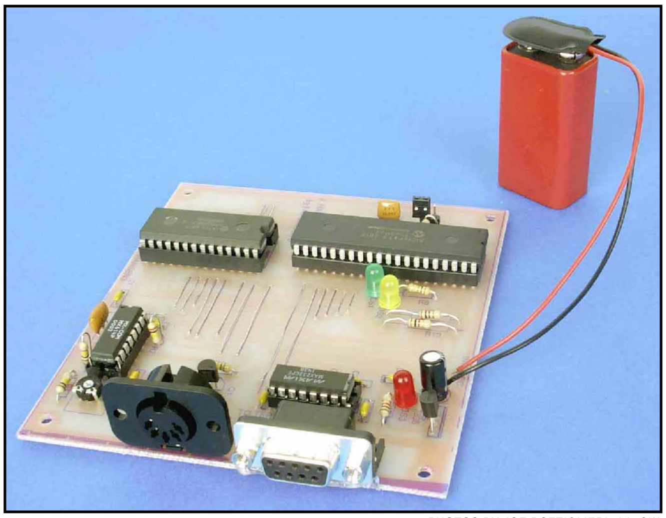
By John Hansen, W2FS

An Inexpensive KISS-Mode TNC

Four ICs on a slice of PC board aren't edible, but the combination makes a deliciously simple TNC!


PHOTOS BY JOE BOTTIGLIERI, AA1GW

For some time, I've thought it possible to use an inexpensive PIC microcontroller with an equally cheap modem chip to construct a packet-radio TNC that would be small, cheap and consume little power. In this project, that idea became real.

Background

This isn't a full-featured TNC, but one that is designed to operate primarily in KISS mode. KISS mode was developed by Mike Chepponis, K3MC, and Phil Karn, KA9Q, and is one of the modes now included in almost all commercial TNCs. If you think of the computer and TNC as the non-RF parts of a data-communication system, originally almost all of the system's intelligence was built into the TNC, not the computer. As a result, it was possible for completely dumb terminals to be used with TNCs to provide packet communications. This was done because the personal computers available when TNCs were first developed weren't very powerful. With time, computers became substantially more powerful. Taking the opposite approach, the Baycom and Poor Man's Packet modems move all of the intelligence out of the TNC and into the computer. KISS mode plows a middle ground, moving a portion of the intelligence from the TNC to the computer, but leaving some intelligence in the TNC as well.

A KISS-mode TNC can't be used in conjunction with just any terminal program. That's because certain TNC functions must be carried out in the computer, not the TNC. However, a significant amount of software has been created that supports KISS mode, including APRS, TCP/IP, my own *HamWeb* software and a range of other programs. (See the sidebar "KISS Mode Packet-Radio Software.")

In addition to being a KISS-mode TNC, this TNC has a second mode. It can be interfaced with a GPS receiver and used to send APRS position reports (in Mic-E compressed format) and monitor incoming unconnected (UI) packet frames.[1] Here, I'll refer to this as "APRS Tracker mode." In this mode, the data can be displayed using any terminal program available on any computer; it does not require KISS-compatible software.

Circuit Description

The hardware for this project (see Figure 1) is relatively simple since most of the heavy lifting is done by the firmware in the PIC. U1 is a *programmed* PIC16F877 microcontroller.[2] This chip sends and receives data from the computer or terminal, formats incoming and outgoing packets, receives and interprets data from the GPS stream (if one is used) and drives the modem chip.

Even if the radio channel is busy and transmission is delayed, data continues to flow from the computer to the TNC. Because KISS mode does not support hardware or software flow control, a substantial amount of static memory is required. To avoid losing data, it's necessary to have a sizeable transmit-data buffer. The PIC itself doesn't contain enough memory to provide this function. Fortunately, static RAM chips of sufficient capacity are now extremely inexpensive. U2, a 62256 32-kB static RAM chip, is one of these devices. Of the 32 kB of memory contained in the 62256 static RAM, 28 kB are devoted to the transmit buffer. In addition, there is a 4-kB receive buffer. This buffer is required because the unit must accumulate an entire receive frame before it can check to ensure the CRC calculation matches the data in the receive frame and decide whether to discard it (if there is an error) or send it on to the computer (if it is not in error).

U3 is a MAX232 level-conversion chip. The PIC microcontroller communicates at TTL levels (0 to +5 V), while almost all computer serial ports communicate at RS-232 levels (±12 V). U3 handles the conversion in both directions. U4 is an MX-Com MX-614 modem IC. It takes the signals coming from the PIC and

[1]Notes appear on page 56.

KISS-Mode Packet-Radio Software

Here's a listing of just a few of the packet-radio programs that support KISS. Please note that I have not tested all of these programs; I list them here solely because they claim to work well with KISS-mode TNCs.

JNOS (TCP/IP Package):
http://www.tapr.org/tapr/html/softf.html

Pr4Win (general Windows-based Packet program):
http://www.geocities.com/SiliconValley/Vista/9244/

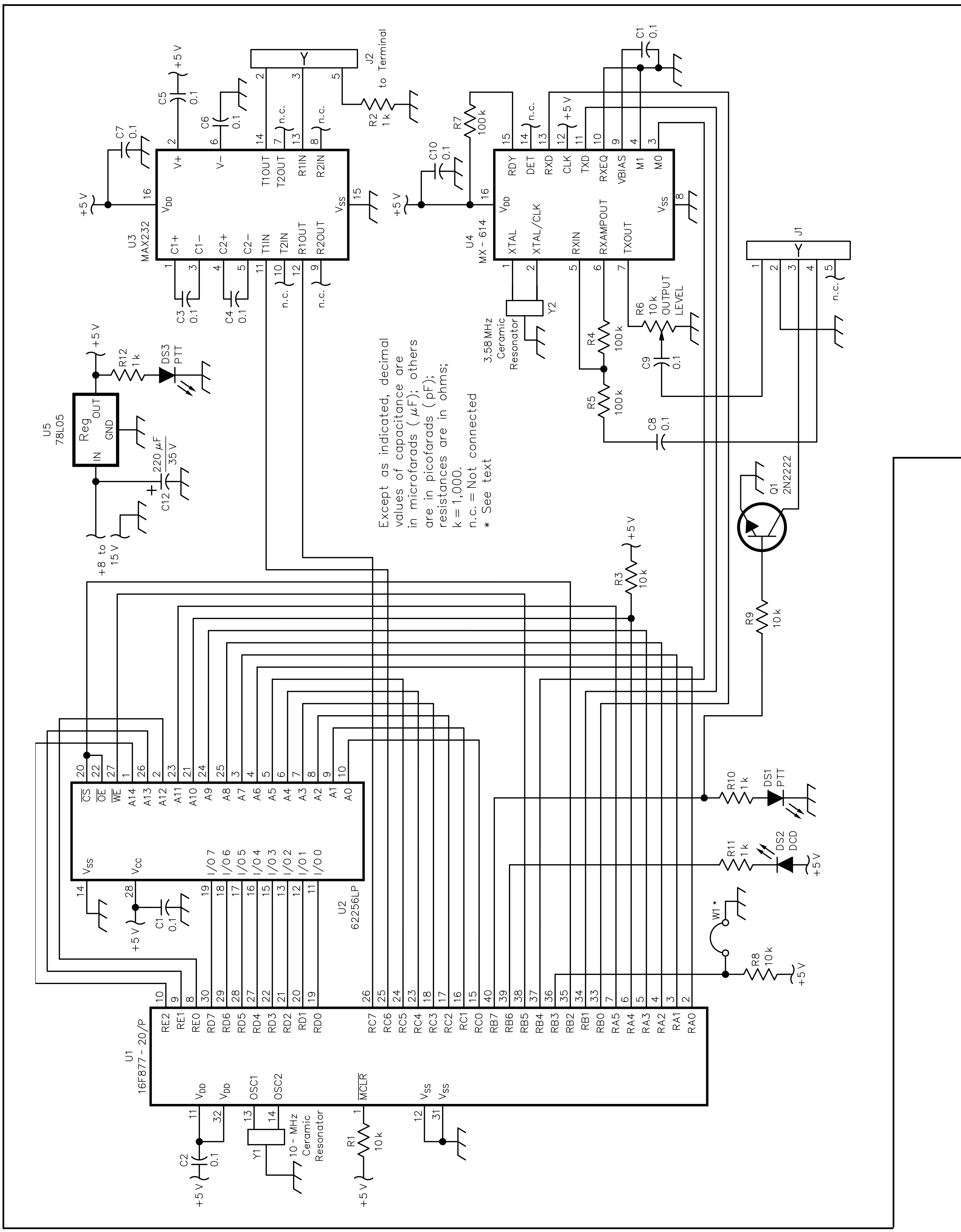
WinTNC (another packet terminal program):
http://www.tapr.org/tapr/html/softf.html

WinAPRS (Windows-based full-featured APRS program):
http://www.tapr.org/tapr/html/sigf.html

CLX (DX Cluster software): **http://www.clx.muc.de/**

G8BPQ (Packet switch software):
http://www.tapr.org/tapr/html/softf.html

—John Hansen, W2FS

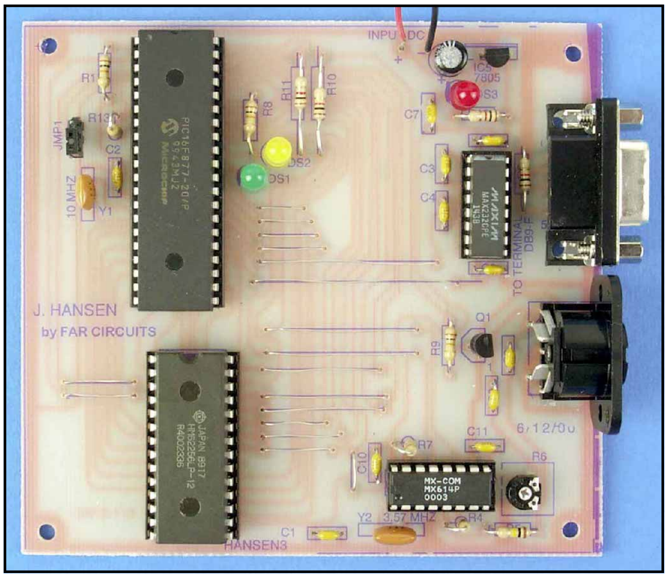


Except as indicated, decimal values of capacitance are in microfarads (μF); others are in picofarads (pF); resistances are in ohms; k = 1,000.
n.c. = Not connected
* See text
U1 16F877-20/P
U2 62256LP
U3 MAX232
U4 MX-614
U5 78L05
Reg
IN
GND
OUT
J1
J2 to Terminal
Y1 10-MHz Ceramic Resonator
Y2 3.58 MHz Ceramic Resonator
Q1 2N2222
DS1 PTT
DS2 DCD
DS3 PTT
R1 10 k
R2 1 k
R3 10 k
R4 100 k
R5 100 k
R6 10 k OUTPUT LEVEL
R7 100 k
R8 10 k
R9 10 k
R10 1 k
R11 1 k
R12 1 k
C1 0.1
C2 0.1
C3 0.1
C4 0.1
C5 0.1
C6 0.1
C7 0.1
C8 0.1
C9 0.1
C10 0.1
C12 220 μF 35 V
W1 *
+5 V
+8 to 15 V
n.c.

Figure 1

Figure 1—Schematic of the KISS TNC circuit. Unless otherwise specified, resistors are 1/4-W, 5%-tolerance carbon-composition or metal-film units. Equivalent parts can be substituted; n.c. indicates no connection. JDR = JDR Microdevices, 1850 S 10th St, San Jose, CA 95112-4108; tel 800-538-5000, 408-494-1400, fax 800-538-5005, 408-494-1420; http://www.jdr.com, Mouser = Mouser Electronics, 958 N Main St, Mansfield, TX 76063-4827; tel 800-346-6873, 817-483-4422, fax 817-483-0931; sales@mouser.com; http://www.mouser.com.

C1-C11—0.1 µF monolithic (JDR 0.1UF-MONO)
C12—220-µF, 35-V electrolytic (JDR 220R35)
DS1-DS3—LED
J1—PC mount 5-pin DIN socket (Mouser 161-0503)
J2—PC mount DB9F (Mouser 152-5609)
Q1—2N2222
R6—10-kΩ PC-mount trimmer pot
U1—*Programmed* 16F877-20P microcontroller; see Note 2.
U2—62256LP 32 kB static RAM (JDR HM62256LP-10)
U3—MAX232CPE RS-232 transceiver (JDR MAX232CPE)
U4—MX-614 modem (available from http://www.tapr.org)
U5—78L05 5-V, 100-mA positive voltage regulator
Y1—10-MHz ceramic resonator, with capacitors (Mouser 520-ZTT400MG)
Y2—3.58-MHz ceramic resonator, with capacitors (Mouser 520-ZTT358MG)
Misc: Enclosure, PC board (see Note 2), hardware, IC sockets



A top view of the KISS TNC.

converts them into standard 1200-baud Bell 202 modem tones that can be fed directly to the radio. It also converts the received tones into TTL-level signals for interpretation by the PIC.

Operating Instructions

The first time you use the KISS TNC, configure it using a computer that can run a terminal program. Attach the DB9 connector to any serial port on your computer using a standard serial cable. Set the computer's software to communicate at 1200 baud, 8 bits, no parity. In the terminal program, turn off hardware and software flow control. Install jumper W1 on the TNC's board (this places the TNC in terminal mode). Apply power to the unit. You should see a menu that looks like this:

```
Configuration Menu
1 Operating Mode
2 Set TX Delay
3 Set Terminal Baud Rate
4 Set APRS Parameters
E Erase All Parameters
Selection:
```

The first time you run the KISS TNC, it's a good idea to choose **E** to erase all parameters. Selecting item **1** allows you to determine whether the device will be operating in KISS mode or APRS Tracker mode. Menu item **2** allows you to set the TX Delay. Item **3** allows you to set the terminal data rate. This is the rate at which the TNC communicates with the attached computer both in terminal mode and when operating in KISS mode. Remember, KISS mode does not support any type of flow control. If you set the terminal rate to a level that's faster than the data can be sent over the air and then use the device to transmit large files, the TNC's memory buffer will eventually be exhausted and data lost. Thus, it's probably a good idea to leave this set at the default (1200 baud).

If you are going to use the TNC in APRS mode, you will also need to select item **4** from the menu to set the appropriate parameters for APRS. The following menu allows you to set these parameters:

```
Current APRS Parameters
1 Station Call sign: W2FS
2 First Path Call sign: RELAY
3 Second Path Call sign: WIDE
4 Third Path Call sign: WIDE
5 Icon Number: 3E
6 Alt. Icon Table? (Y/N): N
7 Delay Between Xmit (×10 seconds): 00
8 Quiet Time (in seconds): 00
9 Message Number: 1
A Set Beacon Rate: 02
B Beacon Text: This is a test of my PIC-based KISS TNC
Select Parameter to Change:
```

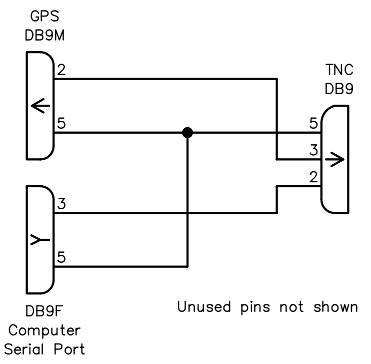


Figure 2—Using a Y adapter with the KISS TNC allows you to connect a GPS receiver and monitor incoming packets. A cheap and easy Y adapter is shown in Figure 3.

Item **1** allows you to set your call sign. It also allows you to specify the SSID to be used. The SSID should be entered as a single digit in hexadecimal format. That is, if you want the call sign W2FS-10, you should enter that as W2FS with an SSID value of A. In items **2** through **4** you can set a path that is up to three call signs long. The next two items (**5** and **6**) allow you to select the icon that is displayed when your station appears on APRS maps. Item **7** allows you to specify the length of the period between position beacons. If you set this value to zero, the unit will never beacon; it will simply monitor the frequency and display received packets. If you don't have a GPS receiver connected, set this value to zero. When the TNC is preparing to transmit a position report, it waits until it receives data from the GPS receiver. If no GPS receiver is connected, make sure that the TNC never tries to beacon, because it would wait forever to receive data from a nonexistent GPS receiver.

Table 1
KISS TNC DB9 Connector Pin-Out

Pin Number	*Function*
1	Transmit Audio
4	Receive Audio
2	Ground
5	Not Connected
3	PTT

The unit will not transmit an APRS position until the radio channel is clear. If you decide to delay transmission for a greater period after the channel is clear, you can set the quiet-time parameter to some value other than zero. Quiet time specifies the number of additional seconds that the unit delays transmitting.

Item **9** on the parameter menu specifies the message number (or "position comment" in Kenwood D700 terminology). You may select any one of the following:

0 Off Duty
1 Enroute
2 In Service
3 Returning
4 Committed
5 Special
6 Priority
7 Emergency

Menu item **A** specifies how frequently the unit should transmit the beacon text. Specifying 02, for example, indicates that the beacon text should be sent every second transmission. When values are specified as two digits in the menu, *two digits must be entered.* Thus, enter 02, not 2. Item **B** specifies what the beacon text will be.

When you are satisfied that you have specified all of the parameters correctly, remove jumper W1 and cycle the power off and on to force the unit to reboot in the proper mode.

If you want to use the unit in KISS mode, simply connect a standard serial data cable to the unit's DB9 connector (J2) and a radio connector to the DIN socket, J1. The DIN socket pin-outs are identical to those used on PacComm and MFJ TNCs, so that any existing cable of this type will work with this TNC as well. The pin-outs (in the order they appear on the connector) are shown in Table 1.

Usually the KISS program asks you to specify which TNC type you are using. This is the case with *WinAPRS*, for example. This information is required because the software needs to know which command to send to the TNC to place it in KISS mode. When using the KISS TNC, this setting is irrelevant because the unit is always in KISS mode. As a result, it doesn't matter which type of TNC you specify. There have been reports of some types of TNCs accidentally falling out of KISS mode. This should never be a problem with this TNC because KISS is its primary operating mode.

If you want to use the device to monitor UI frames, but don't intend to connect a GPS receiver, you can use the same hardware configuration that you used for KISS mode. Just remember to set the mode to APRS and to set the delay between transmissions to zero. If you want to connect your GPS unit to the KISS TNC, but don't care to connect a terminal to monitor incoming packets, you will need to connect the GPS unit to the TNC's serial port through a null-modem adapter. If you want to connect the GPS unit and also monitor incoming packets, you'll need to construct a simple **Y** adapter as shown in Figures 2 and 3. The adapter routes the GPS transmit-data line to the TNC's receive-data line and the TNC's transmit-data line to the computer's serial port receive-data line.

Construction Tips

I've built several of these units on pieces of RadioShack perfboard using point-to-point wiring, but this construction approach is rather difficult because there are quite a few wires interconnecting the PIC with the static RAM chip. Consequently, a substantial potential for wiring errors exists. A PC board is available for this project, as is a kit of parts (see Note 2). The PC board requires the installation of several wire jumpers. To ensure that the board works properly, install all of the parts, including the IC sockets, but *do not* install any ICs. Then, carefully observing proper polarity, apply power and ensure that approximately 5 V is present between the following pin pairs:

U1 socket pins 11 and 12 (pin 12 is ground)
U1 socket pins 31 and 32 (pin 31 is ground)
U2 socket pins 14 and 28 (pin 14 is ground)
U3 socket pins 15 and 16 (pin 15 is ground)
U4 socket pins 8 and 16 (pin 8 is ground)

When you're sure there are no cold solder joints or solder bridges, install the four ICs and apply power. Total TNC current consumption is less than 20 mA; most of that is consumed by the LEDs. You can reduce the current consumption by increasing the value of the current-limiting resistors (R10-R12), or you can omit the LEDs altogether. In this case, it should be possible to run the unit for several days on a 9-V battery.

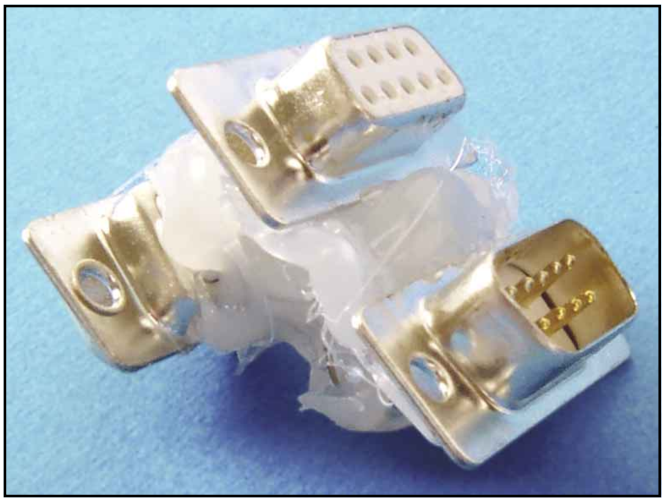


Figure 3—This simple Y adapter consists of one female DB9 socket interconnected with two DB9 male jacks, all held together with noncorrosive sealing compound.

Summary

PIC microcontrollers have rapidly become a staple of Amateur Radio construction projects. This project demonstrates one of the reasons that this has occurred. PICs are extremely flexible devices that can perform the functions of ICs that are considerably more expensive and consume much more power. We are just beginning to explore the range of amateur applications for these inexpensive, powerful devices.

Notes

[1]The APRS Mic-Encoder format compresses the APRS position report and message bits into the destination address and information fields of a standard AX.25 UI frame. For more information, see **http://www.tapr.org/tapr/html/mic-e.html**.

[2]A complete kit of parts including the PC Board, a *programmed* PIC16F877 and all other parts (except an enclosure) are available for $65 from John Hansen, W2FS, 49 Maple Ave, Fredonia, NY 14063. A *programmed* PIC16F877 only is available for $35. A wired and tested board is also available for $90. PC boards for this project are available from FAR Circuits, 18N640 Field Ct, Dundee, IL 60118-9269; tel 847-836-9148 (voice and fax). Price: $8 plus $1.50 shipping for up to four boards. Visa and MasterCard accepted with a $3 service charge. You can download th es ource code (KISSTNC.ZIP) for this project at **http://www.arrl.org/files/qst-binaries/**.

49 Maple Avenue
Fredonia, NY 14063
john@hansen.net