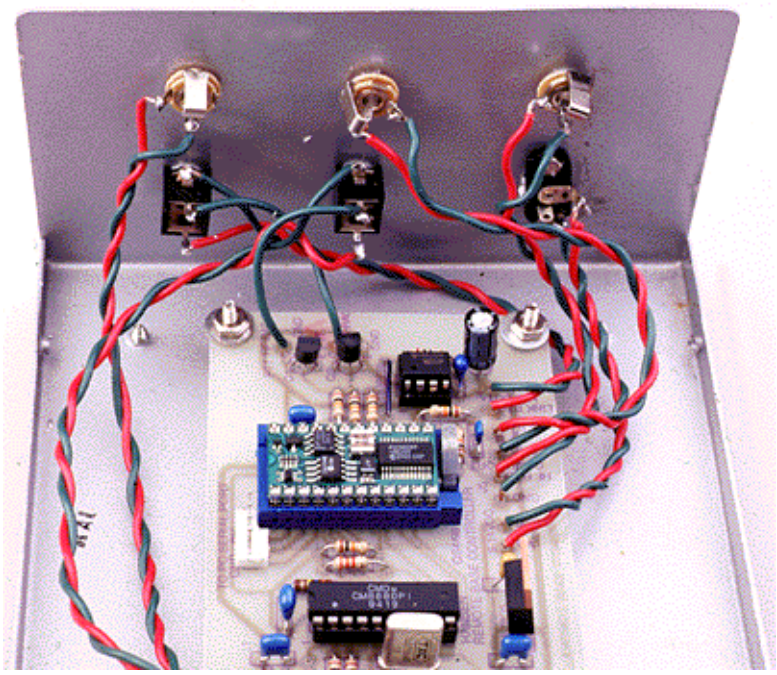
An Inexpensive, Remote-Base Station Controller Using the Basic Stamp

Now *you* can operate your home station while you're in your car, or at another location. Clubs can set up remote HF stations for all members to use, too!

By John Hansen, W2FS



An inside view of the Remote-Base Station Controller. In this unit, a ZIF socket is used for U2 to allow easy replacement when experimenting.

Amateur Radio operators have been using remote bases for years. A *remote-base station* allows you to access most of the functions of an HF radio from a remote location using either a portable (H-T) or mobile link radio. To use a remote base, you need an *HF radio*, a *link radio* (to link to the remote user—generally this is done on UHF) and a *remote-base controller* to act as the glue between them.

A few remote-base controller interfaces have been commercially available in recent years. This controller differs fundamentally from them in two respects: First, it uses an inexpensive, user-programmable microcontroller called a *Basic Stamp* to control the HF radio. This helps keep the project's cost low. Secondly, it controls the HF radio via the radio's standard computer connector. In the case of my ICOM IC-706 transceiver, this is the ICOM CI-V interface. This greatly expands the capabilities of the interface, because the HF radio's frequency, mode, etc, can be read directly from the radio and transmitted to the remote user via Morse code.

Functional Overview

To gain an appreciation of how this interface unit works, refer to **Figure 1** while I describe a typical operating session. Using my H-T, I begin by keying in a two-digit password that activates the system. (The password is used to turn on the system and prevent unauthorized users from accessing it.) After entering the two-digit code, I can hear the HF rig's receive audio coming through. On a typical afternoon, I'll want to operate on 20 meters. I enter a D from the H-T's DTMF keypad, followed by 14*16 and another D. The Ds tell the controller that I want to *directly* enter a frequency, and 14*16 means *go to* 14.16 MHz. The microcontroller's firmware is smart enough to sense whether I mean kilohertz or megahertz. That is, it knows 14.16 means megahertz, but if I enter 7150, it knows to use kilohertz. Similarly, the controller assumes a number such as 146.625 means megahertz. You need enter only the nonzero portion of

the frequency. For example, if I had entered D14D, the radio would have gone to 14 MHz, the bottom of the 20 meter band. This allows for rapid band changes.

Figure 1—Remote control commands issued via the DTMF keypad.

Key	*Function*
1	Tune down one frequency step.
2	Tune up one frequency step.
3	Transmit the Hertz portion of the HF radio's frequency in Morse code.
A	Start the HF radio tuning up the band.
4	Turn off the link, ID and return to standby.
5	Key the HF radio's PTT line.
6	Transmit the kilohertz portion of the HF radio's frequency in Morse code
B	Go to a specific radio memory. For example, **B12** tunes the radio to memory 12.
7	Force the link to transmit the ID in Morse code.
8	Step through the following modes: USB, LSB, AM, FM.
9	Transmit the megahertz portion of the HF radio's frequency in Morse code.
C	Change a portion of the HF radio's frequency. For example, ***C3****XXX* changes the Hertz digits to *XXX*, ***C6****XXX* changes the kilohertz digits to *XX*, ***C9****XXX* changes the megahertz digits to *XXX*.
*	Change the HF radio's step rate: *1=1 Hz; *2=10 Hz; *3=100 Hz; *4= 1 kHz, *5=10 kHz, *6=100 kHz; *7=1 MHz, *8=10 MHz.
0	Transmit the HF radio's mode in Morse code.
#	Transmit the entire HF radio frequency in Morse code.
D	Set the HF radio frequency directly: **D14*235D** sets the radio to 14.235 MHz. *is used as a decimal point after the megahertz digits. The frequency can be set in 1-Hz increments. Any digits omitted at the end of the string are assumed to be zeroes. For example, **D7D** sets the radio to 7.0 MHz.

Once I have the radio on 14.16, I want to tune up the band and see what's on. This is accomplished simply by transmitting an A from the DTMF pad. The HF radio starts scanning up the band just as if I was at home turning the main tuning knob. When I hear a station calling CQ, I stop the scan by pushing any key on the H-T's DTMF pad (with the PTT button engaged, of course). Sometimes, fine tuning is necessary. The pad's 1 key moves the HF VFO down one step, and the 2 key moves it up. The default step size is 100 Hz, but it can be changed to anything from 1 Hz to 10 MHz, by pushing the ***** key followed by the digit representing the step size: 1 =

Hz; 2 = 10 Hz; 3 = 100 Hz, etc.

When I have the station tuned in, I might want to know what frequency it's on. If I push the # key, the remote base transmits the radio's complete frequency in Morse code. However, I knew I was on 20 meters, and if I want to know only the *kilohertz* digits, I press the 6 key. The controller then transmits only the three digits corresponding to the kilohertz part of the frequency. Pressing 3 gives me the *hertz* digits; pressing 9 tells me the *megahertz* digits.

When the station I'm interested in finishes calling CQ, I press the 5 key in the middle of the key pad to engage the HF radio's PTT. Then, anything I say into the H-T is retransmitted on HF. When I'm done with my transmission, pressing any key turns off the HF transmitter. If my H-T battery dies in the middle of a transmission, a built-in watchdog timer automatically unkeys the HF radio after approximately three minutes. When I'm done with my operating session, pressing the 4 key forces a CW ID on the link frequency, then switches the system to standby.

Operating from a remote base is about as easy as it is when I'm sitting at home in the shack! In some ways, it's even easier. The ICOM IC-706 has no built-in mechanism for direct frequency entry, yet this *is* possible when operating through the remote base with my H-T. I recently operated the European Field Day contest with my H-T and worked 10 countries in about a half hour.

The remote base allows you to shift the HF radio's frequencies, change modes, access the radio's memories, scan the band, etc. Using Morse code, the remote base keeps you informed of the HF radio's operating frequency and mode. To keep your operation legal, the remote base also remembers to transmit your ID on the link frequency in Morse code every nine minutes. As an added benefit—because the microcontroller in this unit communicates using 5 V TTL-level signals—you can run its computer output directly to a '706 (or other ICOM radio; more on this later) *without purchasing* a computer interface level converter (such as the CT-17 CI-V level converter unit). Thus, most of the cost of this project can be recouped in savings by not having to buy a level converter!

Basic Stamp Overview

The heart of the remote-base controller is a Basic Stamp II microcontroller. I'm an economist, not an engineer, and this is the first piece of equipment I've ever designed myself. That, in itself, gives you a clear idea of just how *easy* it is to work with the Basic Stamp microcontroller! Those who argue that the advent of highly integrated circuits sounded the death knell of home design and construction have simply never worked with the easy-to-use digital and RF building blocks that are now available for so little money. This circuit is a case in point.

The Basic Stamp II (BS2) is produced by Parallax, Inc. [**1**] It is a general-purpose micro-controller based on the PIC16C57 IC. (For a comparison of the Basic Stamp to a PIC, see the sidebar, "**Basic Stamps and PICs—What's the Difference?**" The BS2 can easily be programmed using the BASIC computer language. The BS2 retains its program even when its power is removed. Turning on the power merely causes the BS2 to automatically restart the program you've stored in it. The BS2 is programmable (and reprogrammable) without any special-purpose hardware—you simply hook the chip to your computer's serial port. You don't need a UV eraser to erase the BS2, either. Loading a new program automatically overwrites the existing program. The BS2 can be reprogrammed up to *10 million times*, so you can feel free to load a draft of your code just to see how it runs.

The BS2 has 16 pins that can be used as inputs *and* outputs. Inputs might include such items as pushbuttons, potentiometers, logic circuit outputs, various kinds of sensors (temperature, pressure, etc), or even serial data from a computer (or the computer port of a radio). Outputs can be transistor switches, relays, LEDs, alphanumeric displays, audio—even serial data streams. The BS2 simply takes the information provided to it by the inputs, processes it and performs some action on the outputs.

Here's How

An example may help to clarify how this process works. With my remote-base controller, I push the # key when I want to know which frequency I'm on. When I push the H-T's # key, the audio heard by the link receiver is routed to a DTMF decoder. That IC sends the BS2 a number indicating which key it heard (# = 12). The Stamp knows that this is a request for the current operating frequency, so it then sends serial data on an output pin to my IC-706 that asks what frequency it is on. The Stamp then *changes that pin from an output to an input*, and receives the incoming data from the '706. Another program section converts this data into a string of numbers representing the frequency. These numbers are then routed to a Morse code section of the program that transmits each number in Morse as audio on another output pin. That output is then fed to the link transmitter and I hear the transmission on my H-T! Complicated? Not at all!

You can perform this type of programming with little investment of time and money. The programming software is *free* at Parallax's Web pages (**http://www.parallaxinc.com**). At the same location, you can pick up a copy of the BS2's manual in Adobe Acrobat format. This extraordinarily good manual will have you up and running quickly. It is, however, a very long document, so have

plenty of paper on hand if you plan to print a copy!

So, to get started with the BS2, all you need to buy is the Stamp itself (about $50) and a board to hold it. Parallax sells prototyping boards for about $20, but I bought a multipurpose PC board at Radio Shack (276-150) for $1.19, and a 28 pin IC socket (276-1997) for another 99 cents. (PC boards are available for this project.) [**2**] You need to power the circuit, of course, and for experimental purposes, a 9 V battery works just fine. You also need to build a cable to hook the chip to your computer's serial port. Diagrams for doing this are also available at the Parallax Web site.

If you don't know anything at all about computer programming, but want to learn, I'm hard-pressed to think of a better place to start than by programming the Basic Stamp.

Circuit Description

I got the idea of building a remote base controller using the Basic Stamp when I discovered that Scott Edwards Electronics had developed a DTMF transceiver application kit to go with the Stamp. [**3**] This kit provides a California Micro Devices CM8880 DTMF transceiver chip, along with complete documentation for interfacing it to the Stamp or other PIC microcontroller, including circuit diagrams and sample programs using the device. (See the sidebar "**Why I Chose the CM8880**.") Combining the CM8880 and the Stamp allows you to remotely control virtually *anything* from your H-T. The Stamp, for example, has an X-10 home automation support built into it, so you could use such a system to remotely turn appliances on and off.

The major components of the circuit (see **Figure 2**) are the CM8880 DTMF transceiver, U1, and the Basic Stamp, U2. A 555 timer (U3) keeps track of when it's time to ID. U2 pin 12 triggers U3, which causes U2, pin 17 to go high for about nine minutes. Every couple of seconds, U2 checks to see if pin 17 is still high. If it isn't, U2 IDs and triggers pin 12 again.

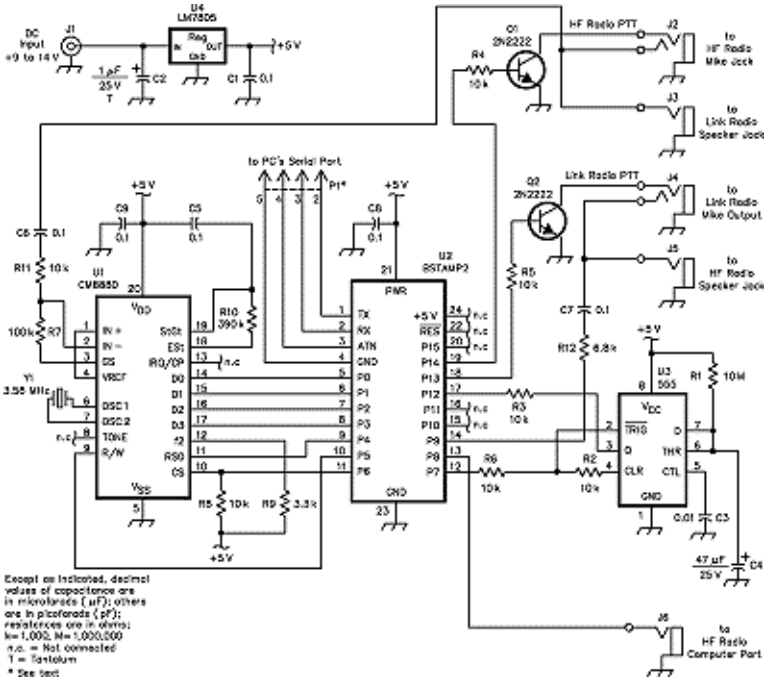


Figure 2—Schematic of the remote-base controller circuit. Equivalent parts can be substituted. Unless otherwise specified, resistors are 1/4 W, 5% tolerance carbon-composition or film units. Part numbers in parentheses are Radio Shack. Observe proper polarity for polarized components.

C2—1 µF, 25 V tantalum

C4—47 µF, 25 V tantalum or aluminum electrolytic

J1—Dc input connector (RS 274-1565)

J2, J4—1/8-inch, 3-circuit normally open phone jack (RS 274-351)

J3, J5, J6—1/8-inch, 2-circuit normally open phone jack (RS 274-249)

P1—4-pin SIP header.

Q1, Q2—2N2222 (RS 276-2009)

U1—CM8880 DTMF decoder (see text and Note 3)

U2—Basic Stamp II (see Note 1)

U3—555 timer (RS 276-1723)

U4—7805 5 V, 1 A regulator (RS 276-1770)

Y1—3.58 MHz color-burst crystal (RSU 11321437)

Misc: PC board (see Note 2), hardware, IC sockets, enclosure, mating connectors and cables.

Q1 and Q2 key the HF radio PTT lines (via pin 19) and the link radio (via pin 18). Y1 can be a common TV colorburst crystal. Although U2 has an on-board 5 V regulator that can provide 5 V to external circuits, it doesn't have the current capacity to power U1. As a result, I added a 7805 regulator, U4. The current through U4 is sufficiently small that no heat sink is required.

The circuit is simple because most of the logic that you might expect to see in a remote-base controller is implemented not in logic ICs, but in the Basic Stamp firmware that recognizes inputs and responds with changes in outputs. The inputs for the Basic Stamp are the connections to the DTMF transceiver (that decodes the tones into numbers on pins 5-11), the input from the timer (that tells the Stamp via pin 17 when to ID) and the data connection to the radio (pin 13). Because the Stamp is doing nothing else after you key the PTT line on the HF radio, even the watchdog timer can be implemented entirely in software!

A second watchdog timer implemented in software times out the link radio. Although the link radio IDs every nine minutes, if your H-T battery dies, you may wish you had turned the link transmitter off! A timer is used to turn off the link if it hears nothing from the H-T in any 10-minute period. Every time you send a DTMF tone to control the HF radio, this timer is reset.

U2's outputs are the two PTT lines (pins 18 and 19), the line to trigger the timer circuit (pin 12), the audio output for the various CW functions (pin 14) and the serial link to the radio (pin 13). Because the '706 sends and receives serial data on the same line, it is fortuitous that the Stamp is able to switch pin 13 between being an input and being an output—only *one pin* is required for all communication to and from the radio!

Construction Notes

I built my remote base in a standard Radio Shack project box (RS 270-274) though any box suitable for mounting the PC board would work well. The programming connector (P1) is required only when you're programming U2. Therefore, I saw no need to provide a permanent panel-mounted connector. Instead, I installed a 4-pin SIP header. I connect an appropriate interfacing cable between the header and computer when needed. Of course, the enclosure must be open to do this. If you are using the pre-programmed Stamp (see **Note 4**), you need not install the programming header.

Use sockets for the ICs. Over 90% of the cost of this project is tied up in two chips. Be *very careful* to ensure they are wired properly before you apply power! Before installing the chips, wire everything else, then conduct a number of tests. Apply power to the voltage regulator and make sure that you measure +5 V on pin 20 of U1's socket and pin 21 of U2's socket. Then, remove power and check each adjoining pair of pins of each IC socket to make sure that there are no solder bridges. Using your multimeter, you can check your wiring against the schematic to ensure the connections are right. Check and *double-check* everything before installing the ICs and again applying power.

Once the wiring is done, you need to load the firmware into the Stamp. This is not a difficult process and is explained in detail in the Stamp manual. If you prefer to not do the programming yourself, preprogrammed chip sets are available. [**4**]

You could add potentiometers to the audio lines to adjust the audio levels going to each transmitter. I found that using the radio's volume control to adjust the audio coming into the device works well, so I did not add audio gain controls to the circuit. You may find it necessary to change the value of R12, however. It controls the level of the Morse code audio from the Stamp. A value of 6.8 kΩ works quite well in my station, though this is partly a matter of personal preference. Increasing the value of this resistor reduces the volume of the Morse code output.

Using the Remote Controller with Other Radios

The software I wrote for the remote-base controller is intended to work with the ICOM 706, and this is the only radio that has

currently been tested with it. To use it with the ’706, set the following IC-706 menu parameters:

10	SCAN SPEED	LO
21	CI-V ADDRES 48H (Note: The default setting on the ’706 MKII is 4EH, but can be changed to 48H)	
22	CI-V BAUD	Auto (or 1200)
23	CI-V TRN	On
24	CI-V 731	Off

The controller should also work without modification with any ICOM radio that uses five bytes to communicate its frequency, if the address of the radio can be changed to $48 (48 hexadecimal). If the address cannot be changed, only a minor modification of the program is necessary to set the program’s address to that of the radio. Using this technique, it should be possible to get the controller working with the new IC-756 and most other ICOM radios that are controllable through the CI-V interface. The program will *not* work without modification on ICOM radios that use *four* bytes to communicate their frequencies—these include the IC-731 and IC-735. However, relatively minor modifications to the program (simply using four bytes instead of five to represent the frequency) should allow these radios to work as well.

Depending on the capabilities of the radio’s computer interface, it is probably possible to modify the program to work with radios other than those manufactured by ICOM. The communication functions that are used by this program are:

- Get the radio’s operating frequency
- Get the radio’s mode
- Set the radio’s frequency
- Set the radio’s mode
- Start scanning
- Set the radio’s memory

These fairly basic functions are likely to be available on most computer-controlled radios.

Software Modifications

The Basic Stamp Remote-Base Controller source code is not in the public domain and is not shareware. It is available, however, free of charge for use by individual hams for constructing a single unit for their own personal use. If you decide to program your own Basic Stamp for this project, at a minimum you will need to change the call sign in the software so that it uses your call rather than mine. You may also want to change the password that brings up the system and sets the duration of the watchdog timers. If you are using a radio other than the IC-706, you will probably need to change the radio’s address in each of the program statements that begin with SERIN or SEROUT. Instructions for making both of these changes are embedded in the source code itself, which is available at the ARRL’s ftp site. [**5**]

Summary and Notes

Simply because you have the technical capability to listen throughout the range of IC-706’s receiver does *not* mean that you can legally retransmit everything you hear! Your best bet is to stay in the ham bands. Transmission of music, broadcasting, etc, are, of course, still against the law, even when you are using a remote base. This seems obvious, but operation with the Basic Stamp remote base is so transparent that sometimes you forget that you are not actually sitting in front of the HF radio!

Pay close attention to the regulations concerning remote control of Amateur Radio stations. You must communicate with your base station (in *both* directions) on a frequency above 222.15 MHz.

I have run into a problem with ground loops on the base-station side when the HF radio and the link radio were connected to the same power supply. In my station, I solved this problem by using an isolation transformer in the audio line between the link receiver and the remote-base controller. If you run into this problem, you can solve it either by using separate power supplies for the two radios, or by experimenting with audio isolation transformers. If you do have this problem, and you have a couple of miniature 8Ω to 1 kΩ audio transformers, wire them back-to-back to construct an isolation transformer.

Be careful with your selection of a link radio. The entire time you are listening on the HF bands with your H-T, the link radio is transmitting. Choose a radio and power level that accommodate extended key-down periods.

You can view the Basic Stamp code that I’ve written as a starting place for adding your own functionality to the unit. Although

there is a limit to the overall size of the program that can be put into the Stamp, you can remove some of the capabilities that I've incorporated and write code that implements features you prefer. You could, for example, have a one-keystroke command to go to a particular net frequency or a frequency on which you have a regular schedule. Or, write code to use both the A and B VFOs so you can operate split. Furthermore, there are still three unused I/O pins on the Stamp. You could use one pin to control an antenna switch. Or, use two pins to operate your station's antenna rotator directional controls and use the third pin to provide feedback of the rotator's direction, which could then be relayed to you remotely in Morse code. It would even be possible for you to enter the azimuth in degrees via the DTMF pad and have the Stamp automatically guide the rotator to the desired direction. What you can do with the Basic Stamp is limited mostly by your own imagination. I have made source code available so that you can perform modifications to it without having to rewrite all the control routines from scratch.

Formerly licensed as WA0PTV, John Hansen, W2FS, has been an Amateur Radio operator since 1966. His interests include satellites, digital communication and "homebrewing." For a number of years, John was the editor of *The AMSAT Journal*, and is the 1993 recipient of the ARRL Atlantic Division Technical Achievement Award for his work on digital satellite gateway software. John maintains a career as a Professor of Economics at the State University of New York to provide enough income to purchase radio equipment and Basic Stamps. You can contact John at 49 Maple Ave, Fredonia, NY 14063; e-mail **hansen@fredonia.edu**.

Basic Stamps and PICs—What's the Difference?

In recent months, a number of *QST* projects have featured Microchip Technology Inc PIC microcontrollers. These chips are quite inexpensive, yet are a rather powerful means of building stand-alone computing capabilities into your projects. The Basic Stamp—used in this project—is actually just a PIC itself, coupled with a proprietary BASIC language interpreter that allows it to understand BASIC commands and be easily reprogrammed. So, you might ask, when should I use a PIC and when should I use a Stamp? The differences between the two include price, ease of use and power.

Price

The cheapest Basic Stamp costs more than all but the most-expensive PIC. The Stamp used in this project costs just under $50, while PICs generally cost less than $10. If you are building only one copy of a particular item (especially if the overall cost of the project is significant) this price difference may not matter. But if you are making many units, or the overall cost of the project is otherwise small, the savings may be worth the relative inconvenience of using a PIC.

Ease of Use

The biggest advantage of Basic Stamps is that they are very easy to use. They can be programmed in BASIC by simply using a serial port cable—no hardware programming device is needed. You can make changes to your code and quickly load it into the Stamp for testing. The development tools are free for the downloading.

PICs, on the other hand, are most often programmed in assembly language, which is significantly harder to learn than BASIC. Some excellent compilers are available that allow you to program PICs in languages such as C, but none is pubic domain or shareware. Programming a PIC requires that you buy or build a PIC programmer—a hardware device that attaches to your PC to burn the code into the chip. Loading your code into a PIC also takes substantially more time with a PIC than it does with a Stamp, making it more difficult to make changes to your program and see the results. Furthermore, for any of the PIC chips other than the 16C8/16F8 family, you will need an ultraviolet-light EPROM eraser to erase the old code from a PIC before you can put in the new code. This adds another three to four minutes to the programming cycle.

Power

PICs run substantially faster than Basic Stamps. Although the Basic Stamp has a PIC as its processor, the instructions that you write are stored in an EEPROM. The PIC chip is programmed with the BASIC interpreter. Each instruction is read out of the EEPROM by the embedded PIC, processed by the interpreter, and then performed by the PIC. While this is about a thousand times slower than having your code reside in the PIC itself, for many applications (including the remote base controller) you won't notice the difference.

PICs are also available in a wide variety of flavors. Some have really large numbers of I/O pins (more than twice as many as a Stamp). Some have on-board analog-to-digital converters, while others have built-in serial ports. Some have a huge programming space (up to 64 kbytes). Thus, you can select the particular PIC whose features match your needs.

Basic Stamps are an excellent way to learn about the capabilities of embedded microcontrollers and to rapidly develop useful

applications. If you find it to be fun and rewarding, it may be worth your while to consider moving to PICs.—*John Hansen, W2FS*

Why I Chose the CM8880

Because the pin-outs and external component requirements for the MT8870 DTMF decoder are so similar to those of the CM8880, I initially assumed the only difference between the two was that the CM8880 is a transceiver and the ’8870 is only a receiver. An evening’s experimentation and documentation review showed me otherwise. Now I understand why the CM8880 costs more than the ’8870, and why Scott Edwards selected it for his DTMF AppKit for the Basic Stamp.

The ’8870 does just what it’s supposed to do: It receives DTMF tones and latches them on four output pins, just like the ’8880 does. However—the only way to reset the latches is for the ’8870 to either receive another tone, or to wait a specified period of time. The ’8880 has an internal register that can be read, and when it’s read, it’s reset, so you can essentially clear the output pins using one pin of the Basic Stamp.

As a result, for single tones, the ’8870 works great. It will also work okay for tone sequences—if you carefully time how fast you push the keys that make the tones. There may even be some situations where the ’8870 is preferred. If you hold down a key for a long period of time with the ’8870, it continues to send the same tone. This is rather like the continuous-fire button on a joystick. The ’8880, on the other hand, interprets this as *one keypress*, unless you let up and push the key again. For the remote-base project, the latter behavior is clearly preferred. The ’8870 is inappropriate for use in this project because it relies so heavily on multiple key sequences (such as keying in a frequency string).

Now I don’t feel like I’m being cheated by paying $25 where I thought a $2.50 part would work just as well!—*John Hansen, W2FS*