Build an Intelligent Relay

A quick-to-make microprocessor-based device, controlled by your HT.

by Bruce R. Knox N8LXS

If you want to check out a length of coax, remotely reset a cluster or BBS node, or test a doorbell installation, you often get someone's help. Wouldn't it be nice to have a device that could provide you with a contact closure, at your command, using your hand-held transceiver (HT)? This article will show you how to build such a device, which provides you with a set of functions that will make your life a whole lot easier in these situations.

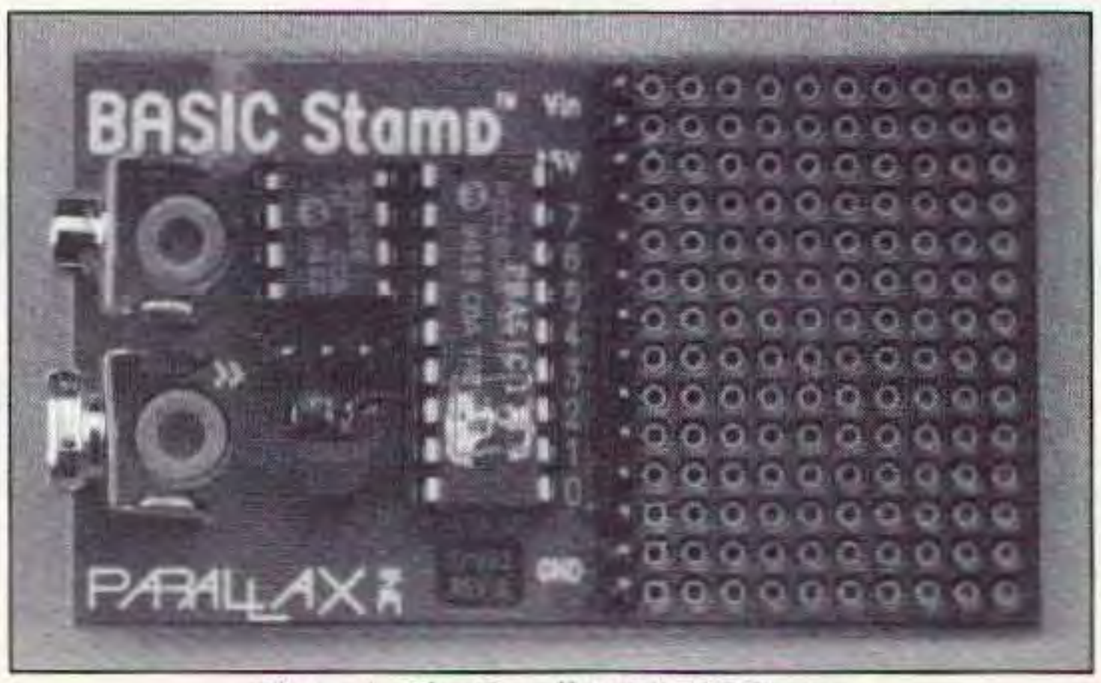

Photo A. The Parallax BASIC Stamp.

You Need a Microprocessor

In order to provide a set of useful functions, some form of intelligence would be required for this device. I looked over the range of microprocessors available and most seemed to be gross overkill for such a simple application. In most cases, even simple single-chip microprocessors like the Motorola 68HC705xx require a multi-hundred-dollar development environment and a lot of construction to gain even basic functionality.

Enter a nifty product from Parallax, Inc., called The BASIC Stamp. This is a simple-to-use but highly functional microprocessor that fits this (and many other) applications perfectly.

What is The BASIC Stamp?

The BASIC Stamp is a PIC microprocessor mounted on a small printed circuit board (2.5" x 1.5", or 6.3cm x 3.8cm). It sports a 5-volt regulator, reset circuit, resonator, input/output (i/o) header for the eight i/o lines, and a 10 x 14 wirewrap/prototyping area. The folks at Parallax have developed code for the PIC microprocessor that interprets "tokens" generated by a BASIC language tokenizer that runs on a standard PC. This enables you to develop applications in a high-level language, download to The Stamp, and run. It runs your instructions at about 2,000 per second (50 microseconds per instruction, typically), and the supplied EEPROM (Electrically Erasable Programmable Read Only Memory) can hold about 80-100 BASIC instructions per program.

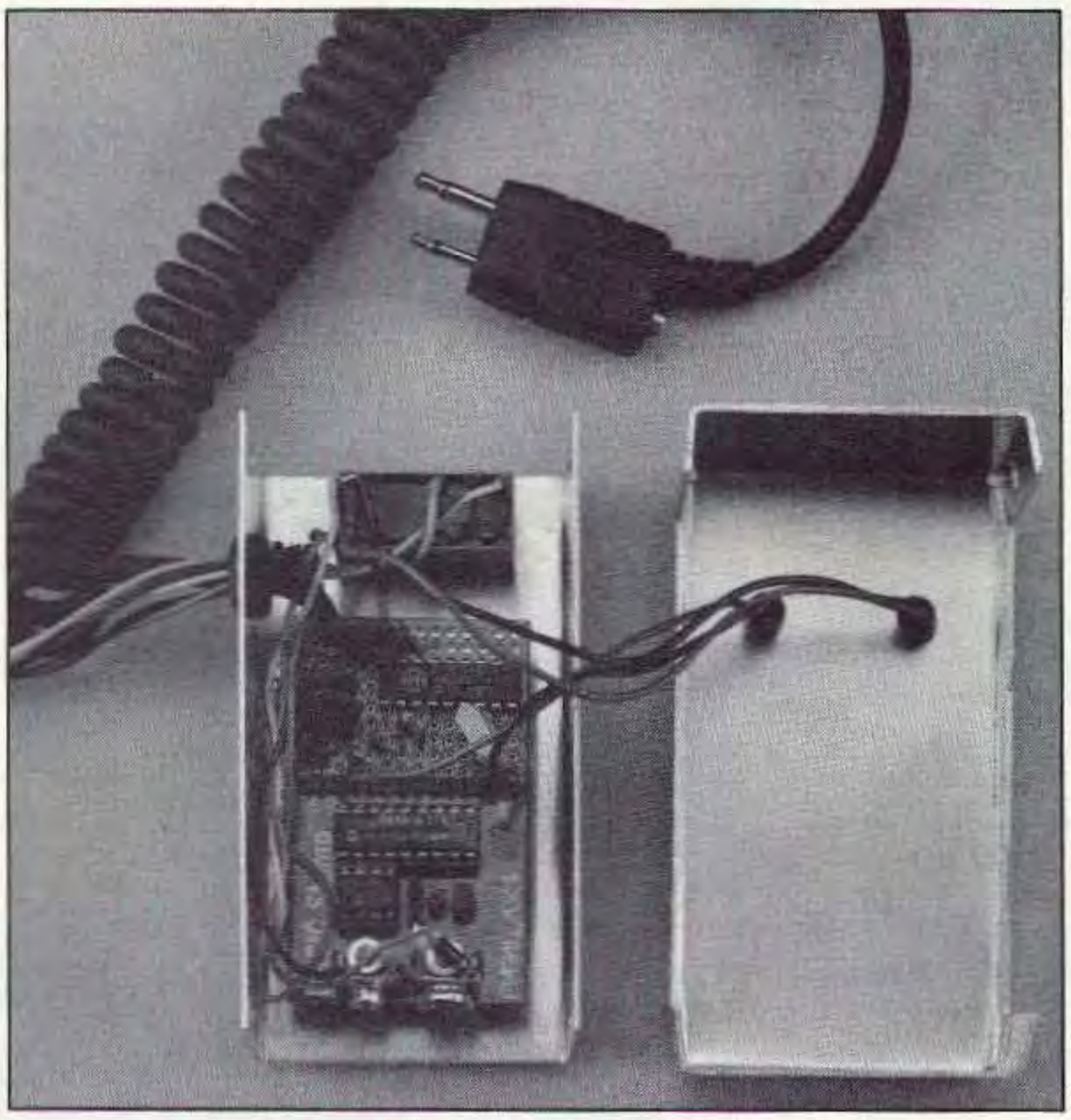

Photo B. The finished Intelligent Relay.

Figure 1 shows a brief summary of some of the more interesting Stamp BASIC commands. While the program speed/size limitations of this device will most likely prevent you from doing many digital signal processing applications on it, there are many others you will be able to do, and in a fraction of the time it might take you with a more conventional microprocessor environment. The best part is that the development environment that runs on your PC costs only $99 from Parallax, Inc., and you can use it over and over again. Stamps cost $39 in unit quantities, and discounts start at five units.

The "Intelligent Relay"

With this handy building block in hand, it was pretty easy to develop the application. Figure 2 shows the schematic for this application. It was built entirely on the prototype area of The Stamp using point-to-point construction techniques. The

basic hardware requirements for this device are:

1. Function from 12 VDC (The Stamp comes this way); 2. Provide a normally open and closed relay output; 3. Provide a DTMF tone decoder to receive commands; 4. Provide a means of transmitting tones from the HT If you are using a remote switch to check out an installation—it is really nice to know if, in the case of an unexpected result, your remote switch is working. To accomplish this, I made a connection to the microphone input of the HT. Through this connection, each command can be acknowledged by transmitting a Morse code "R." In addition, an "O" (a shortened version of "0") can be sent for off, an "A" (a shortened version of a "1") for on, and a callsign for identification purposes. The interface to the HT was implemented using capacitor C2 for the audio tone, and resistor R1 for keying the HT. You may find that you have to adjust the value of R1 or C2 depending on the HT you use. If the HT you choose does not support this hardware arrangement (keying and audio on the same wire), you can easily adapt it to support a relay for the keying and keep the tone generation output separate. Tone generation is handled by a command called "SOUND" in The Stamp's BASIC interpreter, and keying of the HT is handled by a simple logic level command (see the software listing).

Receiving DTMF (Dual-Tone Multi-Frequency) tones is pretty easy these days. A Teltone M-8870 DTMF receiver was chosen for this purpose. The M-8870 (U2) uses a minimum of external parts and runs off a single 5-volt DC supply. The outputs of the M-8870 consist of four data bits that tell you which tone was most recently decoded, and a strobe line that goes to 5 volts when a tone is present. These five lines are run to five of the eight lines on The Stamp. The M-8870 requires a cheap 3.58 MHz crystal (sometimes called a colorburst crystal), three resistors and two capacitors to get it running. In this application, none of the values (except the crystal value) are critical, with a 20% variance being acceptable.

The output from the device is a relay, designated as K1. This is a 12 volt DC relay that is powered from the external supply (before the on-board 5-volt regulator). Resistor R2 limits the base current to transistor Q1, and standard NPN, such as a 2N2222 or 2N3904. When saturated, Q1 energizes K1. Diode D2 is provided to snub the turn-off spikes generated by K1.

Diode D1 was provided for two reasons. The first and most important reason is to protect the device from a potential reverse polarity connection. The other reason is to introduce a volt or two of drop to limit power dissipation in The Stamp's regulator. Capacitor C1, a 0.01 disc, is provided to help keep RF out of the device.

As mentioned above, point-to-point construction techniques were used for this device. Photo B shows the finished unit is a small aluminum enclosure. Small, solid, insulated wire can be used, and the circuit can be built in about two hours. One word of caution: The column of holes in the prototyping area right next to the header are connected to the header! Use these holes to pick up your signal connections to the other parts of your circuit. Finally, a spare coiled cable was available for the device-to-HT connection. If you don't have one of these available, shielded cable and separate connectors will work just fine. The block diagram in Figure 3 shows you how the system is used.

```
Program Control

IF... THEN
GOTO
GOSUB
FOR... NEXT

Numbers

Integer math: +, -, *, /, etc.
Logicals: AND, OR, XOR, etc.

Digital I/O

LOW         set pin low
HIGH        set pin high
TOGGLE      change state of pin
PULSIN      measure pulse width
BUTTON      do button functions

Serial I/O

SERIN       Async serial input
SEROUT      Async serial output

Analog I/O

PWM         Generate PWM output
POT         Measure value of a potentiometer

Sound

SOUND       Generate tones/white noise on a pin

There are more commands.
```

Figure 1. Some of The Stamp's commands. (See Figure 4 for others.)

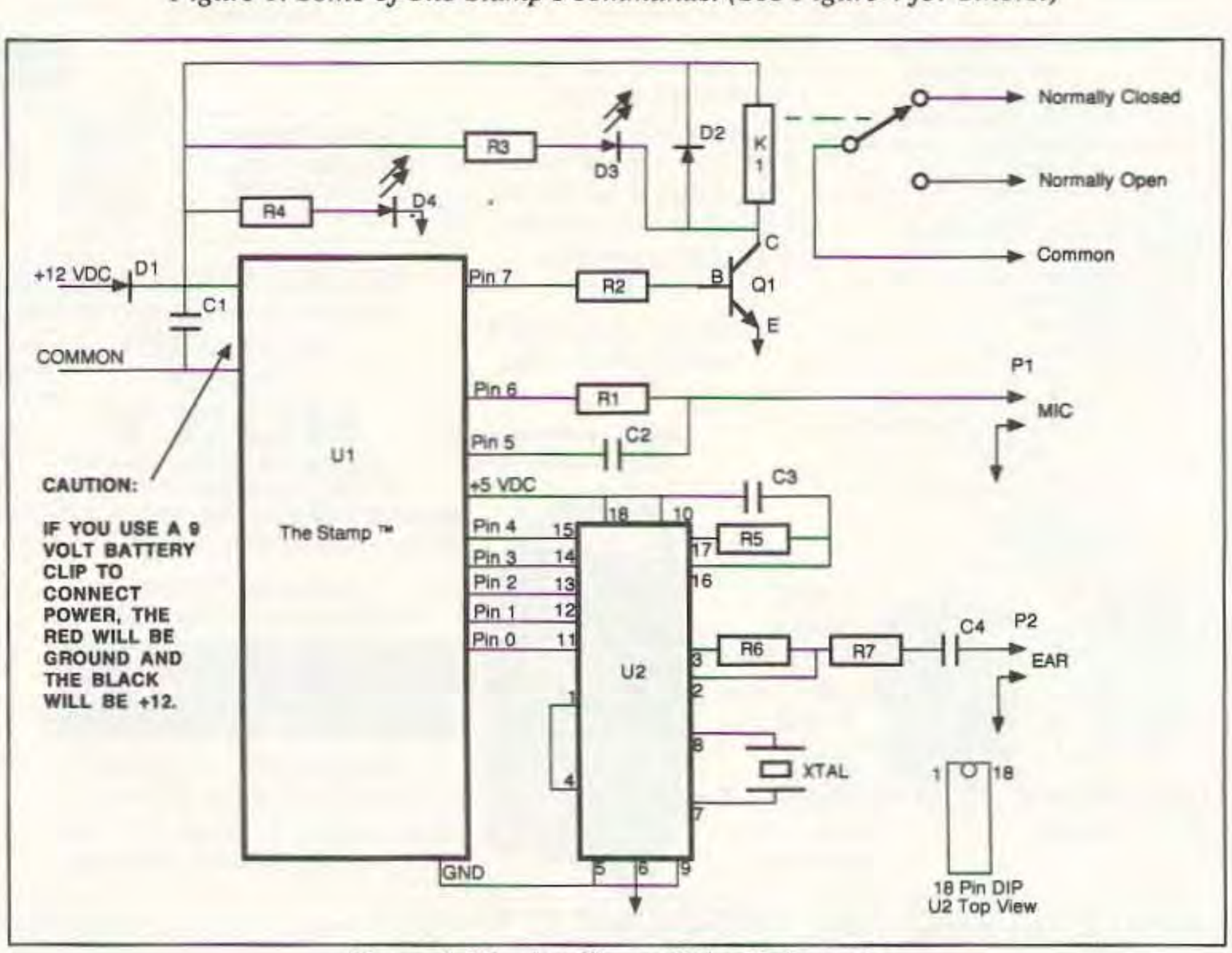

Figure 2. The Intelligent Relay Schematic.

The Software

Figure 4 shows the complete listing of the Intelligent Relay Program. The only reason

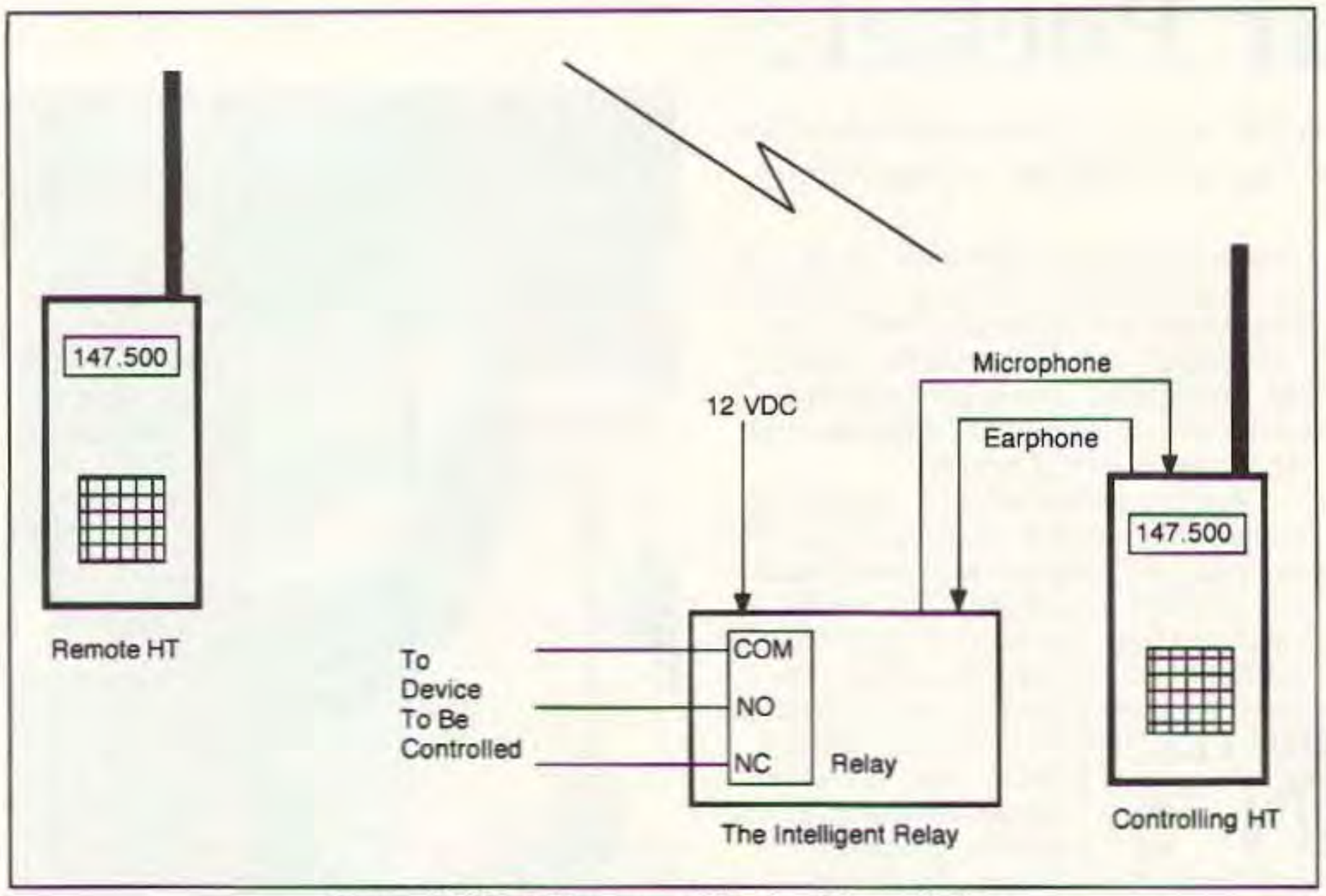

Figure 3. Block diagram of the Intelligent Relay.

the program looks so long is that it is heavily commented. The application performs the following functions:

DTMF 1: Turn relay on for 2 seconds, send "R"
DTMF 2: Pulse relay for x times (digit after 2), send "R"
DTMF 3: Turn relay on, send "R"
DTMF 4: Turn relay off, send "R"
DTMF 5: Toggle relay, send its new state
DTMF *: Send current relay state
DTMF #: Send ID (N8LXS in this case)

The most important feature of the software aspect of this project is the ease with which it can be implemented. You simply connect a three-wire cable from an LPT: Port on your computer to a header on The Stamp, run The Stamp program, enter your program into a text editor (included), type ALT-R when you want to run, and your program will be off and running. Bugs? Well, there's a DEBUG instruction that enables you to print variable values back to the PC to see what might be wrong. With the possible exception of making a pin into an output when it is connected to an external output, it's just about impossible to break The Stamp with software. So go have some fun!

Using the Intelligent Relay

You'll find many ways to use this device in addition to those mentioned earlier. Radio Shack HTX-202s were used to develop the design in this article. A Yaesu FT-470 was used as the remote radio, and this worked fine. However, when the FT-470 was connected as the Intelligent Relay radio, some problems with the radio keying were encountered. You can fix problems like this by playing with the values of R1 and C2. Also noted, a Kenwood TM-732 has a feature that phone patch users enjoy (I presume), but causes a problem here. The TM-732 does not release the PTT for about two seconds after the last DTMF digit is entered. This allows phone patch users to release their PTT while entering strings of digits. Even though the Intelligent Relay carries out its commands, the TM-732 interferes with the acknowledgement transmissions. You could fix this by changing the key-up delay in the program (or by not using the TM-732).

What Else Could You Do With This Design?

There are lots of things that could be done with this application. For example, you could monitor inputs and report them on command. Or you could transmit a call-sign with the relay (could be handy for remote on-air testing). But it is possible to run out of program space. So, depending on your application, you'll either have to get real clever in optimizing your code or you'll have to limit your functionality. The former is my preferred method, even though that method can drive you nuts!

Conclusion

In addition to DTMF-controlled devices, The Stamp lends itself to many other applications. You could make a custom keyer, a rotor controller, an antenna switch, or many other simple controllers. And since the device is software-driven, you can now dazzle your friends with fancy features that couldn't be done before without a large bag of TTL or CMOS chips and a lot of time and aggravation. Have a good time working with this powerful, yet simple, device.

Parts List

R1	2.2k 1/4W	
R2,R3,R4	1.0k 1/4W	
R5	300k 1/4W	
R6,R7	100k 1/4W	
C1	0.01 25W VDC disc	
C2,C3,C4	0.1 25W VDC	
D1,D2	1N4001	
D3	Yellow LED	
D4	Green LED	
K1	Relay	Radio Shack #275-249-A
P1	Mike connector	Radio Shack #274-290
P2	1/8" connector	
Q1	2N2222 NPN transistor	
XTAL	3.579545 MHz crystal	
U1	The BASIC Stamp (Parallax)	
U2	Teltone M-8870 DTMF decoder	
Box	4" x 2-1/8" x 1-5/8"	Radio Shack #270-239
Hardware	4-40 screws/spacers/nuts for mounting The Stamp	
Grommet	3/8" grommet for cable entry	
Wire	Shielded cable/relay/power leads as required	

Parts Availability

A set of components, including The Stamp and a 3.5" disk with the source code on it, is available from RF Applications, Inc. for $95. We can program your Stamp for an additional $20 (please give us your call). Please order this kit by mail or fax, VISA/MC accepted (encouraged), at 9310 Little Mountain Road, Kirtland Hills OH 44060-7951; Fax (216) 974-9506.

Note: Unless you order The Stamp programmed from us, The Stamp RF Applications, Inc. supplies is unprogrammed. You'll need the Development Environment described in this article to load the software. The BASIC Stamp Development Environment is available from Parallax, Inc. 3805 Atherton Road, #102 Rocklin, CA 95765. Their telephone number is (916) 624-8333. The Stamp is a trademark of Parallax, Inc.

```
Rem BASIC STAMP Control Program
Rem Intelligent Relay
Rem By Bruce R. Knox
Rem RF Applications, Inc.
Rem August 28, 1994

Rem You may copy this code, but please acknowledge its origin.

Rem There is a DTMF decoder on 0-3, strobe (active high) is on 4.
Rem Pin 5 is used for transmit tone
Rem Pin 6 is the PTT for the radio
Rem Pin 7 is the relay output (active high)

Rem Commands:
Rem  1  = Pulse on for 1 second
Rem  2X = Toggle X times
Rem  3  = Turn on the relay
Rem  4  = Turn off the relay
Rem  5  = Toggle the relay then Function 11
Rem  11 = Report relay status (A=on, O=off)
Rem  12 = ID (N8LXS in this case)

Rem Set up Pin Direction Registers (5 ins and 3 outs)

Rem Pin 7 = Output: Relay,    1=ON
Rem Pin 6 = Output: Key Radio,     0=Keyed
Rem Pin 5 = Output: CW Tone Out
Rem Pin 4 = Input:  DTMF Tone Present, 1=Tone Present
Rem Pin 3 = Input:  DTMF Bit 3
Rem Pin 2 = Input:  DTMF Bit 2
Rem Pin 1 = Input:  DTMF Bit 1
Rem Pin 0 = Input:  DTMF Bit 0

dirs = %11100000

Rem relay off (there's an NPN transistor there, 1=on)

low 7

Rem A low on Pin 6 keys up the radio, so start unkeyed

high 6

Rem Wait for HIGH (1) on DTMF Data Present Line

loop0:
     if pin4 = 0 then loop0

Rem Mask off unwanted top four bits

     b2 = pins & 15

Rem Sort out command

     if b2 = 1 then func1

     if b2 = 2 then func2

     if b2 = 3 then func3

     if b2 = 4 then func4

     if b2 = 11 then func11

     if b2 = 5 then func5

     if b2 = 12 then func12

     loop1:
     if pin4 = 1 then loop1

     goto loop0

Rem Pulse relay on then off

func1:

     high 7
     pause 2000
     low 7
     goto wait_gone

Rem Pulse relay x times (x follows the 2)

func2:

     func2_loop:
          if pin4 = 1 then func2_loop

     func2_loop1:
          if pin4 = 0 then func2_loop1

     b3 = pins & 15
     b3 = b3 * 2

     for b4 = 1 to b3
     toggle 7
     pause 100
     next b4

     low 7

     goto wait_gone

Rem Turn on relay

func3:
     high 7
     goto wait_gone

Rem Turn off relay

func4:
     low 7
     goto wait_gone

Rem Toggle relay

func5:
     toggle 7

Rem Fall right into func11 to tell the state of the relay


Rem Use "O" for off, "A" for on (shortend Morse 1)

func11:
     if pin4 = 1 then func11

     if pin7 = 1 then func110
     pause 200
     low 6
     pause 400
     gosub dash
     gosub dash
     gosub dash
     pause 200
     high 6
     goto loop0

func110:
     gosub key_down
     gosub dit
     gosub dash
     pause 200
     high 6
     goto loop0

Rem Send call (N8LXS in this case)
Rem 1=dash, 2=dit, 3=intercharacter space

func12:
     gosub key_down
     for b2 = 0 to 21
     lookup b2,(1,2,3,1,1,1,2,2,3,2,1,2,2,3,1,2,2,1,3,2,2,2),b3
     gosub send
     next b2
     pause 200
     high 6
     goto loop0

Rem wait for DTF tone to go away before proceeding

wait_gone:

     if pin4 = 1 then wait_gone

     gosub key_down

     gosub dit
     gosub dash
     gosub dit
     pause 200

     high 6

     goto loop0

Rem Morse code/radio support routines

dit:
     sound 5,(100,5)
     pause 15
     return

dash:
     sound 5,(100,15)
     pause 15
     return

send:
     if b3 = 1 then dash

     if b3 = 2 then dit

     if b3 = 3 then send_delay

     return

send_delay:

     pause 80
     return

key_down:
     pause 200
     low 6
     pause 400
     return
```

Figure 4. The software. Nothing to it!