PIC Programming Project

REMOTE CONTROL IR DECODER

ROGER THOMAS

Allows PIC programming enthusiasts to remotely control their designs.

THIS design was created to enable PIC microcontroller circuits to be enhanced by the addition of a low cost infra-red sensor and suitable decoding software. The operation of the PIC software can then be selected via a remote control handset. This control option may be preferable to interfacing external switches to the PIC.

The circuit and program could also be used just as a simple tester to show that a remote control is working.

BASIC FUNCTIONS

Referring to Fig.1, the Remote Control Decoder uses an infra-red sensor (IC2) the demodulated output from which is connected to a PIC16x84 microcontroller (IC1) for decoding.

Remote control handsets can use a variety of different protocols. The PIC software decodes either the RC5 (Philips) or SIRC (Sony) transmission protocol as these are most likely to be used to control equipment in the home. These protocols are described later so that the decoding software can be understood and incorporated as part of another program for a more elaborate circuit.

To help demonstrate the decoding process, and provide programming examples, the PIC circuit incorporates two light emitting diodes (D1 and D2) connected to Port B. Certain remote control key codes are recognised by the PIC software and used to switch these l.e.d.s on or off.

Resistors R1 and R2 limit the l.e.d. current from the PIC. Additional l.e.d.s with suitable current limiting resistors can be added but note that the PIC can only source a maximum current of 20mA per port pin, with a maximum current total of 100mA for Port B.

The circuit can easily be built on stripboard and requires a regulated +5V power supply. No constructional details are offered. Software is available as stated later.

SERIAL INTERFACE

It can be difficult to predict what command code a particular remote control handset key will generate. Instead of switching on or off l.e.d.s, the value of the command code generated by the remote control handset can also be serially transmitted to a PC-compatible computer.

To achieve this, R3 is a series current-limiting resistor and connects Port B pin RB3 to pin 2 of a 9-pin D-type serial port socket (SK1 in Fig.2) so that the data from the PIC circuit can be sent direct to the PC's serial port. In serial mode, the PIC software needs to be amended with the l.e.d. output routine replaced by the serial port emulation software.

By running the PC serial link version of the PIC software the command values of different remote control handset keys are displayed. The lists which illustrate various command codes are given later, but can only be used as a general guide to what command code a given key on the handset may generate.

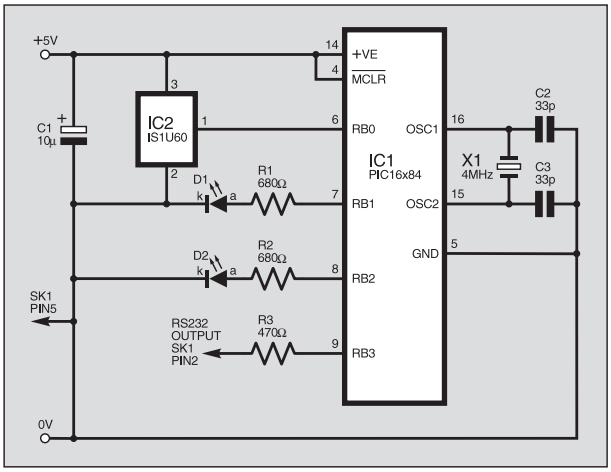

Fig.1. Circuit diagram for the Remote Control IR Decoder.

INFRA-RED SENSOR

The IS1U60 remote control infra-red sensor, IC2, is manufactured by Sharp. As can be seen from the block diagram in Fig.3, this device filters, amplifies and demodulates the infra-red signal. The final stage is a comparator circuit which gives a clean TTL output signal. Using this device is considerably easier (and cheaper) than building a circuit using a separate infra-red detector and amplifier. Pinouts are given in Fig.4.

Data output from the sensor is connected directly to the PIC at Port B pin RB0. (It could also be added to an existing PIC circuit with minimal additional wiring if a spare port pin is available.)

With no infra-red signal the output of the device is 5V (logic 1) and consumes a maximum current of 4·5mA (2·8mA typical). The recommended power supply range is 4·7V to 5·3V.

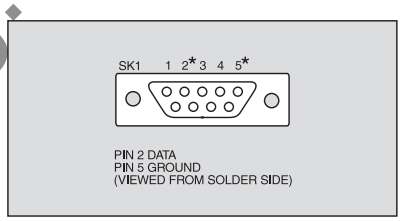

Fig.2. 9-pin D-type female serial connector.

COMPONENTS

Resistors

R1, R2	680Ω	(2 off)
R3	470Ω	

See SHOP TALK page

Capacitors

C1	10µF elect. 10V
C2, C3	33pF ceramic (2 off)

Semiconductors

D1, D2	red l.e.d. (2 off)
IC1	PIC16x84 microcontroller, preprogrammed (see text)
IC2	IS1U60 infra-red sensor

Miscellaneous

SK1	9-pin D-type serial connector, female
X1	4MHz crystal

Stripboard, size to suit; 5V power supply (see text)

Approx. Cost Guidance Only **£8** *excluding PSU*

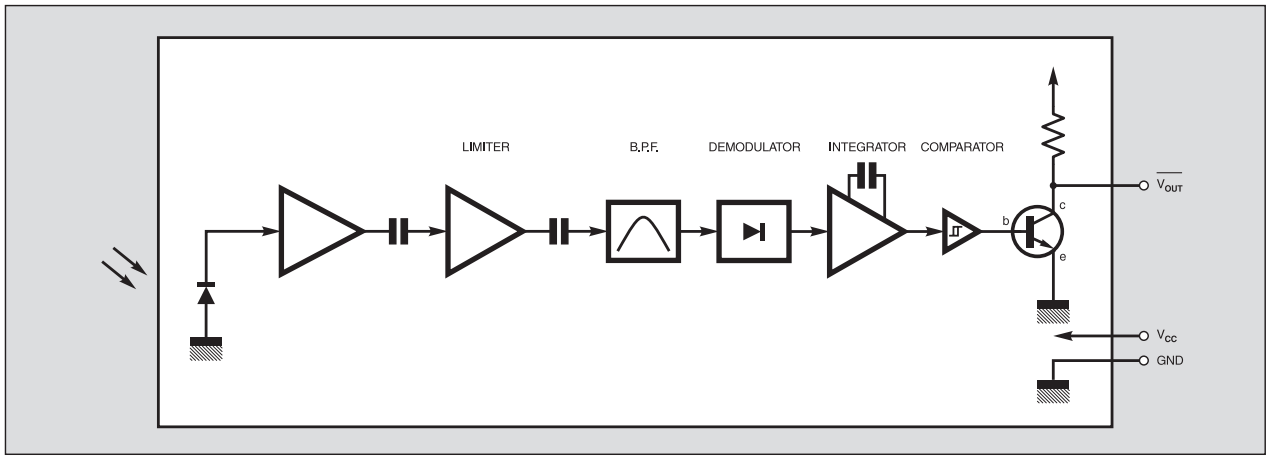

Fig.3. Block diagram for the IS1U60 remote control IR sensor.

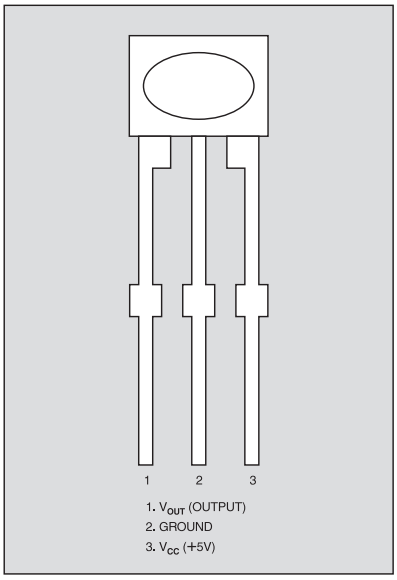

Fig.4. Pinouts for the IS1U60 sensor.

RC5 PROTOCOL

The RC5 remote control code protocol was developed by Philips and is used by several other manufacturers. However, it is worth noting that not all products manufactured by Philips use this protocol.

An RC5 transmission has a duration of approximately 25 milliseconds and contains 14 bits of data. A logic 0 is encoded by a high-to-low transition and a logic 1 by a low-to-high transition. This is called bi-phase coding, as illustrated in Fig.5.

The arrangement of the 14-bit code is given in Fig.6. The first two bits (S) of the transmission are Start bits and are always transmitted as logic 1. This allows the IR receiver to adjust its automatic gain control to suit the infra-red signal strength. The Control bit (C) toggles whenever a new key is pressed, or if a key is held down and a repeated transmission is made every 113 milliseconds.

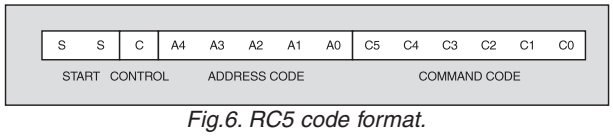

Fig.6. RC5 code format.

Next is the Address (A4 to A0) of the equipment that is to respond to the command transmitted. With five bits there are 32 different devices that can be addressed. Some of the more common addresses are given in Table 1. Note that the software of the decoder described here does not actually decode the device address but the program could be altered to do so.

After the address come the six Command code bits (C5 to C0), giving a total of 64 different commands that can be transmitted. Some of the more common commands are listed in Table 2. Commands 0 to 17 are used mostly to control a TV receiver, commands 41 to 46 are used for teletext, and 47 to 55 used to control a video tape recorder.

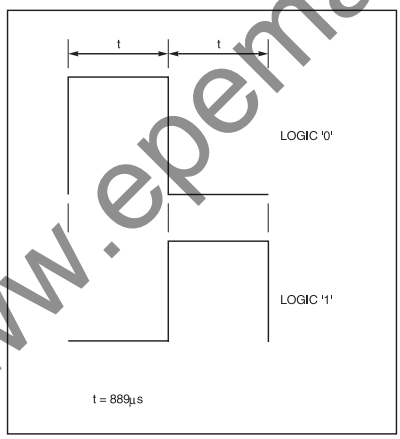

Fig.5. RC5 timing of logic 0 and logic 1 data.

Table 1. Example RC5 device addresses.

Address	*Device*
0	TV receiver 1
1	TV receiver 2
2	teletext
5	video recorder 1
6	video recorder 2
7	experimental
8	satellite
16	preamplifier 1
17	tuner
18	audio tape recorder 1
19	preamplifier 2
20	CD player
23	audio tape recorder 2

Table 2. RC5 command codes.

Command	*Function*
0 - 9	numerals 0 to 9
10	digits
11	select
12	stand-by
13	mute
14	presets
15	display
16	volume +
17	volume –
41	page
42	timer
43	large
44	reveal
45	cancel
46	subtitle
47	store
48	pause
49	erase
50	fast reverse
51	fast forward
52	rewind
53	play
54	stop
55	record

RC5 DECODING SOFTWARE

RC5 transmissions are relatively slow in comparison to the operation of the PIC microcontroller. However, due to the bi-phase encoding, a more complicated decoding algorithm is needed than might be expected.

The decoding software works by using the falling edge of the RC5 signal to generate an interrupt. The 8-bit internal RTCC (Real Time Clock Counter) timer value is read (**TIMERVAL**) after every interrupt and the RTCC timer is then set to zero and begins to count up again. PIC software times the IR sensor output from falling edge to falling edge. With a 4MHz crystal clock and prescaler set to 16, the timer is incremented every 16 microseconds.

As can be seen from the various logic combinations in Fig.7, despite the number of different waveform permutations, the edge-to-edge timing can be one of only three different values.

The output from the infra-red sensor is high and goes low when a signal is received, so on the first interrupt the timer value is not valid. Program variable **BITS**

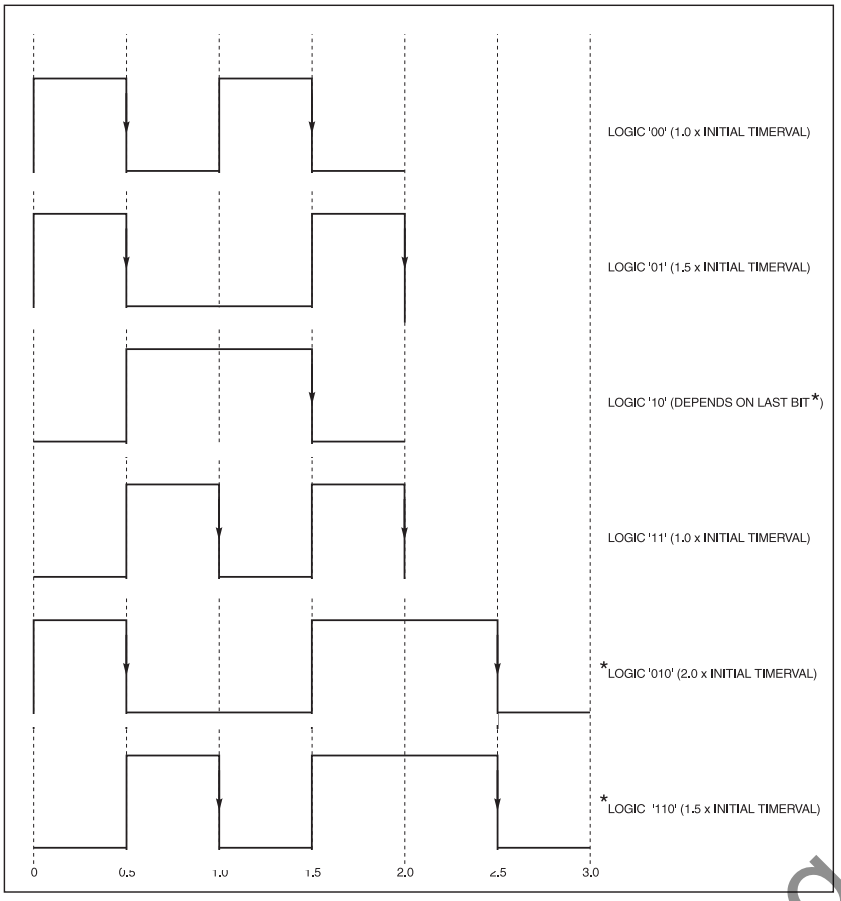

Fig.7. Example RC5 timing diagram.

is used to ensure that on the first interrupt (**BITS = 1**) the program variables are initialised but the RTCC timer value is not used.

On the second interrupt (**BITS = 2**) the RTCC timer value of the start bit is assigned to variable **TIMERVAL**. This value is used as a reference and all subsequent timer value calculations use it.

The **XVALUE** variables are used to set the three different **TIMERVAL** value ranges, this determines the waveform timing (see Listing 1). Using ranges of values rather than direct comparison to the first reading ensures that any timing discrepancy does not affect the operation of the program. Small variations in the RTCC value are inevitable due to PIC interrupt latency and tolerances between different remote controls.

Once the **TIMERVAL** comparison is made, the appropriate waveform time can be determined. If **TIMERVAL = 1** then the result will be the same as the last bit (value of variable **LASTBIT**). If **TIMERVAL = 1·5** then the result is to invert the last bit received. If **TIMERVAL = 2** and the previous bit was 0 then the result is binary 10.

LISTING 1. Setting TIMERVAL values.

```
XVALUE1 = 0.5 x initial TIMERVAL
XVALUE2 = 1.25 x initial TIMERVAL
XVALUE3 = 1.75 x initial TIMERVAL
if (current TIMERVAL > XVALUE1 and
   < XVALUE2) then TIMERVAL = 1
if (current TIMERVAL > XVALUE2 and
   < XVALUE3) then TIMERVAL = 1.5
if current TIMERVAL > XVALUE3 then
   TIMERVAL = 2
```

The **ADDBINARY** routine is then called, which updates the value of the **CBINARY** (command binary) variable using the **THISBIT** variable value. The **BITS** value must be greater than eight so that only the command part of the RC5 sequence is decoded.

If **THISBIT = 1** then the appropriate bit within the **CBINARY** variable byte is set to 1. This is done by logic ORing **BITVALUE** and **CBINARY**. Dividing **BITVALUE** by two sets the next bit within this variable to 1.

Initially the value of **BITVALUE** is 32 (binary 100000), so dividing **BITVALUE** by two gives 16 (binary 010000). Division by two is done by shifting the variable to the right by one place using the **RRF** instruction (Rotate Right File). If **THISBIT = 0**, only **BITVALUE** needs to be altered as the relevant bit within **CBINARY** is already zero.

An alternative decoding method considered was to use a timer-generated interrupt to sample the waveform every 889 microseconds, after detecting the initial waveform edge. However, if the RC5 transmission is faster or slower due to differences between remote handsets, then there is a possibility that accumulated timing error would cause either a pulse to be missed or the same pulse to be sampled twice.

Observation of the waveform will show that if the last pulse of an RC5 transmission is zero, then there is no final falling edge to enable an interrupt to read the timer. With no interrupts the RTCC timer will reach 255 (maximum byte value) and start counting from zero. This "roll-over" sets the timer overflow flag, which is used to indicate the end of transmission and the **LEDDISPLAY** output routine is called.

RC5 ASSEMBLER PROGRAM

Once the RC5 assembler listing is programmed into the PIC the decoding software can be tested. If the key marked "1" on the remote control is pressed one l.e.d. (D1) will come on, if the "2" key is pressed then the other l.e.d. (D2) will come on. If the "3" key is pressed then both l.e.d.s come on.

To change which key alters the l.e.d., change the **CBINARY** comparison value in the **LEDDISPLAY** routine. For example, using a VCR remote control, change the three comparisons to D'53', D'54', D'55' (to change from hexadecimal to decimal notation replace H'nn' with D'nnn' in the assembler program).

On the remote control handset pressing the VCR Play key should generate command code 53 and one of the l.e.d.s should light. Pressing the Stop key should generate code 54 and light the other l.e.d. Pressing the Record key should generate 55 and both l.e.d.s should be on.

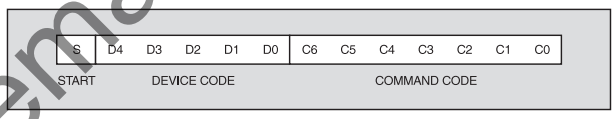

Fig.8. SIRC code format.

SIRC PROTOCOL

SIRC (Serial Infra-Red Control) protocol is the name given to Sony's IR remote control system. The 12-bit protocol is the most common format used with domestic products but there are others, including 15-bit and 20-bit versions. Control-S protocol is the hard-wired TTL version of the infra-red SIRC signal.

In most respects these transmissions are easier to decode than RC5. Several of the routines and variables used in the SIRC decoding program are similar to the ones used in the RC5 program. The command word is made up of 12 bits, and consists of a 5-bit device code followed by a 7-bit command code, see Fig.8. This SIRC format uses pulse width modulation of the infra-red signal to transmit the data.

The SIRC transmission is preceded by a single start bit, unlike the RC5 code. The SIRC decoding software waits for this start bit of 2·4 milliseconds. When it is correctly received the **START** variable is set to 1 to allow the rest of the transmission to be decoded.

Using a unique signal as a start bit helps prevent the software trying to decode an incomplete transmission. The infra-red sensor uses this start pulse to set its automatic gain control.

The SIRC command sequence is usually transmitted at least three times and, for some reason best known to Sony, the data is sent in reverse order. There is no equivalent to the control toggle bit as used in the RC5 protocol. Like the RC5 transmission there is no additional information transmitted to allow for error detection.

The SIRC data consists of either pulses of 0·6ms or 1·2ms duration, meaning logic 0 and logic 1 respectively. Each pulse is preceded by a 0·6ms pause. The pulse length is measured by the falling edge of the waveform generating an interrupt. The timer value is incremented every 16 microseconds and is read on every interrupt.

To work out the likely timer values, divide the expected pulse width by the timer "tick", illustrated in Fig.9 and Fig.11.

$$\frac{\text{pulse width}}{\text{timer}} = \frac{\text{start pulse}}{16\mu s} = \frac{2{\cdot}4ms}{16\mu s} = 150$$

Fig.9. SIRC timer formula.

2·4ms	= 150 (start)
2·4ms + 0·6ms	= 187 (start)
1·2ms + 0·6ms	= 112 (logic 1)
0·6ms + 0·6ms	= 75 (logic 0)

Fig.10. SIRC TIMERVAL for all pulse widths.

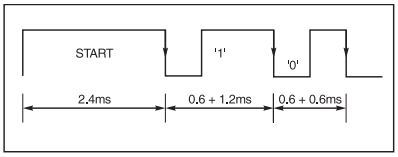

Fig.11. SIRC timing details.

The program uses the timer value to determine the waveform. For example, if the value is between 90 and 150 then a logic 1 is assumed and **THISBIT = 1**. If the value is between 50 and 90 then a logic 0 is assumed and **THISBIT = 0**. The **ADDBINARY** routine is called and the appropriate bit within **CBINARY** is set to the value of **THISBIT**.

SIRC ASSEMBLER PROGRAM

With the SIRC code running press the Increase Volume key on the Sony remote control and one l.e.d. (D1) will come on the other l.e.d. (D2) will be off. Press the Decrease Volume key and the l.e.d.s will invert.

To change which key controls the l.e.d., select the appropriate value for the key function and use that value in the **LEDDISPLAY** routine. Note that the remote control may generate different numbers for the same function so that the Sony equipment can distinguish between, for example, Play for the CD player and Play for the tape recorder. See Tables 3 to 5.

As there is no error detection or data verification with either of the IR protocols, errors can occur if the IR signal is not received correctly. Also, strong sunlight falling on the sensor can generate a signal.

Table 3. SIRC device code.

Command	*Device*
1	TV receiver
2	video tape recorder 1
4	video tape recorder 2
6	laser disk
12	surround sound unit
16	cassette deck/tuner
17	CD player
18	equaliser

Table 4. SIRC VCR FUNCTIONS

Command	*Function*
0 - 9	numerals 0 to 9
9	10/0
20	x2 play
21	power
22	eject
24	stop
25	pause
26	play
27	rewind
28	fast forward
29	record

Table 5. SIRC TV FUNCTIONS

Command	*Function*
0 - 9	numerals 0 to 9
9	10/0
16	channel +
17	channel –
18	volume +
19	volume –
20	mute
21	power
22	reset
23	audio mode
24	contrast +
25	contrast –
26	colour +
27	colour –
30	brightness +
31	brightness –
38	balance left
39	balance right
47	power off

SERIAL PORT

The PIC16x84 microcontroller does not have a built-in serial port but one can be implemented in software. Replace the entire routine **LEDDISPLAY** with the **TXDATA** code in the PIC assembler program. Add the two equates to the top of the assembler program and the **BCF PORTB,RS232** to the **MAIN** (SIRC) or **START** (RC5) procedure. This enables the RB3 port pin to be used as an output. In routine **LOOP** replace **CALL LEDDISPLAY** with **CALL TXDATA**.

The **TXDATA** routine works by ANDing each data bit with the relevant bit in **CBINARY**. This sets the output bit (called **RS232**), then the **OP** (output) routine is called and takes the pin RB3 high or low according to the value of bit **RS232**. Directly changing RB3 in the **TXDATA** routine would cause a timing error.

Once the RB3 output is set, this data output value has to be held, consequently several **NOP** commands are required to ensure correct timing. There is no handshaking or data transmission from the PC, therefore the connection from the PIC to the serial socket has only two wires.

PC SOFTWARE

The Windows 95/98 software does not decode the IR transmission but displays the value of the **CBINARY** variable sent from the PIC. The program also displays, if available, a text message describing the key pressed. This text is read from two text files, called RC5.TXT and SIRC.TXT, these files must be located in the same directory as the program.

The text can easily be altered using Notepad to coincide with the intended remote control handset. These files store the relevant text in ascending order. For example, the first line is text for **CBINARY = 0**, the second line is for **CBINARY = 1**, and so on.

Operation of the PC software is very simple, select the serial communications port that the PIC circuit is connected to and then select the required protocol. The Reload key reloads the text files if they have been changed while the program is active. The relevant protocol l.e.d. should flash when data is received from the PIC circuit.

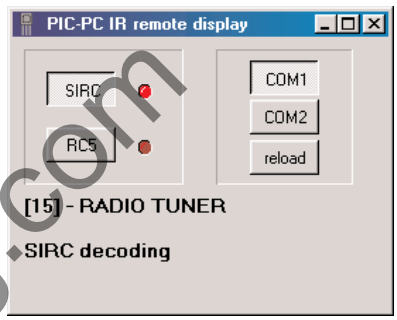

Fig.12. Example PC screen display.

REMOTE CONTROLS

A manufacturer using remote control of its equipment can allocate any command number to any key. Remote controls are not required to be compatible or exchangeable with equipment from another manufacturer, hence the plethora of remote controls and protocols found in most homes. The author has come across a remote control for a portable TV that uses RC5 coding for some of the keys and another protocol (not SIRC) for the remaining keys.

As neither PIC program decodes the device address then the result is a wider choice of remote controls being available. However, if a suitable remote control handset is not available then replacement remote controls are readily obtainable with a variety of functions and key layout. Most of these handsets are programmable and can replace many different models; therefore there is an implied choice of protocol.

Clearly the IR decoding program could alter a variable value or the status of a Boolean flag or control a device attached to one of the PIC ports. A number of different functions could be added as the assembler code associated with the **LEDDISPLAY** routine can be increased as necessary. Numeric input to a PIC program via a remote control handset can easily be achieved.

RESOURCES

The software discussed in this article is available as stated on this month's *Shoptalk* page. □