

FEATURE ARTICLE

Peter Gibbs

IR Remote-Controlled Video Multiplexer

It seems logical that there would be a push for smaller, lightweight, high-capacity batteries with today's outcropping of all kinds of portable equipment. Battery technology is making strides towards enhanced algorithms for faster charging and minimal battery damage. In this article, Thomas looks at the next generation of microcontrollers leading the way past the competition.



Here I work, the Physics Department needed to visually monitor a number of unattended experiments in various rooms. Multi-channel video monitoring and security systems, however, are usually high-cost, elaborate pieces of equipment requiring expert installation, which results in an isolated user who must call a technician for even minor modifications.

Therefore, I decided to construct a system that would use off-the-shelf components, allow user modification without denting the department's budget, and still allow you to monitor the experiments at will or under an automated scanning sequence from a remote location (see Photo 1).

THE OBJECTIVES

The main design objectives were to provide remote control of a multi-channel video system that was low-cost, had minimum components, and was expandable.

To achieve these goals, I used a Radio Shack 3-in-1 universal infrared remote controller (UIRC), normally used to control a TV, VCR, or standard household cable receiver, eliminating the need to build a hand-held control unit.

To decode the infrared signals from the UIRC, I used a low-cost Basic Stamp I microcontroller from Parallax Inc. The Stamp would also be used to control the video multiplexer and provide minimum visual and audible indication of the system's status.

The Basic interpreter language and the small RAM (256 bytes) on the Stamp I provided a challenge to fit the application software into the available space, but unique statements in the Stamp's Basic language came to the rescue.

CIRCUIT OPERATION

A Sharp GP1U52X infrared detector module was used to detect the signals from the hand-held control unit. It operates from a 5-VDC power supply, and its serial digital data output is fed directly to input bit 7 of the Stamp. It's important to remember that the metal case of the GP1U52X module must be grounded to minimize interference from stray radiation.

After software decoding the trans-

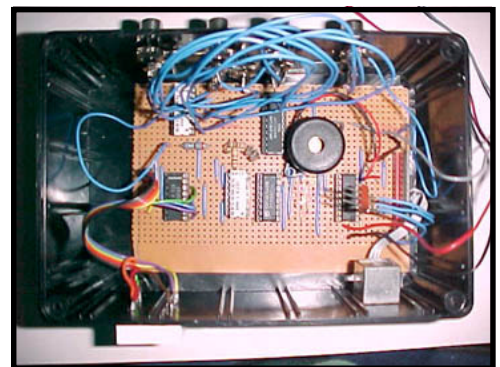


Photo 1—Here you can see an internal view of the video decoder with an IR sensor on the lower right, a seven-segment digit on the lower left, and a buzzer near the center of the perfboard.

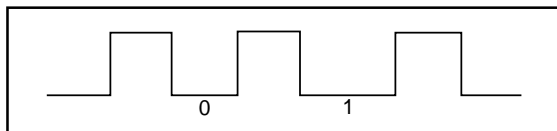


Figure 1—Notice how the low-pulse width of a logic 0 is distinctly shorter than that for a logic 1.

mission, the channel number selected is placed on output bits 0, 1, and 2 of the Stamp. This data is latched by a 74LS175 Quad D flip-flop using a clock pulse generated on output bit 3 of the Stamp.

The latch was included for the following reason. When the system is in scan mode, a delay of 3 s between channel changing is implemented. This may be changed in the software. The Sleep command is used to achieve this delay. However, when the Stamp executes the Sleep command, it momentarily places its I/O bits in tristate mode (approximately every 2.3 s during this delay). This produces a noticeable transient flicker on the display. To eliminate this, I had to isolate the Stamp I/O bits using the latch. If you can live with this flicker, the latch may be eliminated with a small reduction in software size and component count.

The outputs from the latch are used to address the video multiplexer (MAX455) and to drive a seven-segment decoder (74LS48) whose output is displayed on a large (2²) seven-segment LED digit. This digit, which shows the active channel in use, requires a transistor to drive each segment. I used a hex NPN transistor driver (CA3082) simply because it was available in-house. You may wish to use a display device that doesn't require external transistors and further reduce component count.

DECODING THE UIRC

Most infrared remotes use a 40-kHz modulated signal with pulse length coded data, encoded differently by each manufacturer. Researching the Internet and *Circuit Cellar's* archives revealed that there is no single standard, although some methods are widely used. As I mentioned earlier, I used a 3-in-1 universal remote control. It was used in the TV mode, and after experimentation, I determined that the remote uses a 12-bit code known as the

RECS-80 code.

To decode the transmissions, the Sharp GPIU52X infrared receiver module first removed the 40-kHz carrier, and then filtered and amplified the digital signal.

The TTL-compatible output was fed directly into an input bit of the Stamp (bit 7) for software analysis.

The Sharp module takes a short time to lock onto the signal, so a long leader pulse that satisfies this lock-on delay precedes transmissions. This is common on most remote controls.

Data is encoded using a method that distinguishes logic 1 from logic 0 by the length of a low pulse that follows each fixed duration high pulse (see Figure 1).

I reasoned that measuring pulse widths from logic 0 pulses only yields a binary string pattern unique for each key on the UIRC. The Pulsin command of the Stamp measures the duration of a pulse and returns an integer value

between 0 and 255 to an 8-bit variable and, hence, could measure the pulse widths.

SHORT PROGRAM

Listing 1 (*IRTEST.BAS*) shows a short Stamp program that allows you to visually examine the digital code produced by the UIRC buttons. A storage scope may also be used. Hook up the Sharp module to the Stamp (bit 7) and download Listing 1 from the PC to the Stamp.

Note that although the high pulse preceding each low pulse is read, the value and the leader pulse width are not stored. For the UIRC used, logic 1 returned a count between 135 and 145, and logic 0 fell between 70 and 80. You can try this method with other remotes and look for unique patterns.

Table 1 shows the binary data patterns (first 12 bits only), obtained for the buttons on my UIRC, in TV mode. I converted the count values to binary using integer division by 110, a num-

Listing 1—*IRTEST.BAS* reads UIRC and sends pulse width values to the PC.

```

Here:      pulsln 7,0,b0      ' checks for first low
pulse width on bit 7, part of header
      if b0=0 then Here      ' otherwise repeat check
      pulsln 7,1,b1          ' read but discard high pulse
width
      pulsln 7,0,b1          ' read & store first low data
pulse width
      pulsln 7,1,b2          ' remaining statements simply re-
repeat for data pulse widths
      pulsln 7,0,b2
      pulsln 7,1,b3
      pulsln 7,0,b3
      pulsln 7,1,b4
      pulsln 7,0,b4
      pulsln 7,1,b5
      pulsln 7,0,b5
      pulsln 7,1,b6
      pulsln 7,0,b6
      pulsln 7,1,b7
      pulsln 7,0,b7
      pulsln 7,1,b8
      pulsln 7,0,b8
      pulsln 7,1,b9
      pulsln 7,0,b9
      pulsln 7,1,b10
      pulsln 7,0,b10
      pulsln 7,1,b11
      pulsln 7,0,b11
      pulsln 7,1,b12
      pulsln 7,0,b12
      debug c!s,b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,cr,cr
      ' send values to PC
      goto Here

```

Button	Code
0	000000101100
1	000000100000
2	000000101000
3	000010100000
4	000010101000
5	100000100000
6	100000101000
7	100010100000
8	100010101000
9	000000100100
Mem	100001100000
PLAY	110001100010
STOP	110001101010
REW	110010100000
FF	110010101000
PAUSE	110000101010
ENT	000010101100
Ch+	000001100000
Ch-	000001101000
Vol+	000011100000
Vol-	000011101000
Prev Ch	010011101100

Table 1—Here you can see the binary bit-string patterns for the 3-in-1 Radio Shack UIRC.

ber midway between the two ranges.

THE MULTIPLEXER DESIGN

For the video multiplexer design, I only required buttons 0 through 7 for 8-channel selection, and I used the memory (Mem) button to start the scanning mode. Close examination of Table 1 shows that, to decode a specific button from among the nine buttons, only bits 1, 5, 6, 7, 9, and 10 are required (bit 7 is retained for decoding button 1, see Table 2 and Figure 2). All

other bits were identical for this button group.

Because of the simple software decoding scheme I used (although the memory button is used for selecting the scan mode), any button not decoded will initiate this mode. Note that, depending on the UIRC used and the protocol of the manufacturer, the count values mentioned above will vary. Also, using a faster microcontroller would require a 16-bit variable because the count could exceed 255.

SOFTWARE

Listing 2 (IR.BAS) is the final program downloaded to the Stamp. It polls bit 7 for a low pulse to determine if a button is pressed. If no button is pressed, it checks to determine if the system is in scan mode or locked onto a specific channel and then takes appropriate action.

The main limitation of the design was the lack of an interrupt, which would allow instant recognition of button activity. Unfortunately, a button pressed during the 3-s sleep period is ignored. As a result, during the polling stage, you must hold down a button until it is recognized. This is only necessary during scan mode.

To save precious scan memory, variable b0 is reused as a garbage variable when pulses are read in. Bit values not necessary for decoding are discarded. Finally, it is used to calculate the button value. This value is then stored in variable b3, and the Lookdown command is used to load pins. The state-

$$\begin{aligned}
 0 &= *b_1 \cdot *b_5 \cdot *b_6 \cdot *b_7 \cdot *b_9 \cdot *b_{10} = 000111 \\
 0 &= *b_1 \cdot *b_5 \cdot *b_6 \cdot *b_7 \cdot *b_9 \cdot *b_{10} = 000100 \\
 0 &= *b_1 \cdot *b_5 \cdot *b_6 \cdot *b_7 \cdot *b_9 \cdot *b_{10} = 000110 \\
 0 &= *b_1 \cdot *b_5 \cdot *b_6 \cdot *b_7 \cdot *b_9 \cdot *b_{10} = 010100 \\
 0 &= *b_1 \cdot *b_5 \cdot *b_6 \cdot *b_7 \cdot *b_9 \cdot *b_{10} = 010110 \\
 0 &= *b_1 \cdot *b_5 \cdot *b_6 \cdot *b_7 \cdot *b_9 \cdot *b_{10} = 100100 \\
 0 &= *b_1 \cdot *b_5 \cdot *b_6 \cdot *b_7 \cdot *b_9 \cdot *b_{10} = 100110 \\
 0 &= *b_1 \cdot *b_5 \cdot *b_6 \cdot *b_7 \cdot *b_9 \cdot *b_{10} = 110100 \\
 \text{Mem} &= *b_1 \cdot *b_5 \cdot *b_6 \cdot *b_7 \cdot *b_9 \cdot *b_{10} = 101100
 \end{aligned}$$

Figure 2—I've provided the logic equations used in Listing 2 to decode the nine keys from Table 2.

ment PULSOUT 5, 5000 beeps a buzzer for audible indication of channel change, and PULSOUT 4,5 clocks the latch.

The complete schematic of the system is shown in Figure 3. I take the composite video output to a small video transmitter, set it to Channel 13, and view the display on a small TV in my office.

For power, I use a standard ± 5 -VDC power supply from our digital electronics labs; demand is small and any supply capable of 500 mA will suffice.

I am sure you can improve and simplify my program, which was written to merely get the system operational. I've provided a parts list of the items I used in the system. So far, the system has worked, and it's been in use since August of 1999.

Peter Gibbs has received a BS in Physics and Math, an Education certification, and an MS in Physics. He enjoys doing research in condensed matter physics, computational physics, and microprocessor application, instru-

mentation, and control. Peter has lectured about physics and electronics at the University of the West Indies (Barbados) since the 1980s. And when not working, he enjoys triathlons, water polo, and deep-sea fishing.

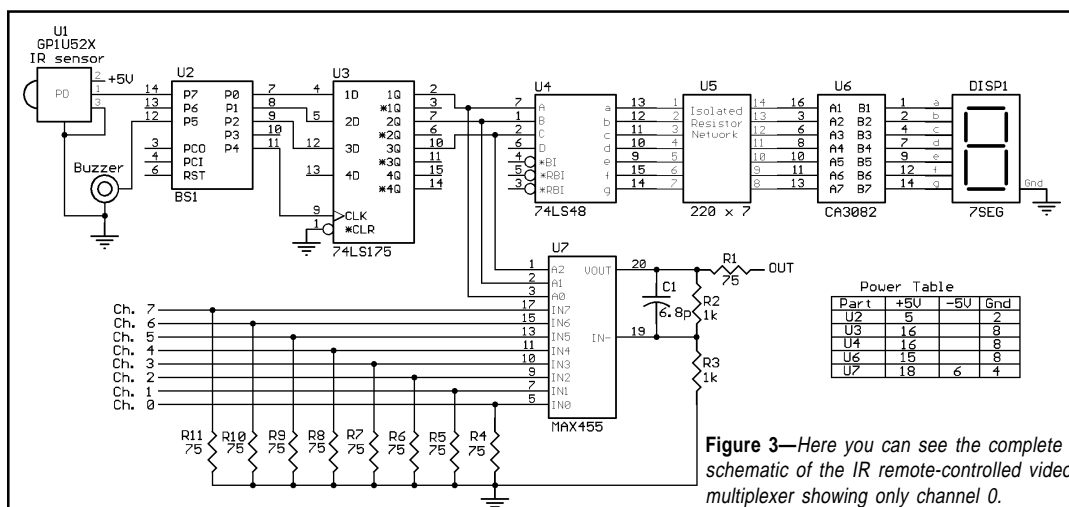


Figure 3—Here you can see the complete schematic of the IR remote-controlled video multiplexer showing only channel 0.

SOURCE

IR Module
Sharp Electronics Corp.
(360) 834-8611
Fax: (201) 529-8425
www.sharp-usa.com

CA 3082, MAX 455, buzzer, seven-
segment digit
JDR Microdevices
(800) 538-5000
(408) 494-1400
Fax: (408) 494-1420
www.jdr.com

	Bitstream											
Key	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	1	0	1	1	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0
2	0	0	0	0	0	0	1	0	1	0	0	0
3	0	0	0	0	1	0	1	0	0	0	0	0
4	0	0	0	0	1	0	1	0	1	0	0	0
5	1	0	0	0	1	0	1	0	0	0	0	0
6	1	0	0	0	0	0	1	0	1	0	0	0
7	1	0	0	0	1	0	1	0	0	0	0	0
Mem 1	0	0	0	0	0	1	1	0	0	0	0	0

Table 2—The bitstream pattern of the nine keys used in the final design can be seen here.

Circuit Cellar, the Magazine for Computer Applications. Reprinted by permission. For subscription information, call (860) 875-2199, subscribe@circuitcellar.com or www.circuitcellar.com/subscribe.htm.