

# Optical encoder controls range switch

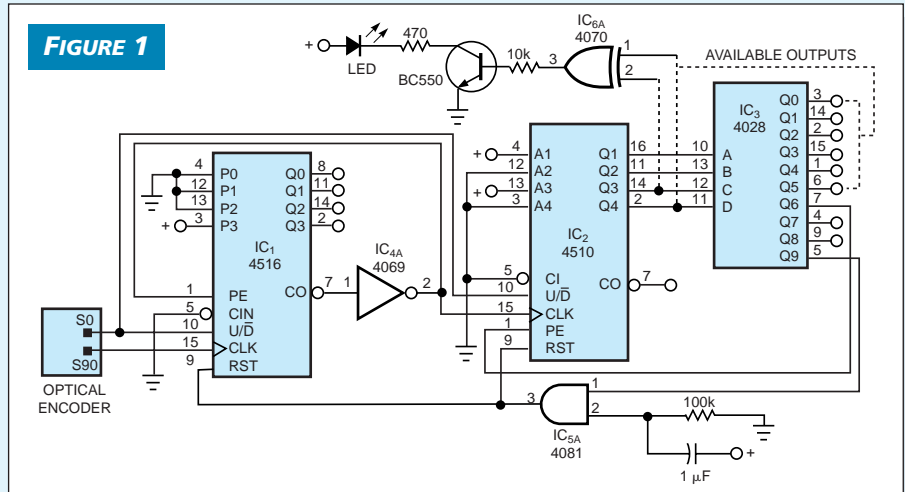
W DIJKSTRA, WAALRE, THE NETHERLANDS

Instead of using a counter-controlled, pushbutton-activated range switch, you can use an optical encoder. Inexpensive encoders are available, and they occupy minimal space on the front panel of an instrument. Moreover, an encoder gives you the opportunity to select the optimum operating speed. However, at positions near the transition points in counter position, mechanical shocks can provoke false switching. The circuit in **Figure 1** overcomes the false-switching problem.

Output S0 of the optical encoder controls the up/down inputs of counters IC<sub>1</sub> and IC<sub>2</sub>. Output S90 connects to the clock input of the HEF4516 binary counter (IC<sub>1</sub>). When this counter reaches 0 or 15, the output CO goes low and clocks (via an inverter) the HEF4510 decimal counter (IC<sub>2</sub>).

Simultaneously, the binary counter assumes a value of eight. Thus, it requires eight pulses of the optical encoder to alter the position of the decimal counter. You must stop turning the optical encoder within eight pulses, which in practice is eminently possible. When you want more security, you can feed the outputs Q3 and Q4 of the decimal counter to an exclusive OR gate. When the output of the XOR gate is high, changing the state of the decimal counter requires a minimum of four pulses from the optical encoder.

The circuit provides control with bidirectional hysteresis.



**FIGURE 1** An optical encoder, immune to false switching, takes the place of a counter-controlled range switch.

To stop the decimal counter at zero when counting down, the counter resets when output Q9 of the 1-of-10 decoder HEF4028 (IC<sub>3</sub>) goes high. To limit the number of decoder positions, you can load the decimal counter one position lower than the maximum output position you want to reach with the output decoder. The configuration in **Figure 1** loads the decimal counter with the value five when Q6 (the seventh position) of IC<sub>3</sub> goes high. (DI #2255). **EDN**

To Vote For This Design, Circle No. 399

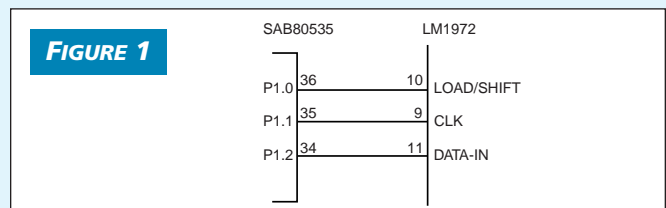
## μC controls digital potentiometer

LUKASZ SLIWZYNSKI, UNIVERSITY OF MINING AND METALLURGY, KRAKOW, POLAND

Many digitally controlled potentiometers (for example, the LM1971/2/3 from National Semiconductor, www.national.com) incorporate a three-wire serial digital interface, using data, clock, and enable lines. In **Figure 1**, the potentiometer's nomenclature for these lines is Data-In, Clk, and Load/Shift, respectively. The assembler program in **Listing 1** provides an interface to an SAB80535 μC. The main idea of the method is to use the capture/compare capability of Timer 2 in the μC to provide the timing relationship between the Data-In and Clk signals. The principal control of the interface comes from subroutine S16BIT in **Listing 1**.

To program the potentiometer, the μC must send 2 bytes to it—the "channel address," followed by the attenuation value with its most significant bit first. Sending a byte starts

by loading the number of bits to send into the μC's BCOUNT register and initiating the P1 lines (setting P1.0 through P1.3 to a low state). Next, Timer 2 starts with overload enabled. The routine sets two digital compare/capture units (CC1 and



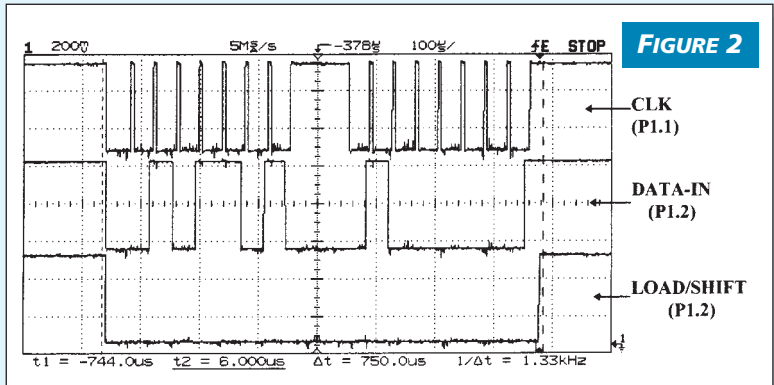
The rising edge of the clock signal validates data loading into the Data-In and Load/Shift pins of the potentiometer.

CC3 in the  $\mu\text{C}$ ) to the "compare" mode by writing 88H into the CCEN register. The contents of the register decrease after each interrupt. In this way, eight consecutive interrupts occur, each to send 1 bit of data. The interrupt subroutine at address 006BH manages the transmission of the data bits via P1.2.

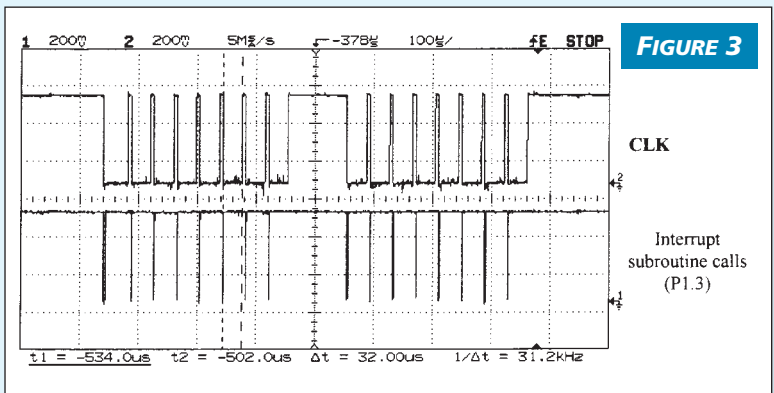
The CC1 unit generates the Clk signal on pin P1.1 when the contents of the Timer 2 count register (composed of TH2 and TL2 8-bit registers) equal the values set in registers CCL1 and CCH1. This value depends on the length of the interrupt subroutine. After transmission of the last bit, the routine stops Timer 2 by setting the value 0ECH in register T2CON. During data transmission, the main program spends its time waiting in the loop, as long as the bit FLAGS.0 is at logic 1. This bit clears in the last pass of the subroutine and sets just before Timer 2 starts. Transmission of the second byte of data occurs in exactly the same way after the routine reprograms the related registers. The Channel and Volume registers hold the 2 bytes to send.

Figures 2 and 3 show the timing relationships of the interface. In Figure 2, the 2 bytes Channel and Volume are 05AH and 081H, respectively. Data is valid on the rising edge of the Clk signal. Figure 3 shows the time dependence in the interrupt-routine calls and the Clk rising edges. In this design, it takes approximately 740  $\mu\text{sec}$  to program 2 bytes into the potentiometer with an 8-MHz clock frequency (a 1.5- $\mu\text{sec}$  machine-cycle time). (DI #2256). EDN

To Vote For This Design, Circle No. 400



The rising edge of the clock signal appears approximately 32  $\mu\text{sec}$  after the CC3 compare/capture unit in the  $\mu\text{C}$  generates an interrupt.



The subroutine in Listing 1 provides a simple serial digital interface to National's (and others') digital potentiometers.

## LISTING 1—S16BIT SUBROUTINE

```

*****REGISTERS ADDRESSES
DEFINITIONS*****
.EQU FLAGS,002FH
.EQU BCOUNT,004FH
.EQU CHANNEL,004DH
.EQU VOLUME,004EH

```

```

*****INTERRUPT SUBROUTINE
HANDLER*****
;THE NUMBER OF MACHINE CYCLES TO EXECUTE THE COMMAND ARE
GIVEN AT THE ;RIGHT-HAND SIDE. THREE ADDITIONAL CYCLES ARE ADDED
FOR THE RESPONSE TO ;THE INTERRUPT REQUEST

.ORG 006BH
CLR IEN0.7, DISABLE ALL INTERRUPTS      1
PUSH ACC;                               2
PUSH PSW;                               2
MOV A,BCOUNT;                             1
JZ ALL_BY; TEST IF THE LAST BIT TO SEND 2-> 8 + 3 = 11
DEC A;                                     1
MOV BCOUNT,A;                             1
MOV A,CHANNEL;                             1
RLC A;                                     1
MOV P1.2,C; SEND THE BIT                  2-> 14 + 3 = 17
MOV CHANNEL,A;                             1
SETB IEN0.7;                               1
POP PSW;                                   2
POP ACC;                                   2-> 22 + 3 = 25
RETI;                                       2
ALL_BY: MOV A,CHANNEL; THE LAST BIT TO SEND 1
        RLC A;                               1
        MOV P1.2,C; SEND THE BIT              2-> 12 + 3 = 15
        CLR FLAGS.0, INFORM S16BIT FUNCTION  1
        POP PSW;                             2
        POP ACC;                             2
        ANL T2CON,#0ECH; STOP TIMER 2        2
        SETB IEN0.7; ENABLE INTERRUPTS      1
        RETI;                                2-> 22 + 3 = 25

```

```

*****LM1972 INTERFACE CONTROLLING
FUNCTION*****
;THIS COMMANDS SHOULD BE PLACED SOMEWHERE IN THE BEGINING OF
THE MAIN PROGRAMM
MOV SP,#06H; SET THE STACK POINTER
MOV IP,#020H; SET THE INTERRUPT PRIORITY LEVEL OF CC3
MOV FLAGS,#00H; CLEAR ALL BITS

```

```

S16BIT: MOV TL2,(255 - 25);SET TIMER 2 COUNT REGISTER
        MOV TH2,#0FFH
        MOV CRCL,(255 - 25);SET TIMER 2 RELOAD REGISTER
        MOV CRCH,#0FFH
        MOV CCL1,#0FCH; SET VALUE FOR CC1 - CLK SIGNAL
        MOV CCH1,#0FFH;
        MOV CCL3,(255 - 25); SET THE BEGINING OF INTERRUPT
        MOV CCH3,#0FFH
        MOV BCOUNT,#07H; NUMBER OF BITS TO SEND - 1
        SETB FLAGS.0
        ANL P1.#070H; INITIATE PORT 1
        MOV CCEN,#088H; ENABLE COMPARE MODE FOR CC1 AND CC
        SETB IEN1.5; ENABLE INTERRUPT FROM CC3
        ORL T2CON,#011H; START TIMER 2 WITH OVERLOAD ENABLED
S16_1:  JB FLAGS.0,S16_1; WAIT UNTIL THE FIRST BYTE IS SEND
        MOV TL2,(255 - 25); REINITIATE TIMER 2 COUNT REGISTER
        MOV TH2,#0FFH
        MOV BCOUNT,#07H; REINITIATE NUMBER OF BITS TO SEND
        MOV A,VOLUME; SWAP VOLUME AND CHANNEL CONTENTS
        MOV CHANNEL,A
        SETB FLAGS.0
        ORL T2CON,#011H; START TIMER 2 WITH OVERLOAD ENABLED
S16_2:  JB FLAGS.0,S16_2; WAIT UNTIL THE SECOND BYTE IS SEND
        MOV CCEN,#00H; DISABLE CC1 AND CC3 UNITS
        CLR IEN1.5; DISABLE CC3 INTERRUPT
        ORL P1.#07H; SET OUTPUT PINS TO THE HIGH STATE
S_END:  RET

```

# Timer inputs double as interrupt-request lines

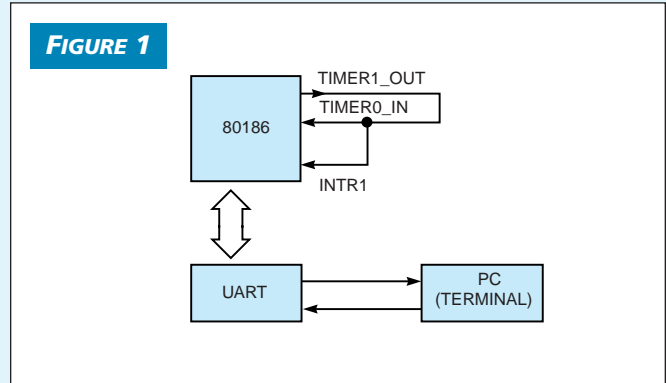
SK SHENOY, NAVAL PHYSICAL AND OCEANOGRAPHIC LABORATORY, KOCHI, INDIA

The Intel 80186 is a highly integrated 16-bit  $\mu$ P that is common to embedded applications. This  $\mu$ P's built-in interrupt controller has four interrupt-related pins that you can configure in various ways to achieve the maximum of four maskable interrupt-request lines.

In applications that require more than four interrupt-request lines, the only way out is to add an external interrupt controller, such as the 8259A. However, an alternative approach exists: You can through proper programming make the 80186's two timer-input pins function like normal edge-triggered interrupt request lines. The only penalty is an additional latency of about 1  $\mu$ sec for an 8-MHz CPU, which is insignificant for many applications. This pseudo interrupt-request line is useful for interfacing any device (UART, DMA controller, coprocessor, or DSP processor) that works with internally vectored, edge-triggered interrupts.

The idea is based on the fact that you can program the timer in a mode during which the timer count resets and then the timer starts counting on a 0-to-1 transition on its input pin. Further, you can set the timer to interrupt when it reaches the value set in its Max Count register. Thus, if the Max Count setting equals 1, the timer generates an interrupt immediately after one timer-clock period from the 0-to-1 transition on the input pin. For an 8-MHz system, with the timer using the internal clock (which is one-fourth the CPU clock), the interrupt time is 500 nsec. The  $\mu$ P can serve the timer interrupt as any other normal interrupt would. The  $\mu$ P can also selectively mask and unmask the interrupt using the timer/interrupt-control registers.

You can download a demo program from EDN's Web site,



With the proper programming, the Timer0\_In input serves as a pseudo interrupt-request line.

[www.ednmag.com](http://www.ednmag.com). (At the registered-user area, go into the Software Center to download the file from DI-SIG, #2277.) The program demonstrates the use of the Timer0\_In line as a pseudo interrupt-request line. The program, along with the setup in **Figure 1** (which uses the Timer0\_out line to generate the interrupt), also compares the latency of this pseudo interrupt with the latency of the normal interrupt line, Intr1. The program was written and compiled using Intel's IC86 compiler and was tested on an 8-MHz 80186 system. The same technique should work for  $\mu$ Ps and  $\mu$ Cs that have similar timer capabilities. (DI #2277) **EDN**

To Vote For This Design, Circle No. 401

# DSP algorithm measures frequency and damping

OLGA BELOUSOVA, LOS ALAMOS, NM, AND ALEXANDER BELOUSOV, NEW YORK, NY

The time-domain DSP algorithm described here allows you to measure the key parameters—natural frequency and damping—in linear, second-order electromechanical systems. The method applies to a range of electromechanical transducers (electromagnetic or electrostatic), including dynamic speakers, seismic geophones, micromachined sensors, and other systems. The algorithm is based on the integral transforms of the terminal voltage of the transducers in a free transient mode (after application and removal of the step-function stimulus). It provides high immunity to both electrical noise and mechanical vibration.

Compared with traditional FFT methods, the algorithm significantly simplifies the computational task, provides better resolution in locating the spectral peak (which corre-

sponds with the natural resonant frequency of the transducer), and allows you to calculate the damping coefficient (which you can not directly extract from an FFT). **Figure 1** uses an electromagnetic transducer, stimulated by a step-current function, with sequential integration of its terminal voltage in a free transient mode. The general equation (assuming zero initial phase) is:

$$v(t) = V_0 \cdot \exp(-w_0bt) \cdot \text{SIN}(w_0(1 - b^2)^{0.5}t),$$

where  $V_0$  is the final amplitude of the terminal voltage,  $\beta$  is the damping coefficient (an unknown value), and  $\omega_0$  is the natural angular resonant frequency of the transducer system. The algorithm is based on the following integral transforms of the terminal voltage,  $v(t)$ :

$$J_1 = \int_0^{\infty} v(t) dt,$$

$$J_2 = \int_0^{\infty} |v(t)| dt,$$

$$J_3 = \int_0^{\infty} (v(t))^2 dt.$$

Omitting the intermediate math, you can write the computational formulas for  $\beta$  and  $\omega_0$  as

$$b = (1 + p^2 (\ln((J_2 + J_1)/(J_2 - J_1)))^{-2})^{-0.5},$$

$$\omega_0 = 4b \cdot J_3 / J_1^2.$$

For a practical implementation in DSP form, you must substitute the indefinite integrals  $J_1$  through  $J_3$  with corresponding finite sums  $S_1$  through  $S_3$ . The ADC must digitize the terminal voltage during a period of time long enough to allow the free transient to settle. Then, the DSP must calculate the three finite sums using the following equations:

$$S_1 = (v_i), S_2 = |v_i|, \text{ and } S_3 = (v_i)^2.$$

The final computational formulas are as follows:

$$b = (1 + p^2 (\ln((S_2 + S_1)/(S_2 - S_1)))^{-2})^{-0.5},$$

$$\omega_0 = (4b \cdot S_3 / S_1^2) / \Delta t,$$

where  $\Delta t$  is the sampling period.

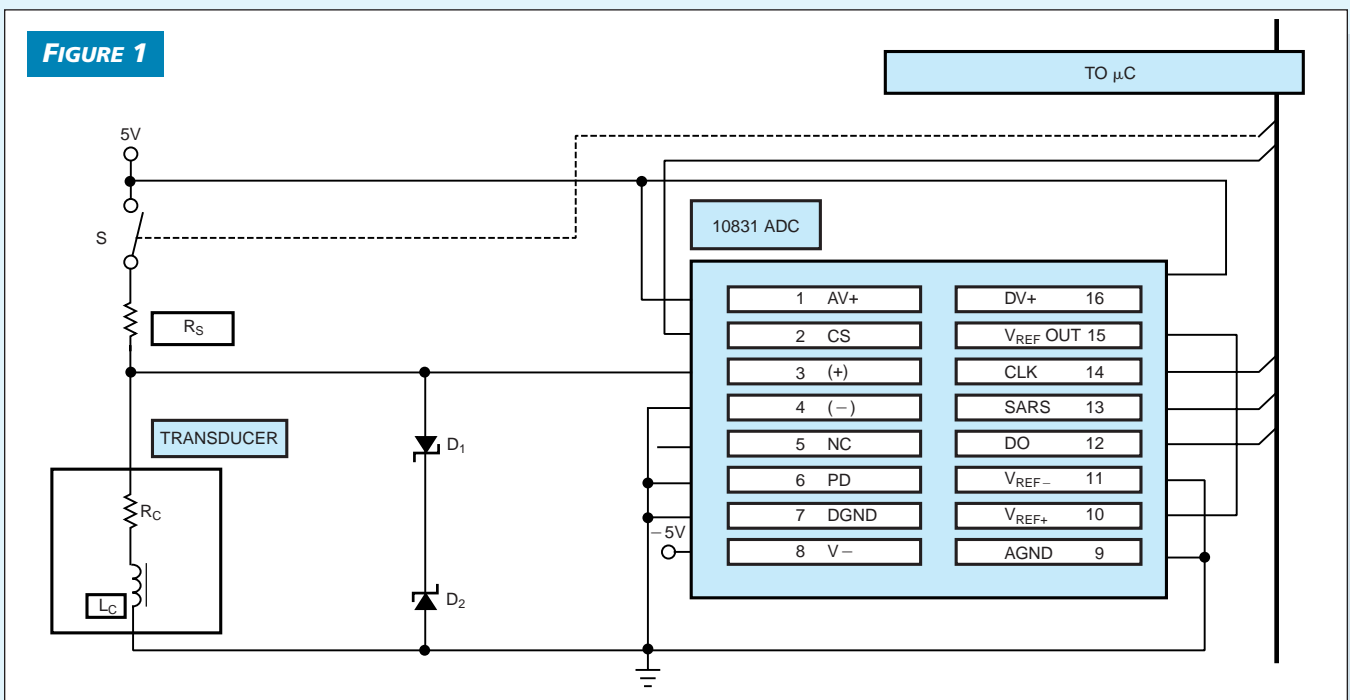
The circuit in **Figure 1** consists of the transducer under test, shown as inductor  $L_C$  in series with the coil resistance,  $R_C$ ; a current-stimulus circuit (analog switch  $S$  with a current-

limiting resistor,  $R_S$ , and the integrated ADC. We chose the low-power 10831 ADC from National Semiconductor ([www.national.com](http://www.national.com)) because of its convenient serial interface to the  $\mu C$ . Zener diodes  $D_1$  and  $D_2$  (approximately 5V breakdown) protect the ADC's input against overvoltage from inductive spikes).

The principal of operation is simple. First, you apply the current step function to the transducer. The duration of the current stimulus should be long enough to allow the transient to settle; this value depends on the estimated natural frequency and damping in the system. For example, for a typical seismic geophone with  $f_0=10$  Hz and  $\beta=0.6$ , the step function should last 0.5 to 1 sec. The same rule applies to the measuring cycle in a free transient. In general, you can stop the measurement when the terminal voltage drops to less than 1 LSB in the chosen ADC. The sampling period,  $\Delta t$ , should be small enough to avoid methodical errors that accrue from substituting the analog integral transforms with discrete sums.

Because the resonant frequencies of mechanical systems are typically low, it is relatively easy to avoid the methodical errors. For example, the maximum sampling rate of the ADC 10831 is 74 kHz; thus, the resulting errors are low. You could also use a 10-bit 10732 ADC, a differential-input, single-supply device. For higher frequency and better resolution, you could use a 12-bit 12130 ADC, which has 14- $\mu$ sec throughput time. You can use any embedded-system  $\mu C$  for the method. (DI #2244) EDN

To Vote For This Design, Circle No. 402



A time-domain DSP algorithm measures natural frequency and damping in electromechanical systems.





# $\mu$ C provides wireless keypad control

LLOYD KHUC, MOTOROLA INC, AUSTIN, TX

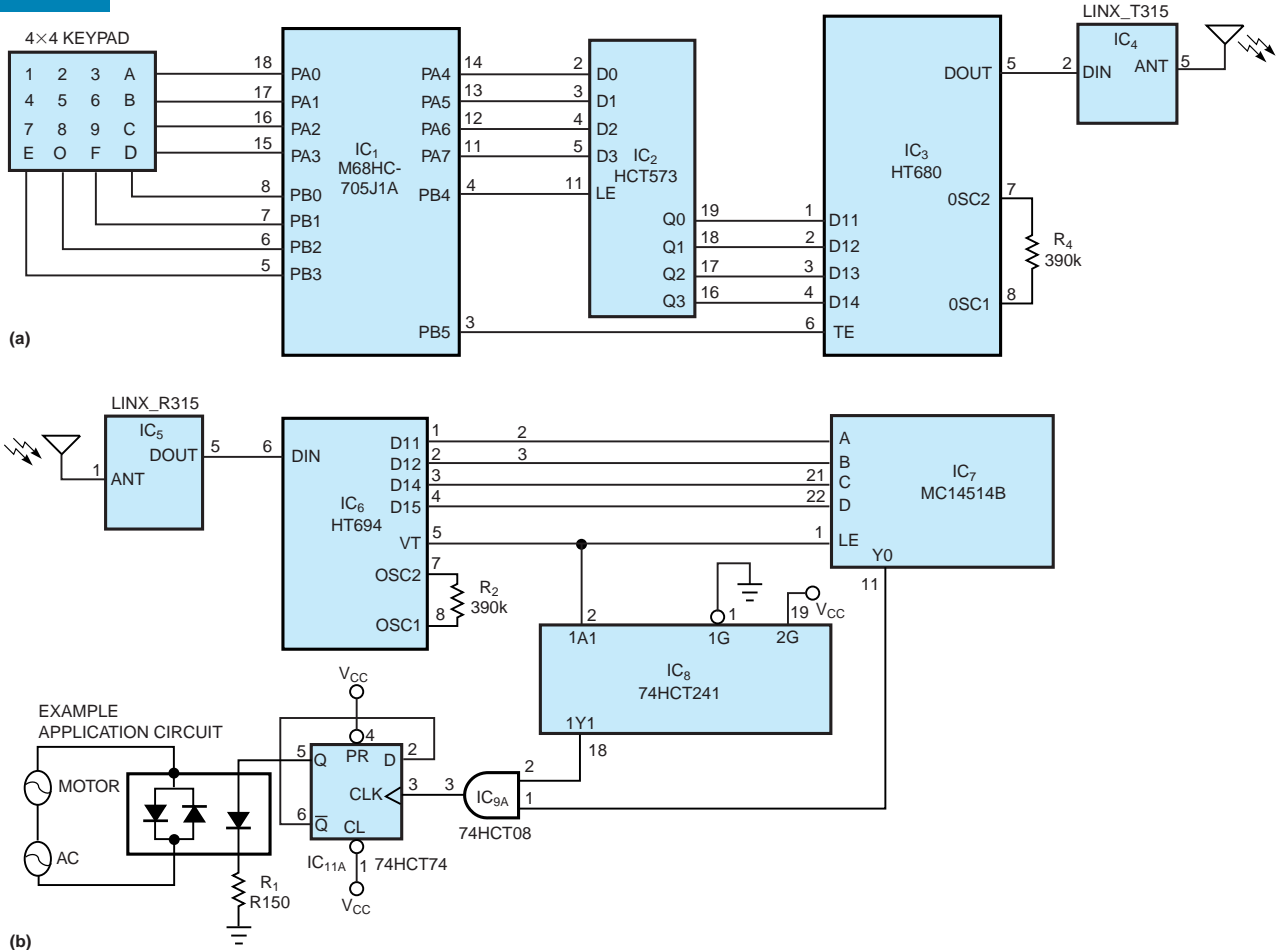
The circuit in **Figure 1** is a simple, 4×4 keypad remote-control system. A 68HC705J1A  $\mu$ C, IC<sub>1</sub> (**Figure 1a**), costs less than \$1 and controls the keypad functions. When you depress any key, the  $\mu$ C provides a 4-bit hex-data output and then enables a latch signal to latch the data into IC<sub>2</sub>. Next, the  $\mu$ C enables IC<sub>2</sub> to transmit the signal to the IC<sub>3</sub> encoder to convert the 4-bit hex data to serial data to send to IC<sub>4</sub>. IC<sub>4</sub>, the RF data-transmitter module, mixes the serial data with the 315-MHz carrier frequency to transmit.

In the receiver circuit (**Figure 1b**), the IC<sub>5</sub> data-receiver module removes the 315-MHz carrier frequency from the receiver signal, and IC<sub>6</sub> decodes the serial data into a 4-bit hex

parallel data output. IC<sub>7</sub> converts the 4-bit hex data into 16 data bits to control 16 application circuits. IC<sub>3</sub> and IC<sub>6</sub> are a matching encoder/decoder pair that eliminates any unwanted interference frequencies. IC<sub>4</sub> and IC<sub>5</sub> are a matching RF data-transmitter/receiver pair with a 315-MHz carrier frequency. You can download the assembler code for the  $\mu$ C system from EDN's Web site, [www.ednmag.com](http://www.ednmag.com). At the registered-user area, go into the Software Center to download the files from DI-SIG, #2261. (DI #2261). EDN

To Vote For This Design, Circle No. 405

FIGURE 1



A simple 4×4 keypad gives you wireless control of 16 lines, using an inexpensive  $\mu$ C, a few logic blocks, and a transmitter/receiver module pair.