design ideas

Edited by Bill Travis

Key-reading circuit saves I/O pins

Gustavo Santaolalla, Digital Precision Systems, Buenos Aires, Argentina

Some microcontroller applications usually use too many I/O pins to read keys or onboard switches; in many cases, few pins remain available for other uses. Some alternative ways to read keys yield more free pins. First, consider some ways to effect key reading. **Table 1** presents a comparison of four methods with references to circuit configurations (**figures 1**, **2**, **3**, and **4**). As you can see, the best choice for reading many keys is to use the A/D converter that is inherent in many microcontrollers. This option needs many lines of code and is not amenable to resistive buttons, such as flexible key pads (**Reference 1**). Another option is to read one key with one I/O port, but it needs as many pins as the keys to read.

Figure 4 shows another possibility, the subject of this Design Idea. In this configuration, the microcontroller reads six keys with only three lines of I/O pins. This method entails the use of a few external components. The idea is to turn one of the three lines into output, set it at logic 1, and use the other two lines as inputs. Then, the microcontroller scans each input for a logic 1, and, if it finds it, a key press has occurred. If not, it turns the next line into an output, sets it, and turns the other two into inputs, and so on. In this way, you can confirm each time a key press takes place. R_4, R_5, and R_6 are current-limiting resistors, and R_1, R_2, and R_3 are simple pull-down resistors. The circuit uses three LEDs for debugging.

Listing 1 shows the complete program. An MC68HC908JK3 tested the software, but the routine is probably applicable to other microcontrollers. The program has no debounce function; you

(continued on pg 120)

TABLE 1—VARIOUS KEY-READING SCHEMES

	No. of keys	No. of pins	Notes
Figure 1	N	No. of pins	16 pins=16 keys
Figure 2	N	N/8	With ADC, 16 keys=two pins
Figure 3	$N_{IN}XN_{OUT}$	NIN+NOUT	16 keys=eight pins (four inputs and four outputs)
Figure 4	N	N	16 keys=five pins (four more keys available)

Publish your Design Idea in *EDN*. See the What's Up section at www.edn.com.

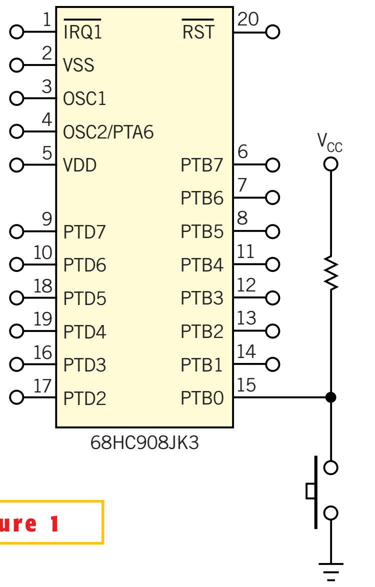


Figure 1

In this simple configuration, you need as many I/O pins as you have keys.

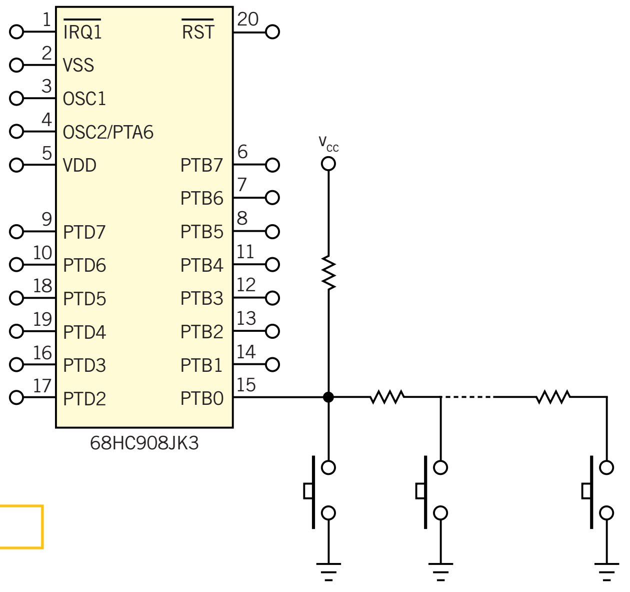
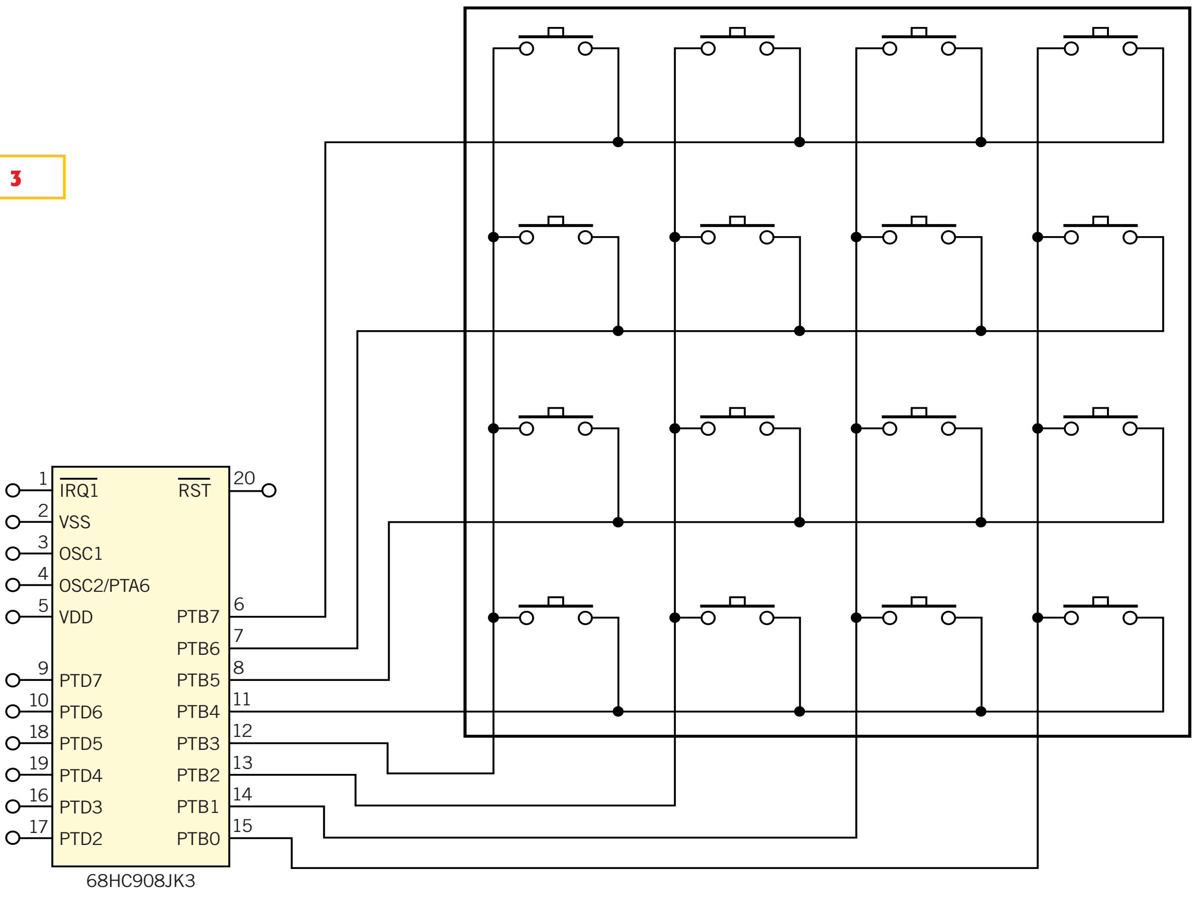


Figure 2

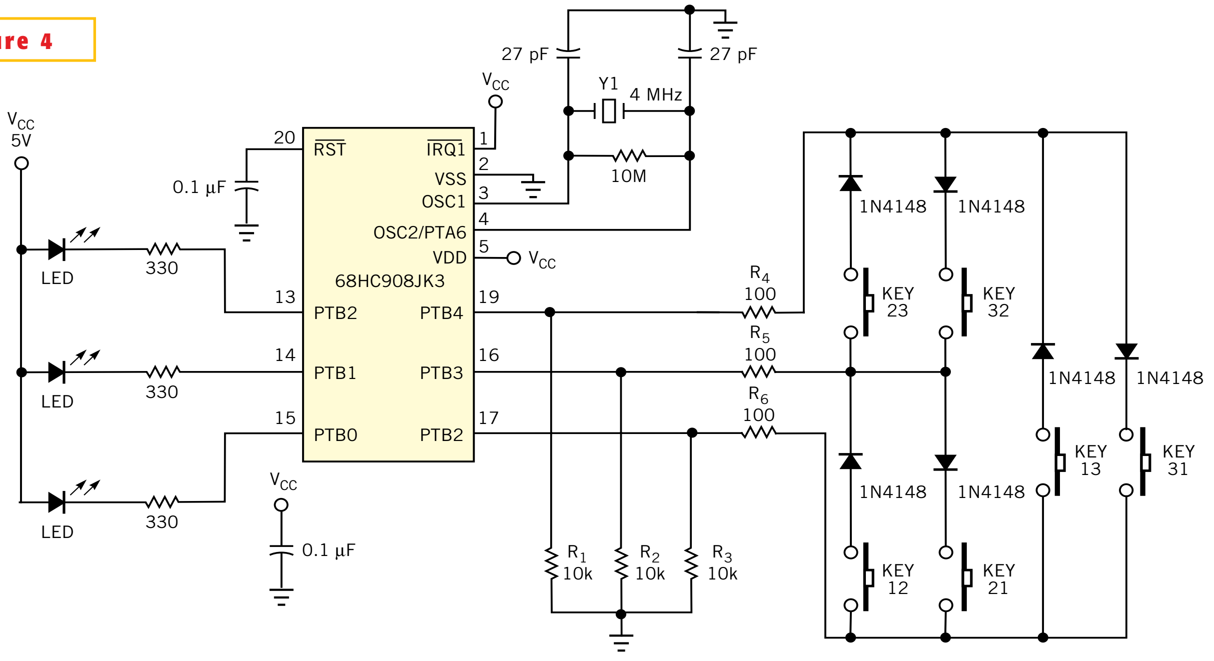
Using the microcontroller's internal ADC, you need only two I/O pins to control 16 keys.

Figure 3



In a matrix configuration, you need eight I/O pins to control 16 keys.

Figure 4



With an expansion of this scheme, five I/O lines control 16 keys with room to add four more keys.

(continued on pg 117)

can add it for a real application. The program shows the variable KeyVal on three LEDs. You can probably write more efficient code that that of **Listing 1**; the code shown is just for testing. **Table 1** shows how many pins you need to read 16 keys. As you can see, with the circuit in **Figure 4**, you need only five pins, but you can add four more keys. You can download **Listing 1** from the Web version of this Design Idea at www.edn.com.

REFERENCE

1. Motorola application note AN1775, www.motorola.com.

Is this the best Design Idea in this issue? Select at www.edn.com.

LISTING 1—PROGRAM TO READ SIX KEYS AND DISPLAY RESULTS ON SIX LEDs

```
;**********************************************************
; 6stones.asm
; This program reads 6 keys and shows their value on six leds
; v1r0 220802 for D.I.
; Assembler: CASM08W from P&E Microcomputers Systems
; Target: HC908JK3
;**********************************************************

;**********************************************************
; EQUS
;**********************************************************

PTA    EQU $0000    ; Ports and data direction
PTB    EQU $0001
PTD    EQU $0003
DDRA   EQU $0004
DDRB   EQU $0005
DDRD   EQU $0007

CONFIG1 EQU $001F   ; System configuration register
CONFIG2 EQU $001E   ; System configuration register

;equs PORT D:
k1          equ    2
k2          equ    3
k3          equ    4

;alias PORT D:
PKey           equ    PTD

;alias DATA DIRECTION PORT D:
DPKey          equ    DDRD

;Keys values
Key12          equ    1
Key13          equ    2
Key21          equ    3
Key23          equ    4
Key31          equ    5
Key32          equ    6

FlashBegin    equ    $EC00
RamBegin      equ    $0080

;Reserve de variables

          ORG    RamBegin
RamBeg         equ    $
KeyVal       rmb    1           ;key value
Last_Var       equ    $

          ORG    FlashBegin

;**********************************************************
; RESET
;**********************************************************
Reset          equ    $
          jsr    Init_Hard           ;init ports
          jsr    Init_Soft           ;init soft
          cli

;**********************************************************
; MAIN PROGRAM
;**********************************************************
Main           equ    $
          jsr    ReadKeyPad            ;read keys
          tsta
          beq     Main                ;no key pressed
          sta     PTB                ;view KeyVal - just for debug
          jmp     Main

ReadKeyPad     equ       $
          clr      KeyVal
          bset      k1,DPKey             ;k1 output
          bclr     k2,DPKey          ;k2 input
          bclr     k3,DPKey          ;k3 input
          bset      k1,PKey          ;k1 = 1
          brclr    k2,PKey,NOK12        ;k2 pressed ?
          brset     k2,PKey,*         ;yes, wait for unpressed
          lda      #Key12           ;accA = Key12
          bra      ENDRKP            ;return
NOK12        brclr    k3,PKey,NOK13       ;k3, pressed ?
          brset     k3,PKey,*         ;yes, wait for unpressed
          lda      #Key13           ;accA = Key13
          bra      ENDRKP            ;return

NOK13        bclr      k1,DPKey            ;k1 input
          bset      k2,DPKey           ;k2 output
          bclr     k3,DPKey          ;k3 input
          bset      k2,PKey          ;k2 = 1
          brclr    k1,PKey,NOK21        ;k1 pressed ?
          brset     k1,PKey,*         ;yes, wait for unpressed
          lda      #Key21           ;accA = Key21
          bra      ENDRKP            ;return
NOK21        brclr    k3,PKey,NOK23       ;k3, pressed ?
          brset     k3,PKey,*         ;yes, wait for unpressed
          lda      #Key23           ;accA = Key23
          bra      ENDRKP            ;return

NOK23        bclr      k1,DPKey            ;k1 input
          bclr     k2,DPKey          ;k2 input
          bset      k3,DPKey           ;k3 output
          bset      k3,PKey          ;k3 = 1
          brclr    k1,PKey,NOK31        ;k1 pressed ?
          brset     k1,PKey,*         ;yes, wait for unpressed
          lda      #Key31           ;accA = Key31
          bra      ENDRKP            ;return
NOK31        brclr    k2,PKey,NOK32       ;k2, pressed ?
          brset     k2,PKey,*         ;yes, wait for unpressed
          lda      #Key32           ;accA = Key32
          bra      ENDRKP            ;return
NOK32        equ    $
ENDRKP        rts

Init_Hard     equ    $          ;1 -> output  / 0 -> input
          lda    #%11111111     ;as output
          sta    DDRA
          lda    #%11111111     ;as output
          sta    DDRB
          sta    PTB
          lda    #%11100011     ; k1,k2,k3 as inputs
          sta    DDRD
          rts

Init_Soft    equ    $
          lda    #%00000001
          sta    CONFIG1       ;cop disabled
          rts

not_used       rti

;**********************************************************
; VECTORS
;**********************************************************
          org    $FFDE
          fdb    not_used      ;adc conversion complete
          fdb    not_used      ;keyboard

          org    $FFF2
          fdb    not_used      ;tim overflow
          fdb    not_used      ;tim channel 1
          fdb    not_used      ;tim channel 0
          fdb    not_used      ;not used
          fdb    not_used      ;irq
          fdb    not_used      ;swi
          fdb    Reset
```

Synchronous buck circuit produces negative voltage

John Betten, Texas Instruments, Dallas, TX

Many electronic systems require both positive and negative voltages to operate properly. Generating an efficient, low-voltage positive output from a higher voltage input typically entails the use of a synchronous buck regulator. But when generating a negative output voltage from a positive input voltage, you'd typically use a flyback topology, especially at higher output currents. The operation and control characteristics of a synchronous buck and a negative flyback (also called a buck-boost) differ significantly. **Figure 1** shows the basic components that a negative flyback circuit requires. When FET Q_1 turns on, the input voltage appears across inductor L_1 with no input current going to the load at this point. All the output current delivered to the load at this time comes from output capacitor C_1, because diode D_1 is reverse-biased. The current in the inductor continues building until the control circuit determines the proper time to switch off FET Q_1. At that point, the voltage polarity across inductor L_1 reverses in an attempt to maintain current flow, pulling the top side of the inductor

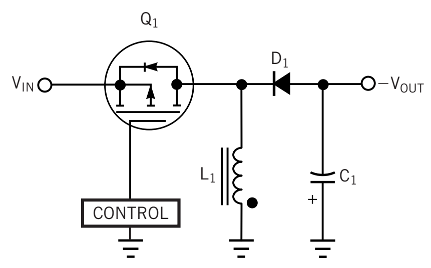


Figure 1

This flyback topology produces negative output voltage from positive inputs.

negative with respect to ground and forcing diode D_1 to conduct. The output voltage goes negative to within a diode drop of the inductor voltage.

The duty cycle at which the control circuit operates also differs from that of a synchronous buck. Although the operating duty cycle of a synchronous buck is $D=V_{OUT}/V_{IN}$, the negative flyback operates at $D=V_{OUT}/(V_{OUT}-V_{IN})$. For example, if the desired output voltage is half the input voltage, the synchronous buck runs at 50% duty cycle, whereas the negative flyback runs at 33% duty cycle. The comparisons between the simple negative flyback circuit of **Figure 1** and the synchronous-buck-controller negative flyback circuit in **Figure 2** are straightforward. In **Figure 2**, FET Q_2 mirrors the function of diode D_1 in **Figure 1** but with a decrease in the forward drop that occurs in the diode. This lower drop significantly improves efficiency. Diode D_3 conducts during the small dead time, when both FETs Q_1 and Q_2 are off, further reducing losses. The feedback voltage appears at the output ground through resistor R_1, because the control circuit is referenced to the negative output voltage. R_2 typically sets the output voltage to the desired level, because it does not change the feedback compensation network, as changing R_1 would. Desired changes to the input voltage, the output voltage, or both may necessitate an inductor-value change. The minimum inductor value is:

$$L_{MIN}=\frac{|V_{OUT}|(V_{INMAX})^2}{2f_{MIN}I_{OUTMIN}\left(|V_{OUT}|+V_{INMAX}\right)^2}.$$

Take note of certain limitations with using the controller in this type of implementation. Because the control circuit is referenced to the negative output-voltage rail, the controller must have an input-voltage rating greater than $V_{IN}+|V_{OUT}|$. The controller must also be rated for V_{IN} (minimum), which occurs at system power-up when the output voltage is zero. A controller that operates over a wide input-voltage range typically works best. The FET's drain-to-source rating must also withstand $V_{IN}+|V_{OUT}|$, and the FET carries peak currents that are greater than twice the output current. Low-resistance, fast-switching FETs produce the lowest losses. High efficiency is the major advantage of this circuit. Because the circuit uses n-channel FETs, as opposed to higher resistance and costlier p-channel parts, the circuit achieves peak efficiencies greater than 90%.

Is this the best Design Idea in this issue? Select at www.edn.com.

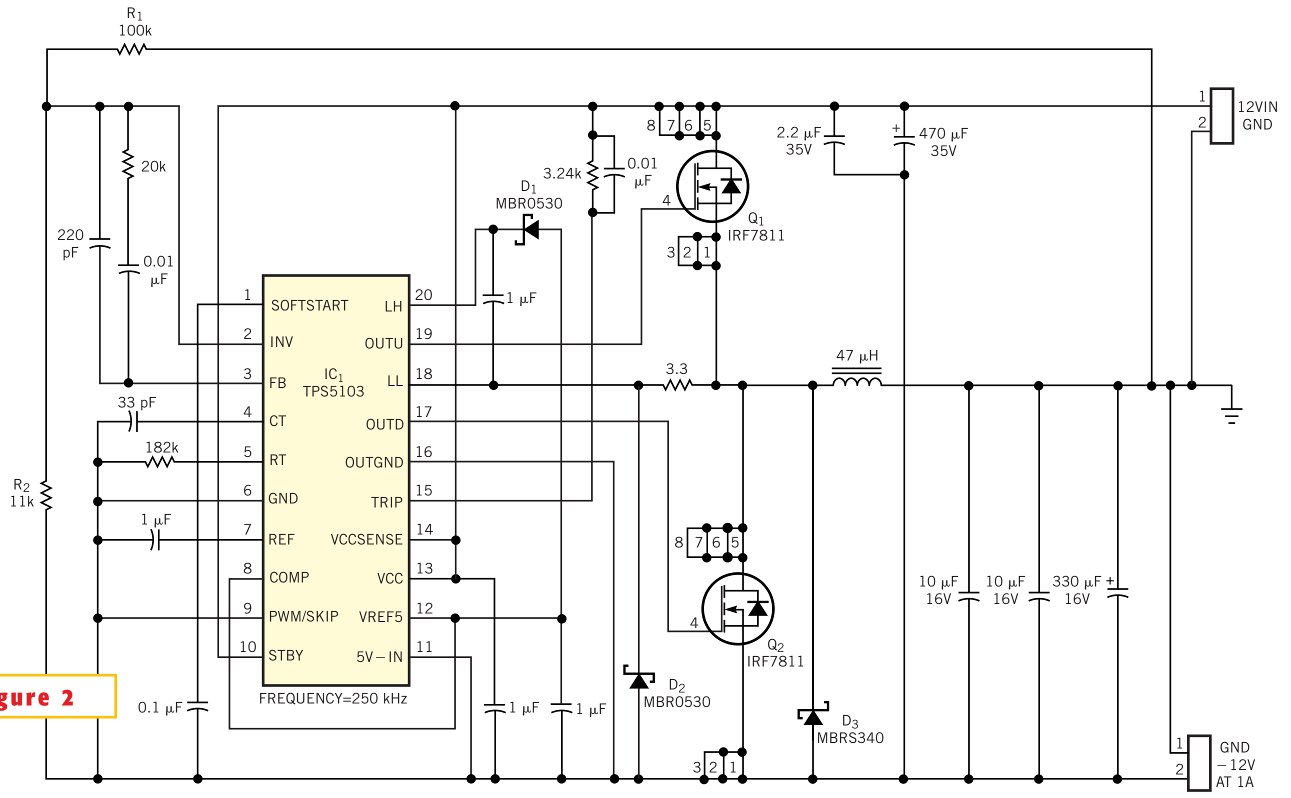


Figure 2

A synchronous buck controller forms the heart of this negative-flyback configuration.

Rotary encoder mates with digital potentiometer

Peter Khairolomour, Analog Devices, San Jose, CA

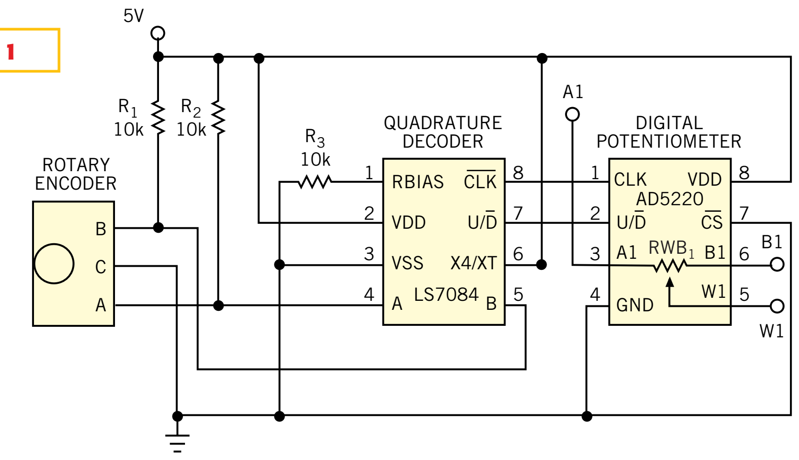
In developing electronic systems, designers look for products or ideas that may benefit from the better performance, smaller size, lower cost, and improved reliability that an IC can offer. Toward that end, the digital potentiometer emerged as an alternative to its mechanical counterpart, the mechanical potentiometer. The digital potentiometer offers most of the cited advantages but falls short for users of mechanical potentiometers, who require a simple rotary interface for front-panel adjustment or calibration without external controllers. The circuit in **Figure 1** represents an attempt to combine the best of both worlds: the simplicity of a rotary interface and the performance of a digital potentiometer. The rotary encoder in this circuit is the RE11CT-V1Y12-EF2CS from Switch Channel (www.switchchannel.com). This type of rotary encoder has one ground terminal, C, and two out-of-phase signals, A and B (**Figure 2**). When the rotary encoder turns clockwise, B leads A (**Figure 2a**), and, when it turns counterclockwise, A leads B (**Figure 2b**).

Signals A and B of the rotary encoder pass through a quadrature decoder (LS7084 from LSI Computer Systems, www.lsisci.com), which translates the phase difference between A and B of the rotary encoder into a compatible output, $\overline{\text{CLK}}$ and U/$\overline{\text{D}}$, that the AD5220 can accept. The AD5220 from Analog Devices (www.analog.com) is a 128- step, pushbutton digital potentiometer. It operates with a negative-edge-triggered clock, CLK, and an increment/decrement direction signal, U/$\overline{\text{D}}$. When B leads A (clockwise), the quadrature decoder provides the AD5220 with a logic-high U/$\overline{\text{D}}$. When A leads B (counterclockwise), the quadrature decoder provides the AD5220 with a logic-low U/$\overline{\text{D}}$. The quadrature decoder also produces a clock in synchronism with its output, which also connects directly to the AD5220. You linearly vary the clock's pulse width by adjusting RBIAS.

Aside from decoding the quadrature output of the rotary encoder and providing a clock signal, the LS7084 also filters noise, jitter, and other transient effects. This feature is important for this type of application. Unlike optical encoders, the RE11CT-V1Y12-EF2CS is a low-cost electrical encoder, in which each turn can create some bounce or noise issues because of the imperfect nature of the metal contacts within the switch. The LS7084 prevents such erroneous signals from reaching the AD5220. The operation of the circuit in **Figure 1** is simple. When the rotary encoder turns clockwise, the resistance from the wiper to terminal B1 of the digital potentiometer, RWB_1, increments until the device reaches full scale. Any further turning of the knob in the same direction has no effect on the resistance.

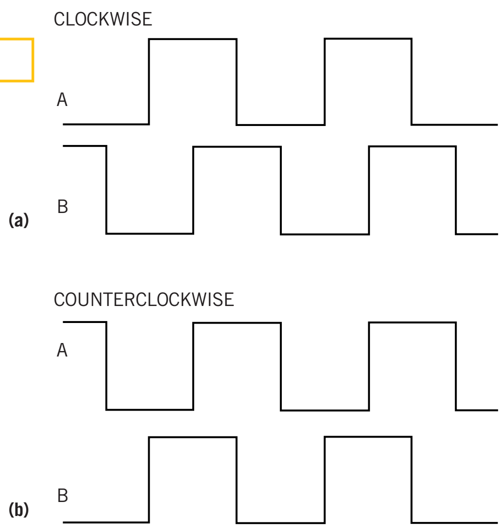
Likewise, a counterclockwise turn of the knob reduces RWB_1 until it reaches the zero scale, and any further turning of the knob in the same direction has no effect. One example of the flexibility and performance this circuit offers becomes apparent when you consider systems with front-panel rotary adjustment. You can lay out the compact digital potentiometer and quad-

Figure 1



A quadrature decoder and a digital potentiometer form a simple rotary-encoder interface.

Figure 2



In clockwise rotation, signal B leads A (a); in counterclockwise rotation, A leads B (b).

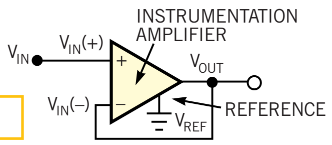
rature decoder anywhere in the system. All the ICs need are two digital control signals routed to the front panel where the rotary encoder is located. This setup proves impervious to interference, noise, and other transmission-line effects that arise in traditional designs with mechanical potentiometers. These designs force the sensitive analog signal to travel all the way to the front of the panel to be processed and then back to its destination.

Is this the best Design Idea in this issue? Select at www.edn.com.

Amplifiers perform precision divide-by-2 circuit

Glen Brisebois and Jon Munson, Linear Technology Corp, Milpitas, CA

Figure 1



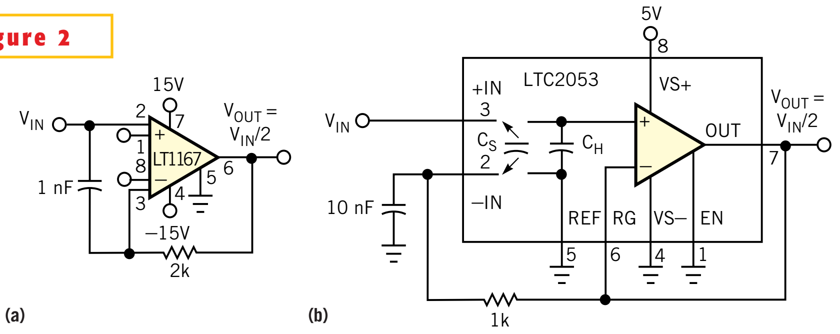
An instrumentation amplifier makes a simple divide-by-2 circuit.

THE CLASSIC IMPLEMENTATION of a voltage-halving circuit uses two equal-value resistors. Using 1% resistors provides a divider output with 2% accuracy. For most applications, this performance is cost-effective and more than adequate. However, when you need extreme precision, this approach requires correspondingly accurate resistors and can become expensive. Putting feedback around a finite-gain instrumentation amplifier yields a divide-by-2 circuit with the added benefit of a buffered output (**Figure 1**). The operation of the circuit is straightforward. The instrumentation amplifier has unity gain, so the voltage it sees across its inputs appears between V_{REF} and V_{OUT}: $V_{OUT}-V_{REF}=V_{IN}(+)-V_{IN}(-)$. But, considering the circuit in **Figure 1**, note that $V_{OUT}=V_{IN}(-)$, and $V_{REF}=0$. Substituting in the first equation, you obtain $V_{OUT}=V_{IN}(+)-V_{OUT}$, $2V_{OUT}=V_{IN}(+)$, or $V_{OUT}=1/2\ V_{IN}(+)$. Thus, you have a divide-by-2 circuit. One of the interesting features of this approach is that the input and the output offsets of the instrumentation amplifier are divided by 2 as well.

You can implement the circuit on the bench using the LT1167 or the LTC2053 instrumentation amplifiers (**Figure 2**). Although benchtests are unnecessary, you can introduce an RC network into the feedback path for noise shaping and to guarantee dominant-pole behavior. To test for LT1167 offset, set $V_{IN}(+)$ to 0V and alternate $V_{IN}(-)$ between 0V and V_{OUT}. This test confirms that the feedback halves the total offset voltage. Dividing 10V to 5V, the LT1167 shows an error of 100 µV. With the more precise LTC2053, the output error in dividing 2.5V to 1.25V is an almost-immeasurable 2.5 µV. Using cold spray and a heat gun, you can degrade this error to 15 µV. However, perhaps equally important are the calculated worst-case results.

Worst-case calculations for the LT1167 show a maximum 1.12-mV error over 0 to 70°C with 10V input and 5V output. This figure constitutes a total error of 224 ppm over temperature. Resistors that guarantee this accuracy would need a maximum tolerance of 112 ppm each over temperature. The error budget of a resistor-divider solution would require an initial ratio match of approximately 50 ppm with a temperature-coefficient match better than 1 ppm/°C. Worst-case calculations for the LTC2053 with 2.5V input and 1.25V output show a maximum 80-µV error over 0 to 70°C. This figure constitutes a total error of 64 ppm over temperature. Resistors that guarantee this accuracy would need a maximum tolerance of 32 ppm each over temperature. The error budget of a resistor-divider solution would require an initial ratio match of approximately 15 ppm (0.0015%) with a temperature-coefficient match better than 0.25 ppm/°C. In either case, resistors of this caliber would be extraordinarily expensive if available at all. Also, the amplifiers provide the additional benefits of high input impedance and output buffering. Moreover, the error calculations include the effects of input offset voltage, bias current, gain error, and common-mode rejection ratio, which a resistor op amp would still have to add.

Figure 2



Practical implementations of the circuit in Figure 1 use the LT1167 (a) and the LTC2053 (b).

Is this the best Design Idea in this issue? Select at www.edn.com.