

designideas

READERS SOLVE DESIGN PROBLEMS

Missing pulse detects position or produces a delay

Michael C Page, Chelmsford, MA

Consider an application that needs a series of pulses to indicate position in which the lack of a pulse “indexes” the count. To achieve that goal, the application uses a rotating, 36-tooth sprocket with one missing tooth. Rotational speed ranges from 500 to 7000 rpm. The mechanism uses an inductive pickup to sense the sprocket’s teeth. With one of the sprocket’s 36 teeth missing, the detector senses 35 pulses, and then a pulse disappears.

Unfortunately, the mechanism frequently breaks down or simply breaks

apart. Because the application uses this wheel just to trick the computer by simulating an operating engine, the application’s designers replaced the rotating gear with a simulator circuit (Figure 1). Given the rotational speed and number of teeth, the maximum pulse frequency is $7000/60 \times 36$, or 4200 Hz. The circuit works well from single stepping to more than 1 MHz before starting to break down. The maximum frequency depends on the logic family and construction methods you use.

Figures 2 and 3 show the outputs

DIs Inside

47 Emulate SPI signals with a digital-I/O card

48 Resistive DAC and op amp form hybrid divider

49 Connect two buttons with just two wires

▶ To see all of EDN’s Design Ideas, visit www.edn.com/designideas.

running at 100 Hz and 1 MHz, respectively. At power-up, capacitor C_1 remains the same, which forces RST on

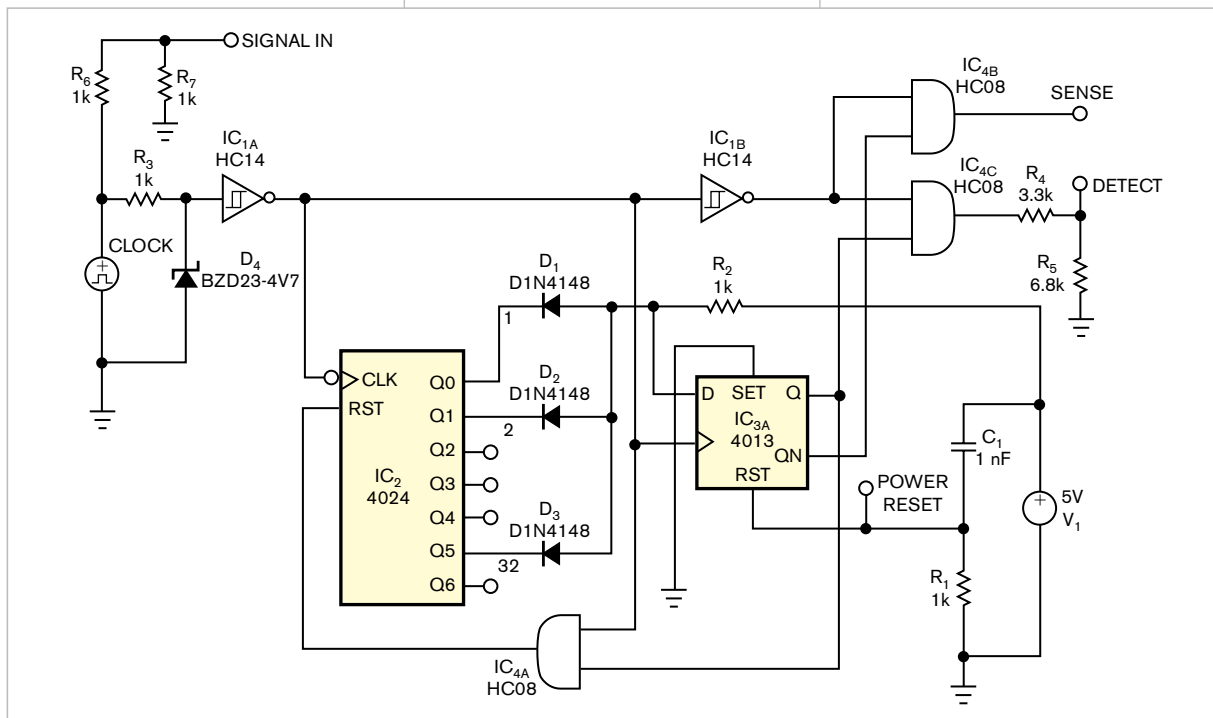


Figure 1 IC₂ combines with three diodes to produce a stream of 36 pulses before resetting.

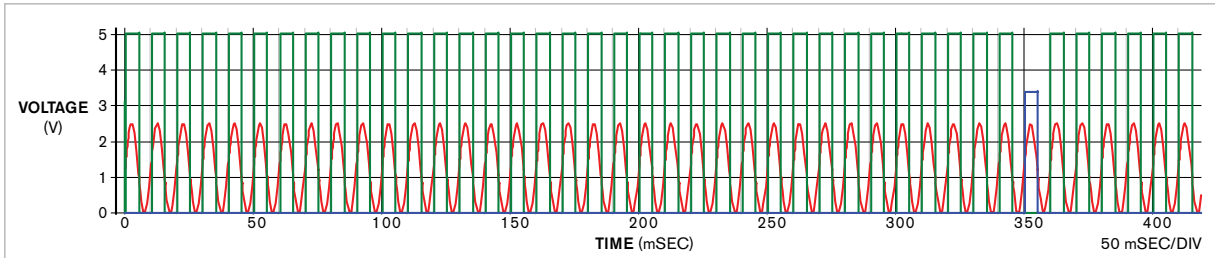


Figure 2 Operating at 100 Hz, the circuit signals include the clock-sine-wave signal (red), the sense-square-wave signal (green), and the detect signal (blue), which indicates the missing pulse.

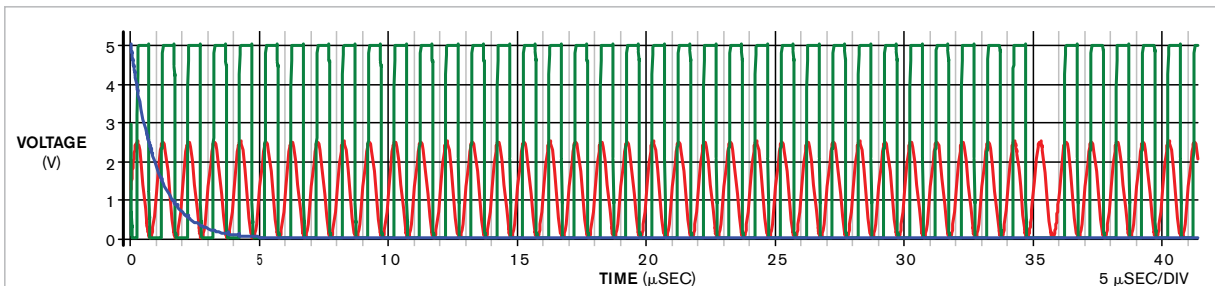


Figure 3 The pulse train at a clock frequency of 1 MHz still shows the missing 36th pulse along with the power-reset signal (blue).

IC_{3A} low. That action puts the D flip-flop into a known state. As C_1 charges through R_1 , the voltage at the power reset falls, letting clock pulses set IC_{3A} 's outputs. You must keep the small value for the C_1/R_1 combination if you use a high input frequency with a low count rate. As **Figure 3** shows, the desired count must exceed the duration of the power reset. The values in **Figure 1** provide a time of approximately $0.66 \times 1 \text{ k}\Omega$ (the value of R_1) $\times 1 \text{ nF}$ (the value of C_1), or $0.66 \text{ }\mu\text{sec}$, leaving a minimum count of approximately three at 1 MHz.

For the clock signal, the circuit uses a sine-wave signal with an amplitude of 5 to 10V from the system. The clock signal goes through R_3 to D_4 and IC_{1A} to produce a 5V square-wave signal. The signal goes to counter IC_2 , and to one input of AND gate IC_{4B} . With the other input of IC_{4B} coming from IC_{3A} 's QN output, which is high from power reset at start-up, the input-pulse train passes through IC_{4B} , which simulates sprocket teeth at the sensor. Resistors R_6 and R_7 halve the clock-signal amplitude just to make the graphics clear at "signal

in." Diodes D_1 , D_2 , and D_3 pull up to 5V through R_2 and form an AND gate to select the desired count. Counter IC_2 's outputs are binary, so, for a 36-tooth sprocket with one missing tooth, outputs Q0, Q1, and Q5 correspond to $1+2+32=35$.

You can produce any count as high as 128 by adding the appropriate diodes on the Q outputs on IC_2 . In other words, you need to generate one missing pulse of 36 to simulate the 36-tooth sprocket. Thus, you select a count of 35; the circuit automatically adds a count of one due to the one-clock delay of the counter. Because you reset IC_2 at power-up, all outputs are low, keeping the D input of IC_{3A} low, with a count of zero.


THE CIRCUIT AUTOMATICALLY ADDS A COUNT OF ONE DUE TO THE ONE-CLOCK DELAY OF THE COUNTER.

As clock pulses continue into IC_2 and when outputs Q0, Q1, and Q5 are all high, with a count of 35, IC_{3A} 's D pin pulls high through R_2 . On the next clock pulse, the Q output of IC_{3A} goes high and the QN output goes low, stopping the pulses from passing through IC_{4B} . This action indicates the missing tooth and produces the sense condition (the missing pulses in **figures 2** and **3**). Meanwhile, the Q output of IC_{3A} 's output goes high, yielding a single detect pulse at IC_{4C} through R_4 and R_5 . On the next clock pulse, with IC_{3A} 's Q output high, IC_2 resets logic zero and is ready for the next count cycle. R_4 and R_5 halve the clock signal just to make the graphics clear at "detect."

The 4024 is an eight-stage binary-ripple counter. You can replace it with a 4040 counter to achieve a count of 2048, and you can cascade counters to get longer counts or delays. The 4040's pinout differs from that of the 4024, but their operation is identical. Some systems have an extra tooth instead of a missing tooth, and some have multiple missing teeth at odd locations around the sprocket, all waiting for replacement by this simple circuit. **EDN**

Emulate SPI signals with a digital-I/O card

Andy Street, Autoliv Electronics, Lowell, MA

 A design-verification tester for millimeter-wave SOC (system-on-chip) devices needed to combine switching, electrical measurements, temperature measurement, a parallel-digital interface, and a serial-digital interface into one instrument. To minimize rack space, the circuit uses an Agilent Technologies (www.agilent.com) 34980A multifunction mainframe because its plug-in cards could support a force/sense dc matrix and multiplexed temperature measurements. The addition of an Agilent 34950A 64-bit digital-I/O card formed the basis of a system that could provide both an SPI (serial-peripheral-interface) bus and a simple parallel bus. The 34950A groups its I/O lines into two banks of four 8-bit channels. It provides 64 kbytes of memory per bank for pattern generation or signal capture. It also has three I/O lines per bank for handshaking.

YOU CAN STORE A MAXIMUM OF 32 TRACES IN THE PATTERN RAM PER BANK.

However, the card's handshake lines provide insufficient control for implementing SPI transactions. To get adequate control, you can emulate the SPI bus using three of the data-I/O lines.

Motorola (www.motorola.com) microcontrollers first used the SPI master-slave protocol. Today, it's become the control interface in a variety of ICs, including PLLs (phase-locked loops) and RF ASICs (references 1 and 2). The SPI bus uses the clock, SS (slave-select), MOSI (master-out/slave-in), and MISO (master-

in/slave-out) lines. The clock line is a signal from the master to the slave. All SPI signals are synchronous with this clock. The SS line selects the slave for communication. The SPI specification defines four modes of operation, which effectively specify the clock edges for toggling and sampling and the clock-idle level. The specification makes no requirements on voltage levels or data rates, and many SPI implementations can clock in excess of 10 MHz. **Figure 1** shows a block and timing diagram of the 34950A's Bank 1, configured for synchronous, buffered output. H0 through H2 denote the handshake lines. The **figure** also shows an 8-bit SPI transaction for reference.

You cannot use the 34950A's handshake lines to emulate all modes of the SPI bus because the bus latches data on the falling edge of the clock, making the bus unsuitable for slaves that use the rising edge. Inverting the clock polarity is not a solution because you may lose the last data bit. Furthermore, if you write a number of transactions to a slave, you must store each transaction as a separate trace memory in the 34950A. Although each bank supports 64k×8 bits, you can store a maximum of 32 traces in the pattern RAM per bank, thereby limiting the number of SPI transactions. In addition, the card lacks a sequencer, so you cannot download a number of bit patterns and then play them in sequence. You must load each pattern into the I/O card's memory and then play each pattern under SCPI (standard commands for programmable instruments) from a host computer, slowing transactions.

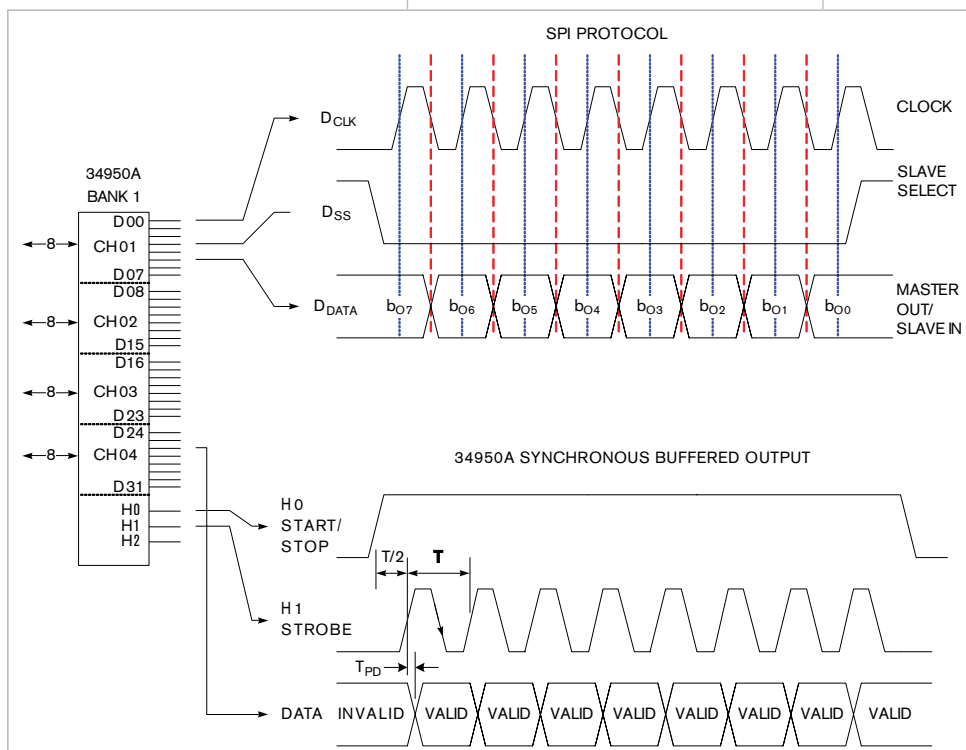


Figure 1 The 34950A synchronous buffered output uses the falling edge, making it unsuited to rising-edge SPI implementations.

ta-I/O lines to emulate the SPI clock, SS, and MOSI. The software driver for the I/O cards then has the responsibility of translating the data to be sent into an SPI-compatible bit stream. **Listing 1**, which is available at www.edn.com/090917dia, contains the algo-

rithm in pseudocode, which translates a hexadecimal string, DH, of characters to an SPI signal. LD, LSS, and LCLK are integers to define which data outputs represent the MOSI, CLK, and SS, respectively.

Assuming a 24-bit register write

with two bits of overhead for the SS prefix and postfix, the 64-kbyte memory can support more than 1000 SPI transactions. The approach has two additional advantages: The three lines that form the SPI bus are under software control, which provides cabling flexibility, and the implementation can support multiple slaves through the use of additional SS lines. **Figure 2** shows an MSO (mixed-signal-oscilloscope) screen that shows the SPI transaction. The SPI clock rate is 5 MHz, which the 34950A's internal 10-MHz clock limits. The different payload sizes correspond to writing data to 16- and 24-bit registers within the slave. **EDN**

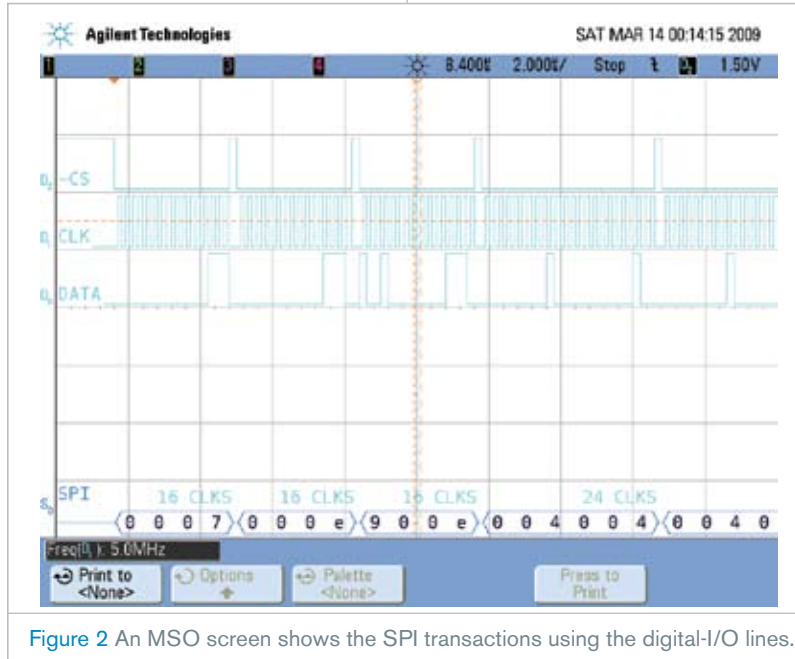


Figure 2 An MSO screen shows the SPI transactions using the digital-I/O lines.

REFERENCES

- Leens, Frederic, "An Introduction to I²C and SPI Protocols," *IEEE Instrumentation & Measurement*, Volume 12, No. 1, February 2009, pg 8, www.imm.ieee-ims.org/docs/ColumnsFebruary2009.pdf.
- "SPI Block Guide V04.01," Freescale Semiconductor, July 2004, www.freescale.com/files/microcontrollers/doc/ref_manual/S12SPIV4.pdf.

Resistive DAC and op amp form hybrid divider

Marián Štofka, Slovak University of Technology, Bratislava, Slovakia

▶ A resistive DAC in a resistive-feedback loop of an op amp lets you create an analog-digital-analog divider. The resistance, R_{WA} , between the W and A terminals of the Analog Devices (www.analog.com) AD5293 (**Figure 1**) decreases linearly with increasing the digital-control data, D:

$$R_{WA}(D) = \frac{1024-D}{1024} \times R_{AB},$$

and the value of the R_{WB} , the resistance between the W and B terminals of the DAC, rises proportionally to D as

$$R_{WB}(D) = \frac{D}{1024} \times R_{AB}.$$

R_{AB} is a constant value of resistance be-

tween the ends of the digital potentiometer. The circuit uses resistance R_{WA} as a feedback resistor, and resistance R_{WB} connects between the inverting input of the op amp and ground. The voltage gain of the noninverting amplifier becomes

$$A_V = 1 + \frac{R_{WA}}{R_{WB}} = \frac{1024}{D}.$$

The output voltage is

$$V_{OUT} = V_{IN} \times \frac{1024}{D}.$$

Both the input voltage and the digital-input data can be time variables, and the clock frequency for fetching digital-input data can be as high as 50 MHz.

The potentiometer's data sheet provides the ground-referred parasitic capacitances at the A, B, and W terminals of the potentiometer. Thorough measurement of the capacitances at these terminals provides enough data to determine capacitances between the terminals. An evaluation of the measured data shows that the direct capacitance between the A and W terminals at the midscale position of the wiper is just 2.4 pF:

$$C_{AW}(X = 1/2) = 2.4 \text{ pF}.$$

If you assume that the five segments of the potentiometer are ordered topologically into a chain, then the direct intercapacitance between the A and B ends of potentiometer is

$$C_{AB}(X = 1/2) = 1/2 C_{AW}(X = 1/2) = 1.2 \text{ pF}.$$

The capacitance per segment of the five segments of the potentiometer is

$$C_{SEGM} \approx 5C_{AB}(X = 1/2) \approx 6 \text{ pF},$$

where $X=1/2$ denotes the midscale of the resistive DAC.

The five-step distributed RC line of the potentiometer has a time constant of

$$\tau_{SEGM} = \frac{R_{AB}}{5} \times C_{SEGM} = R_{AB} \times C_{AB} = 24 \text{ NSEC},$$

where R_{AB} is 20 kΩ. The ground-referred wiper capacitance, C_W , of 40 pF is much higher than the intercapacitances and creates a time constant:

$$\tau_W \approx R_{WB} \times C_W.$$

The feedback network of the amplifier is frequency-compensated for $\tau_{SEGM} \approx \tau_W$. Thus, you can calculate the value of R_{WB} as 600Ω, meaning that the voltage gain of the amplifier, A_v , is 32.3. For gains higher than 32.3, the effect of C_W becomes negligible, and you need not bother about amplifier stability. To suppress the derivative behavior of the amplifier for gain values of two to 32.3, you can add a 40-pF compensating capacitor in parallel to feed back part of the potentiometer. The amplifier thus has an integrating character for all gains down to a value of two.

You fetch the divisor, Y , which is a digital-data word, D , through a stan-

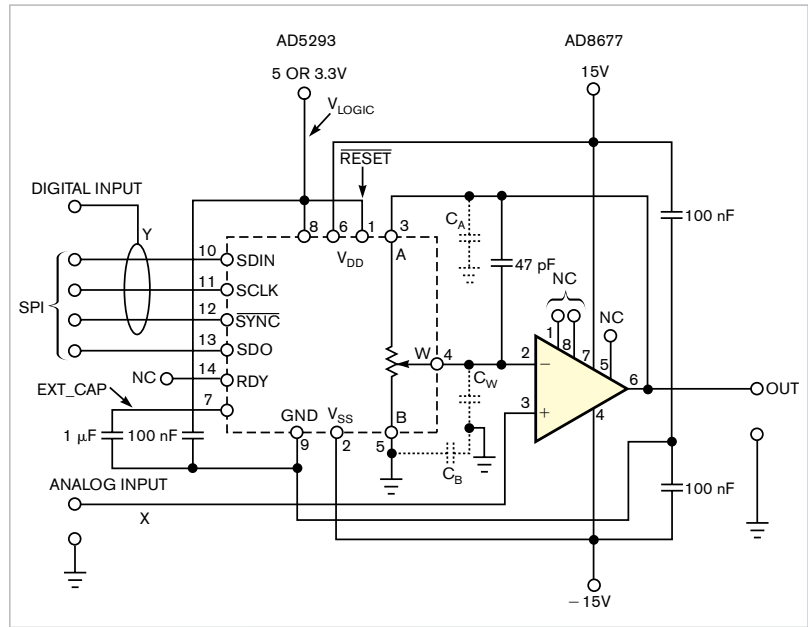


Figure 1 The resistive DAC-potentiometer forming the feedback for an op amp controls the op amp's gain as inversely proportional to the digital-input-data word. The circuit thus becomes a two-quadrant divider.

dard SPI (serial-peripheral interface). After power-on, you must initially neutralize the write-in protection of the resistive DAC. You have to first program the control bit C_1 to the value of one, whereas it is zero by default. You achieve this task by clocking in the word containing C_3 , C_2 , C_1 , and C_0 ,

which equals 0110, and you put the desired C_2 and C_1 values at data positions D_2 and D_1 . After performing these steps, you change the wiper position in which the control bit is C_3 , C_2 , C_1 , and C_0 , which equals 0001, and the data bits, D_9 to D_0 , represent the gain as $1024/D$. **EDN**

Connect two buttons with just two wires

Fikret Yilmaz, Mobil Elektronik, Istanbul, Turkey

Sometimes, you need to read the status of pushbuttons that are as much as 5m away from your electronic circuit. That task is easy if you have just one button. You simply design a constant-current source, connect the current line from your button, and measure the current in the line. If you press the button, current flows through it. Otherwise, current does not flow.

Problems occur, however, when you need to read two or more buttons. Several approaches to this problem are available. For example, you could use an RS-485 interface with two wires for communication and two for power. Alternatively, you could use a single-wire connection

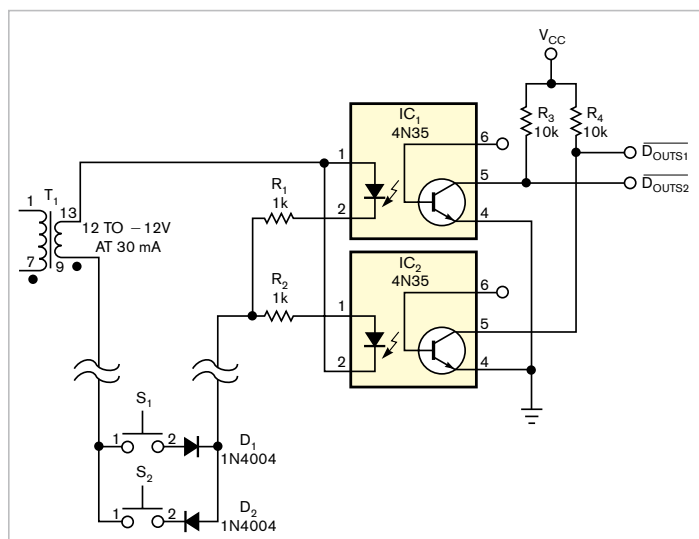


Figure 1 You can connect two buttons using diodes.

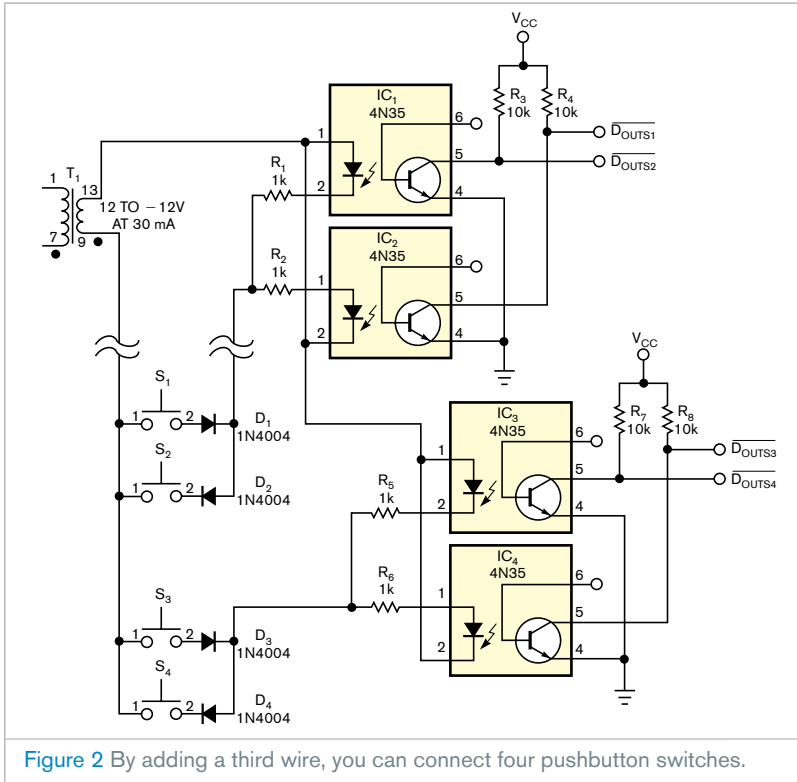


Figure 2 By adding a third wire, you can connect four pushbutton switches.

with one wire for communication and two for power. Another option is to use separate wires for each button. In that case, you would use one more wire than there are buttons. Finally, you could use a POE (power-over-Ethernet) approach, employing four wires for communication and power. All of these approaches require a button reader or a controller, which you must program, adding complexity and cost.

The circuit in **Figure 1** shows you how to connect two buttons using diodes. Because the diodes steer the current, the circuit needs just two wires. On a positive cycle from the transformer secondary and with switch S₂ closed, current flows through IC₁, R₂, and D₂. Thus, output D_{OUTS2} pulls low. Conversely, if S₁ closes on a negative cycle, then current flows through D₁ to R₁ and IC₂, which pulls D_{OUTS1} low. The circuit in **Figure 2** extends the concept to four pushbutton switches by adding a third wire. **EDN**