design ideas

Edited by Bill Travis

Calibrate scope jitter using a transmission-line loop

David Cuthbert, Micron Technology, Boise, ID

Digital-clock-period jitter is the variation in the period of a clock cycle compared with a nominal (average of many cycles) clock period. To accurately measure period jitter using an oscilloscope, you must subtract the oscilloscope jitter from the measured jitter. However, oscilloscopes rarely have a jitter specification, so you must determine the oscilloscope jitter. One method of measuring oscilloscope jitter is to use the oscilloscope to measure the jitter of a pulse generator with known jitter. The measured jitter, assuming the jitter has a Gaussian distribution, is $\sqrt{(\text{scope jitter})^2+(\text{generator jitter})^2}$. Rearranging the formula to solve for oscilloscope jitter, the scope jitter is $\sqrt{(\text{measured jitter})^2-(\text{generator jitter})^2}$. The ideal generator for measuring oscilloscope jitter would have zero jitter. **Figure 1** is a circuit for generating a calibration signal with near-zero timing jitter.

A transmission-line delay loop creates a delayed pulse at the oscilloscope. **Figure 2** shows the circuit in operation. You set the oscilloscope for internal triggering on the first pulse and then configure the scope to measure the time between the first pulse and the delayed pulse. You set the trigger hold-off such that the oscilloscope always triggers on the first pulse. The second pulse in each set of pulses is the first pulse delayed through the transmission-line delay loop. Two 50Ω coaxial transmission lines implement the circuit and connect to the oscilloscope using two BNC T adapters. The line from the 50Ω generator to the 50Ω oscilloscope can be of any length. The length of the delay-loop line determines the delay between the first pulse and the delayed pulse. You set the generator's pulse period to approximately five times the loop delay and the generator's pulse duration to approximately one-half the loop delay.

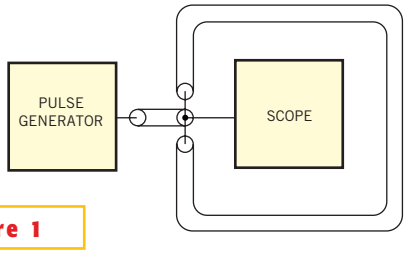


Figure 1

You can use a transmission-line delay loop to accurately measure an oscilloscope's jitter.

The waveforms in **Figure 2** represent a pulse period of 62 nsec and a pulse duration of 6 nsec. The delay loop consists of 2.4m of RG-58 coax and creates a delayed pulse 13 nsec after the first pulse. The 13-nsec delay is equivalent to calibrating the oscilloscope with a zero-jitter, 77-MHz signal. The impedance relationships are such that the delayed pulse has the same amplitude as the first pulse. With the 50Ω pulse generator set for 1V into 50Ω, the generator sends a 1V incident pulse toward the oscilloscope. As the incident pulse "sees," the node at the oscilloscope consists of the 50Ω scope in parallel with the two 50Ω ends of the delay loop. Therefore, the impedance at this node is 16.7Ω, and the reflection coefficient is −0.5. The impedance mismatch causes a pulse amplitude of 0.5V to appear at the oscilloscope and a reflected pulse of −0.5V to travel to the generator, where it dissipates. Two 0.5V pulses travel in opposite directions through the delay loop and meet 13 nsec later at the oscilloscope, forming a 0.5V pulse with a source impedance of 25Ω.

The 50Ω oscilloscope, in parallel with the 50Ω line to the generator, forms a 25Ω load that matches the 25Ω pulse source impedance. The delayed-pulse amplitude at the oscilloscope is 0.5V, and a 0.5V pulse travels to the pulse generator, where it dissipates. You can analyze the circuit's action by keeping track of where the energy goes. The 1V generator sends a 20-mW pulse down the 50Ω line. When this pulse encounters the oscilloscope and delay loop, the energy splits four ways. In this split, 5 mW reflects back to the generator where it dissipates, 5 mW dissipates in the oscilloscope, and 5 mW enters each end of the delay loop. When the two 5-mW pulses exit the delay loop, 5 mW dissipates in the oscilloscope and 5 mW travels to the generator, where it dissipates. You can calibrate several oscilloscopes and one pulse generator with the delay loop. Because the loop has near-zero jitter, the jitter that an os-

cilloscope measures is virtually all oscilloscope jitter. The loop allowed a generator with 19-psec rms jitter to calibrate an oscilloscope having 3.8-psec jitter.

The delay loop has some jitter, created by the conversion of amplitude noise to jitter. Without the loop, generator-amplitude noise causes the leading edge of each pulse to cross the oscilloscope's trip point either early or late. In this way, amplitude noise translates to jitter. The following formula gives jitter versus amplitude noise:

$$\text{JITTER} = \frac{\text{AMPLITUDE NOISE}}{\text{dV/dt}}.$$

An ideal loop would produce a delayed pulse that is identical to the first pulse. Amplitude noise on the delayed pulse would be identical to that on the first pulse; thus, jitter attributable to the noise would cancel out. Because of signal loss in a real loop, the delayed pulse is not identical to the first pulse. Therefore, the amplitude noise of the delayed pulse will be less that that on the first pulse, and a conversion of amplitude noise to jitter will occur. The following formula gives amplitude noise versus jitter in the loop:

$$\text{LOOP JITTER} = \frac{\text{AMPLITUDE NOISE}}{\text{dV/dt}} \times (\text{SIGNAL LOSS}).$$

The pulse generator used to calibrate the oscilloscopes exhibits 250 µV of rms noise and a dV/dt of 0.35V/nsec. The signal loss on the leading edge of the delayed pulse is 0.2V. The amplitude-noise-to-jitter conversion is thus:

$$\text{LOOP JITTER} = \frac{250\,\mu\text{V}}{0.35\text{V/nSEC}} \times (0.20) = 143\,\text{fSEC}.$$

The loop jitter of 145 fsec is so far below the jitter noise floor of the oscilloscopes under calibration that you can consider the delay loop as a zero-jitter source. Because digital-oscilloscope jitter is a function of the timebase setting and the amount of ADC dynamic range you use, you should calibrate the scope with a loop delay and amplitude that match the signal to the actual system or device under test.

Figure 2

The transmission-line delay loop creates a delayed signal with near-zero jitter for calibrating the oscilloscope.

REFERENCE

1. Adler, Joe, "Jitter in clock sources," Application Note, Vectron International.

Is this the best Design Idea in this issue? Vote at www.ednmag.com.

Excel offers painless LCD initialization

Alberto Bitti, Eptar, Lugo, Italy

To display a font or a symbol on an LCD, you need to convert the desired character into numerical data. Creating the data for an entire font set requires specialized tools; even with these tools, the task can be daunting. Alternatively, you can build a font calculator using an Excel spreadsheet. This technique takes advantage of the tabular nature of a spreadsheet to automatically create the required initialization code. The example in **Figure 1** applies to displays arranged in blocks comprised of eight pixels by eight pixels. You can easily adapt it to any other arrangement, such as the seven-by-five-pixel format used in the eight customizable characters found in most alphanumeric displays. Cells inside the yellow area represent the symbol "pixels." You draw the desired font character or symbol using a bold character such as "#." The formula in **Figure 2** is all that

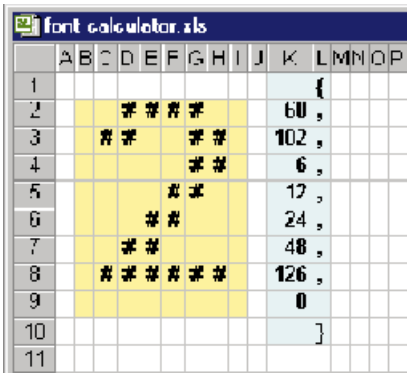


Figure 1 You edit the graphical shape (yellow). A simple formula builds the initialization code (blue), which is ready to paste in your application.

you need to obtain the initialization data. For each of the eight pixels in a row, the formula tests whether the pixel is blank (LEN is zero); otherwise, it adds the pixel's "weight" (1, 2, 4, 8, 16, 32, 64, or 128) to the result.

You can prepare the spreadsheet in minutes. Type the formula in the first position (K2), then copy it to all eight rows in a symbol (the blue area from K3 to K9). Complete it with the separators for your language of choice (commas and parentheses for C). To build an entire character set, copy the whole block as many times as necessary. To edit the fonts, place the cursor over the cells and type "#" (or any other character), and use "Canc" to delete. After editing, select all of the columns K and L; then cut and paste the code into your application. Besides saving you money, this technique is convenient and flexible. You can adapt it in minutes to any language (useful when you switch between assembly dialects). Moreover, the method accommodates useful additions, such as inserting #define KEY_ICON to name a particular data set, to suit your application's requirements.

Figure 2

```
= IF( LEN(B2);128; 0 ) +
  IF( LEN(C2); 64; 0 ) +
  IF( LEN(D2); 32; 0 ) +
  IF( LEN(E2); 16; 0 ) +
  IF( LEN(F2);  8; 0 ) +
  IF( LEN(G2);  4; 0 ) +
  IF( LEN(H2);  2; 0 ) +
  IF( LEN(I2);  1; 0 )
```

The formula (which, in this example, shows cell K2) adds pixel "weights" to obtain initialization values.

Is this the best Design Idea in this issue? Vote at www.ednmag.com.

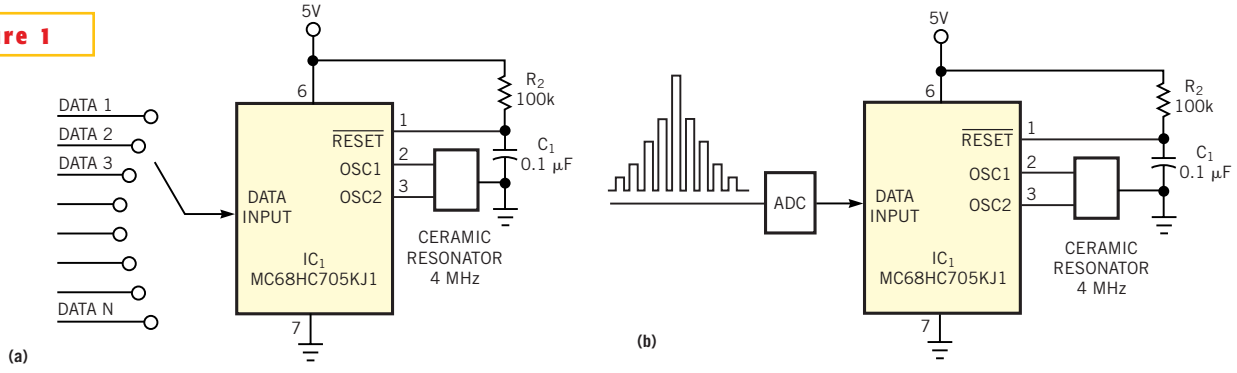
Microcontroller selects minimum/maximum value

Abel Raynus, Armatron International, Melrose, MA

Microcontroller-based systems for measurement, sensor-data processing, or control, sometimes require you to determine a maximum or minimum data value. For example, in an object-detection system, such as a radar or sonar system, the microcontroller receives echo signals from multiple targets and then must select the closest one; in other words, it must determine the minimum distance. Another example is an automatic-tuning system in which the microcontroller acquires data and must determine the maximum or minimum values of the data. A simple way to determine maximum and minimum value involves a microcontroller, IC_1, that receives a set of data from N sources (**Figure 1**). The data should be in 8-bit format. To simplify the process, consider that the data are 1-byte-long integers. In other words, the data cover the range 0 to 255. In most cases, this range produces satisfactory results. The same limitation applies to the number of data sources. If you need more data precision or more data sources, then you can use two or more bytes at the expense of added program complication.

Two approaches exist for the maximum/minimum values. In the first, the microcontroller memory collects the received data and processes the data to de-

Figure 1



In one approach to determining minimum and maximum values, the microcontroller stores data values in memory before processing them (a); in another approach, it processes data on the fly (b).

termine maximum or minimum values (**Figure 1a**). In the second approach, the microcontroller processes the data immediately after receiving it (**Figure 1b**). **Listing 1** shows the program for the first approach. Assume that a data array with N=8 exists in the microcontroller's memory. For demonstration purposes, **Listing 1** illustrates this part of the routine in lines six to eight and 17 to 21, in which the controller loads the data registers from a predetermined table. In a real situation, you would load the data registers from data sources before calling the program. The process of minimum-value detection is based on a method called "the bubble." The search algorithm starts testing the memory data array from its end. It clears the index register (X) after completion of the program. If you need to detect the maximum instead of the minimum value, you need change only one instruction in line 28.

In the second approach, the microcontroller waits for a data value entering the data register and, upon reception, starts processing the value (**Listing 2**). This approach needs no memory array. This variant of the program adds two features: First, it simultaneously selects minimum and maximum values of the incoming data and saves them in the DATAmin and DATAmax registers. Second, the routine considers a data value equal to 0 as no data. Thus, the data range is from 01_h to ff_h. For this project, you can use the inexpensive, one-time-programmable MC68HC705KJ1 from Motorola (www.motorola.com). Thus, the programs use Motorola's assembly language. However, the algorithms are so straightforward, they're amenable to any language and to practically any microcontroller. Go to the *EDN* Web site www.ednmag.com for an electronic version of the listings.

Is this the best Design Idea in this issue? Vote at www.ednmag.com.

LISTING 1—MEMORY-BASED MINIMUM/MAXIMUM DETERMINATION

```
                                1  ********* MIN VALUE SELECTION VAR 1 **************
                                2  *nolist
0000                            3  $include "std-j1a.asm"
                                4  *list
                                5  ****************************************
0000                            6  tabl equ ROMend-$07 ;for demo only !
07C8                            7    org tabl        ;table of Data
07C8        03040502            8    fcb 3,4,5,2,8,14T,9,25T
            080E0919
                                9  ****************************************
00C0                           10     org RAM
00C0                           11  DATA       rmb 8
00C8                           12  DATAmin    rmb 1
0300                           13     org ROM
0300 [05] 3FC8                 14  start clr DATAmin
0302 [03] 5F                   15     clrx
                               16  ****************************************
0303 [05] D607C8               17  m0    lda tabl,x ;put the set of data from
0306 [05] E7C0                 18    sta DATA,x ;table to registers. This
0308 [03] 5C                   19    incx       ;part of program only for
0309 [02] A308                 20    cpx #8     ;the idea demonstration.
030B [03] 26F6                 21    bne m0
                               22  ****************************************
030D [02] AE07                 23       ldx #7      ;N-1 into X-register
030F [04] E6C0                 24    lda DATA,x ;data8 into Acc.
0311 [04] B7C8                 25    sta DATAmin;store data8 in DATAmin
0313 [03] 5A                   26  mn0  decx        ;go to the next data,x
0314 [04] E1C0                 27    cmp DATA,x ;if DATAmin < DATA,x
0316 [03] 2504                 28    blo mn1    ;  then go to mn1
0318 [04] E6C0                 29    lda DATA,x ;otherwise replace DATAmin
031A [04] B7C8                 30    sta DATAmin;  with new data value
031C [03] 5D                   31  mn1  tstx ;if there are more data to be tested,
031D [03] 26F4                 32    bne mn0    ;  then repeat from mn0
031F [02] 8E                   33    stop       ;end of the program
                               34  ****************************************
07FE                           35    org VECTORS+6
07FE        0300               36    fdb start
```

LISTING 2—ON-THE-FLY MINIMUM/MAXIMUM DETERMINATION

```
                                1  ********* MIN/MAX VALUE SELECTION VAR 2
**************
                                2  *nolist
 0000                           3  $include "std-j1a.asm"
                                4  *list
                                5  ****************************************
 00C0                           6     org RAM
 00C0                           7  DATA      rmb 1
 00C1                           8  DATAmin   rmb 1
 00C2                           9  DATAmax   rmb 1
 0300                          10     org ROM
 0300 [02] A6FF                11  init lda #$ff
 0302 [04] B7C1                12    sta DATAmin
 0304 [05] 3FC2                13    clr DATAmax
 0306 [05] 3FC0                14    clr DATA
 0308 [03] 5F                  15    clrx
                               16  ****************************************
 0309 [03] B6C0                17  start  lda DATA ;wait for DATA / 0
 030B [03] 27FC                18    beq start
 030D [03] B1C1                19    cmp DATAmin ; Data > DATAmin?
 030F [03] 2402                20    bhs m0
 0311 [04] B7C1                21    sta DATAmin
 0313 [03] B1C2                22  m0  cmp DATAmax ; Data < DATAmax?
 0315 [03] 23F2                23    bls start
 0317 [04] B7C2                24    sta DATAmax
 0319 [03] 20EE                25    bra start
                               26  ****************************************
 07FE                          27    org VECTORS+6
 07FE        0300              28    fdb init
```

Circuit forms industrial-grade digital potentiometer

Phill Leyva, Maxim Integrated Products, Sunnyvale, CA

Both AC and DC motors in modern industrial systems often receive their control from PLCs (programmable-logic controllers) in a control room safely away from the process. If an operator must manually set the motor speed while observing the process, the component of choice is usually an industrial-grade potentiometer. The wiper of this potentiometer produces a signal of 0 to 10V that feeds back to a motor controller in the control room. Such potentiometers are expensive and prone to wear, however. Because of wear, they can open the control loop, allowing the motor to ramp up uncontrollably. With a few components, you can implement a reliable, low-cost, surface-mount digital potentiometer for industrial applications (**Figure 1**). The result is a direct drop-in replacement for the wear-prone mechanical potentiometer. The digital potentiometer occupies the same space as a mechanical unit within an enclosure. It takes power from the 10V that is formerly supplied to the mechanical potentiometer from the motor controller. The solid-state unit provides a similar output of 0 to 10V and delivers as much as 15 mA to the controller.

The key to the circuit is the low-power, digital-potentiometer 100-kΩ IC_3. Configured as a voltage divider, this IC provides an output of 32 discrete voltage steps between its minimum and maximum settings (0 and 5V). A low-power linear regulator, IC_1, provides a 5V supply rail for IC_2, IC_3, and a resistor ladder internal to IC_3. PB_1 and PB_2 constitute a double-pushbutton industrial switch. Each high-to-low transition that PB_1 produces increments the digital potentiometer's "wiper" by one step. Depressing PB_2 while toggling PB_1 decrements the wiper by one step. IC_2 is a switch debouncer that provides (in addition to the debouncing) a 40-msec fixed delay between its outputs and the switch action. To provide 0 to 10V outputs as required by the motor controller, a single-supply, rail-to-rail op amp, IC_4, amplifies IC_3's output by a factor of two. The input common-mode range for this op amp—250 mV beyond either supply rail—allows it to generate 0 to 10V outputs like a mechanical potentiometer. The circuit's low quiescent current ranges from 86 µA for a 0V output to 186 µA for a 10V output. To even further lower this quiescent current, you can choose the 200-kΩ version of IC_3. Because the op amp is stable with any capacitive load, it easily drives long, shielded, multiconductor cable lines back to the control room.

Is this the best Design Idea in this issue? Vote at www.ednmag.com.

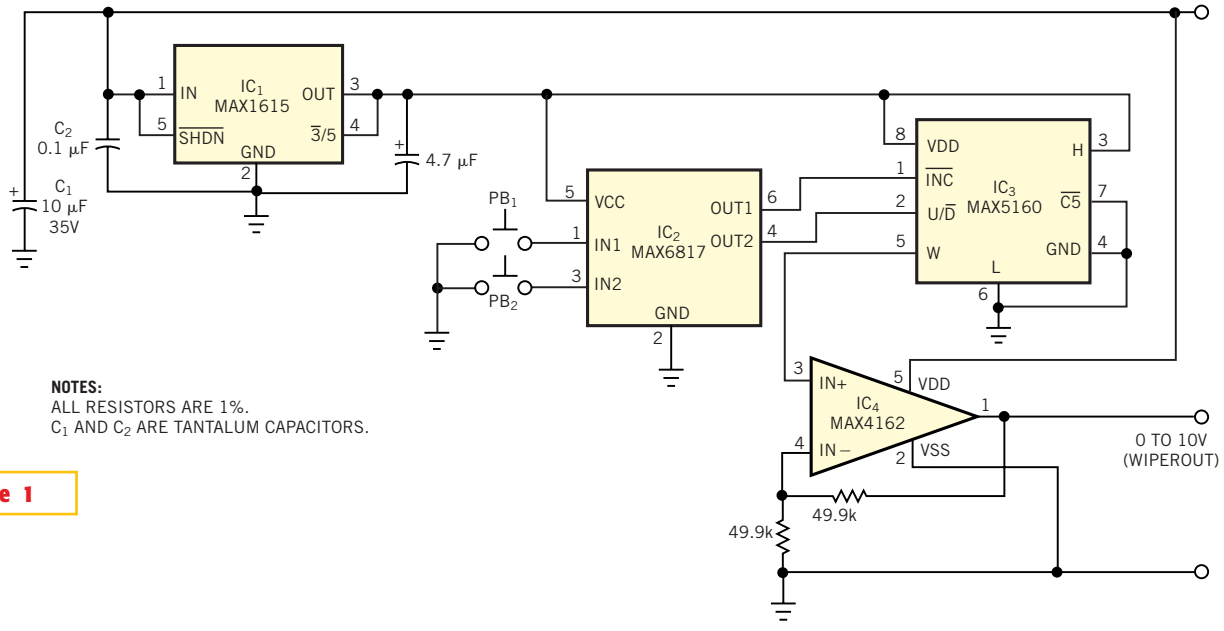


Figure 1

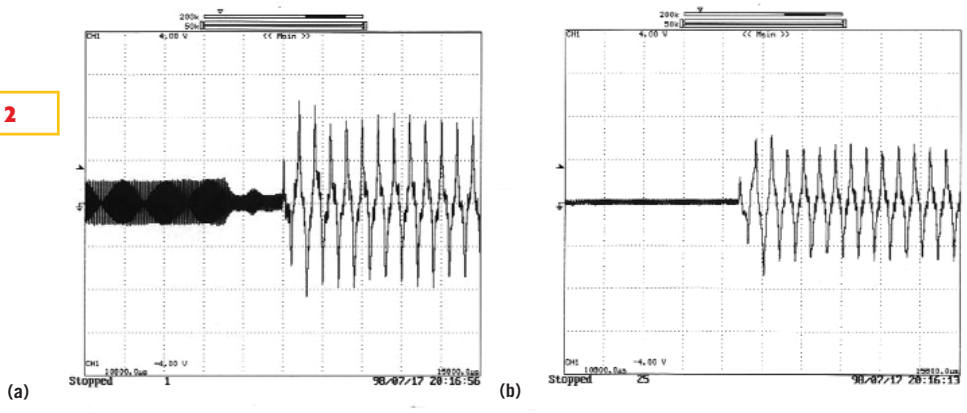
This solid-state potentiometer simulates a mechanical potentiometer and fits in the same space.

Passive filter cleans up power-line communications

Jose Sebastia and JJ Perez, Polytechnic University of Valencia, Spain

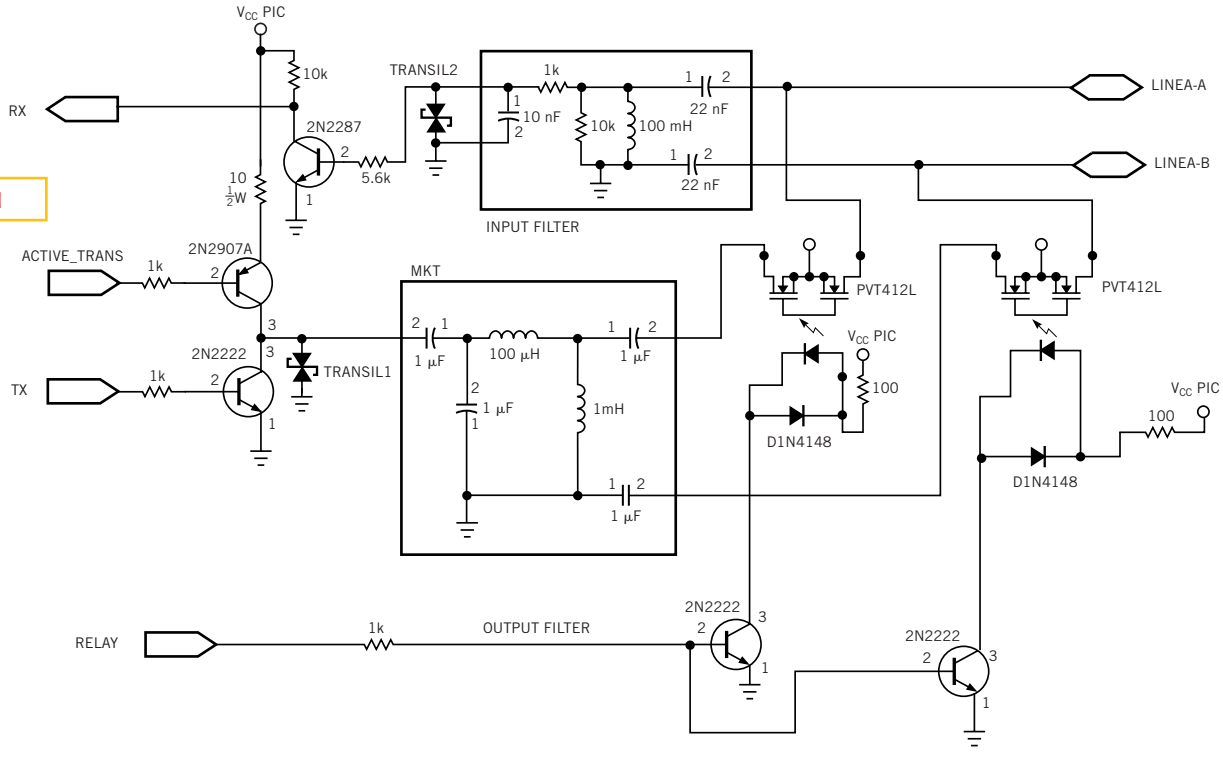
Many applications require the design of custom analog filters (**Reference 1**). The application for this design required simple and low-cost I/O filters for PLC (power-line communications), where low power consumption is a crucial factor. **Figure 1** shows the filters, which use passive components because of the requirement for low power consumption. The PLC system needs an in-

Figure 2



Without an input filter, a great deal of hash accompanies the input signal (a); the addition of the filter considerably cleans up the signal (b).

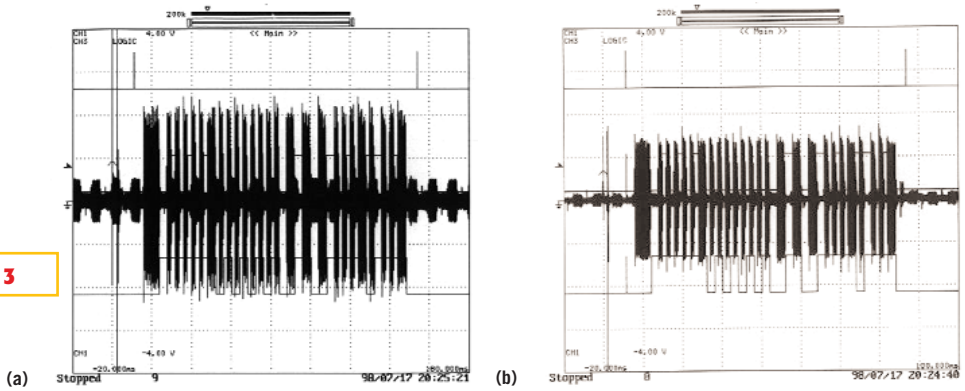
Figure 1



A power-line-communications system needs input and output filters to eliminate interference.

put and an output filter. They are 100-Hz to 20-kHz passband filters; the communication frequency is 5 kHz. The difference between the two filters lies in the input impedance. The input filter must present a 2.2-kΩ impedance, and the output filter must have a 30Ω impedance. The circuit also needs a solid-state relay, the PVT412 from International Rectifier (www.irf.com) to isolate the output filter. When the circuit is active, the relay connects the output filter to the line. A microcontroller controls the relay to implement the signal-transmission and -reception protocols. ACTIVE_TRANS, RELAY, and TX are the microcontroller pins that control transmission, and RX is the pin that controls reception. **Figure 2** shows waveforms before (**Figure 2a**) and after (**Figure 2b**) insertion of the input filter. **Figure 3** shows waveforms before (**Figure 3a**) and after (**Figure 3b**) insertion of the output filter.

Figure 3



Interference and noise are evident in the output signal (a); the addition of the output filter (b) markedly reduces the noise.

REFERENCE

1. Lacanette, Kerry, "A basic introduction to filters: active, passive, and switched-capacitor," Application Note AN-779, National Semiconductor, 1991.

Is this the best Design Idea in this issue? Vote at www.ednmag.com.