Build a Three-in-One Measurement System

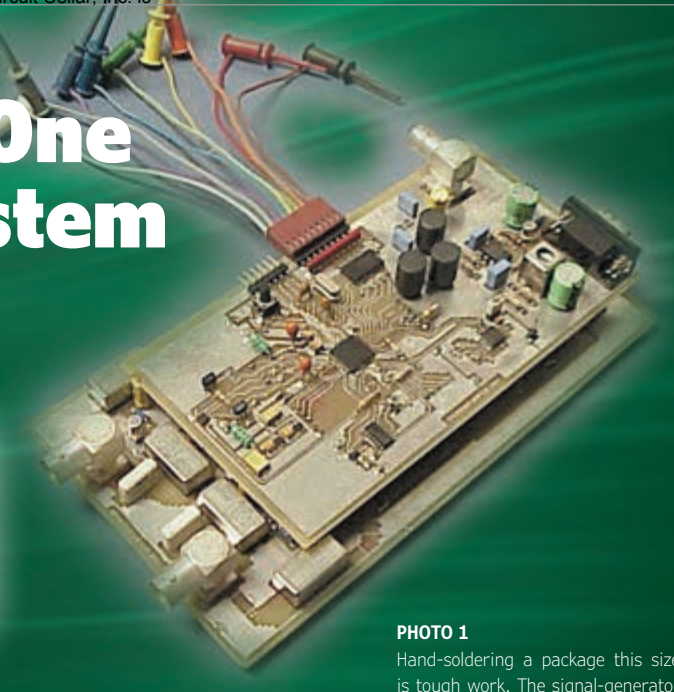


PHOTO 1
Hand-soldering a package this size is tough work. The signal-generator filter has bulky coils. In contrast, the MSP430F149's PQFP64 is tiny.

No home electronics lab is complete without a signal generator, logic analyzer, and digital oscilloscope. But why purchase the measurement devices separately, when you can build one system that houses all three? Salvador shows you how.

By Salvador Perdomo (Spain)

Editor's Note: This article first appeared in Circuit Cellar 156, 2003.

This article deals with some of the most important measurement instruments needed for a general-purpose electronic laboratory. It should prove to be a useful resource for electronic enthusiasts and engineers working in their homes, where signal generators, logic analyzers, and digital oscilloscopes are unavailable.

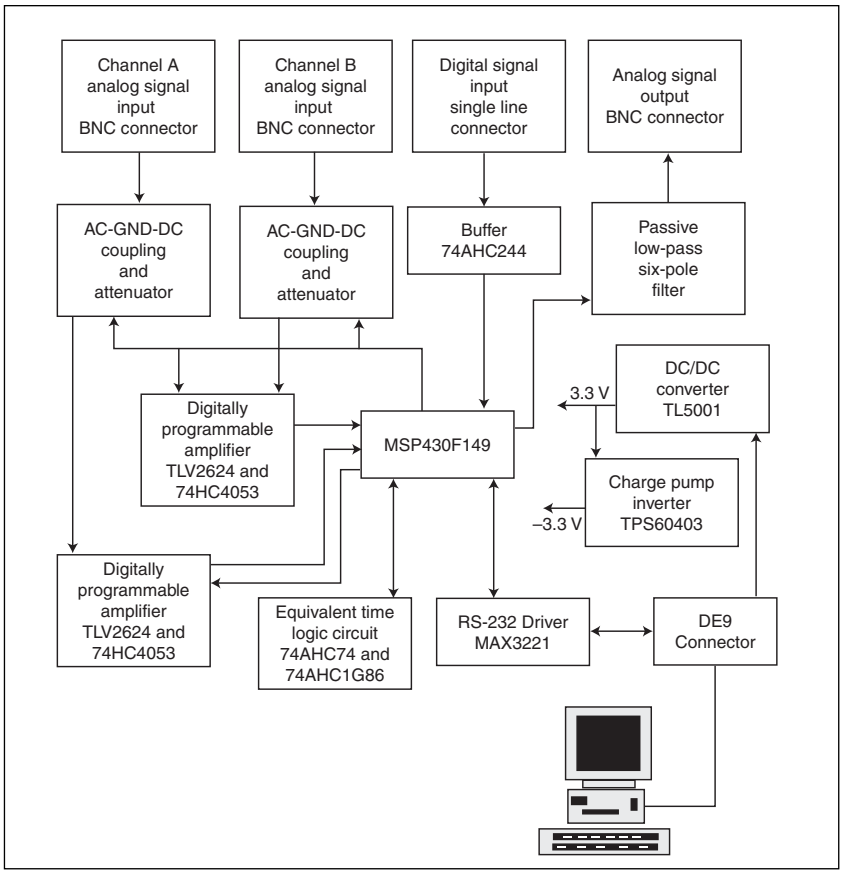


FIGURE 1
It is of interest to have your test benches as clear as possible to search for the faulty part of your design. So, a small measurement system is highly recommended. It's better if it isn't connected to the mains.

I've built an inexpensive and versatile measurement system that contains a signal generator, logical analyzer, and digital oscilloscope. If you build your own, you'll be able to address many of the problems typically encountered on test benches.

The system is not PC-bus connected. Instead, it's external to the computer, making use of the RS-232 serial port shown in **Figure 1**. Also, it doesn't have a power supply input, so the same serial cable feeds it. Because the computer's serial connection provides limited power, low power consumption is a fundamental requirement.

The low-power goal is achieved with a small number of components—the fewer the better. So, I quickly became interested in the MSP430F149, which is a highly integrated device with low power consumption. Note that everything is integrated except the oscilloscope analog chain (coupling and programmable amplifier), part of the trigger circuit, and the input buffer for the logic analyzer. The microcontroller works with an 8-MHz crystal oscillator.

This application uses the register bank, the entire RAM (2 KB), and nearly all of the peripherals. The peripherals used include the 16-bit TimerA and B, ADC, analog comparator, multiply accumulate, and one USART with modulation capability. Only the second USART is spared.

The system has several main features. You can control and display on the PC by running

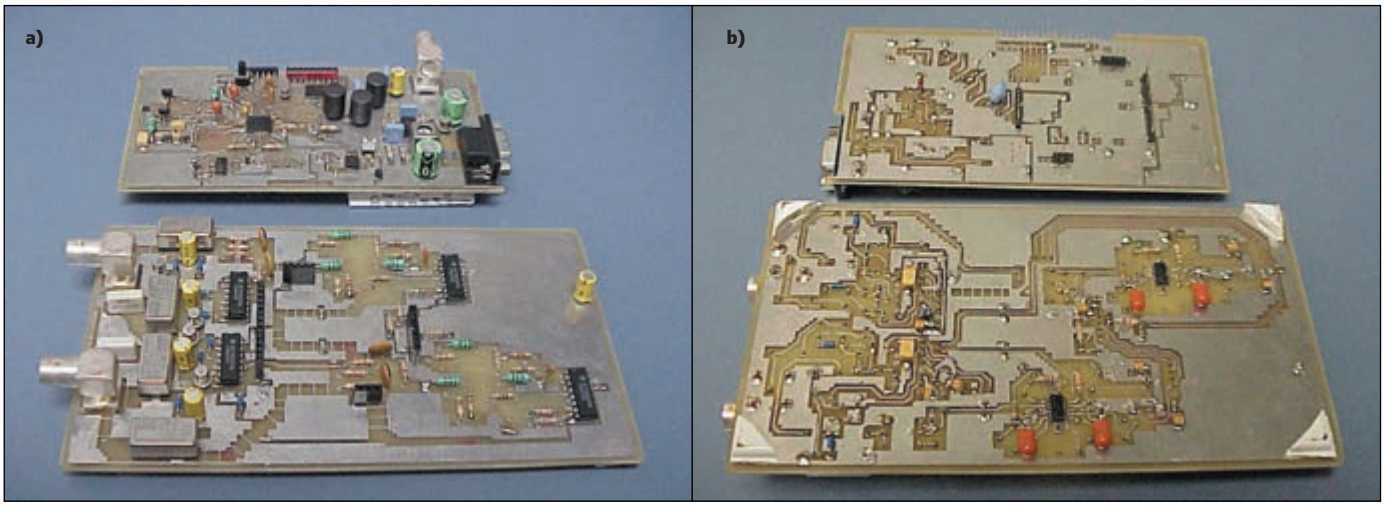


PHOTO 2
a—You can replace the relays in the coupling section and the driver circuit with solid-state relays if you can find ones with low leakage current. **b—**The op-amp's SMD packages are best viewed from the bottom. The larger board is populated on both sides. Note the importance of the parasitic coupling of the PWM D/A outputs to the input of the amplifiers.

software implemented on LabWindows/CVI. In addition, it has a signal generator based on the direct digital synthesis method and a frequency of up to 6 kHz with 0.3-Hz resolution. The output voltage reaches a peak of 1.3-V (±2 dB) fixed amplitude. The signal generator works simultaneously with the oscilloscope and logic analyzer (but not these two).

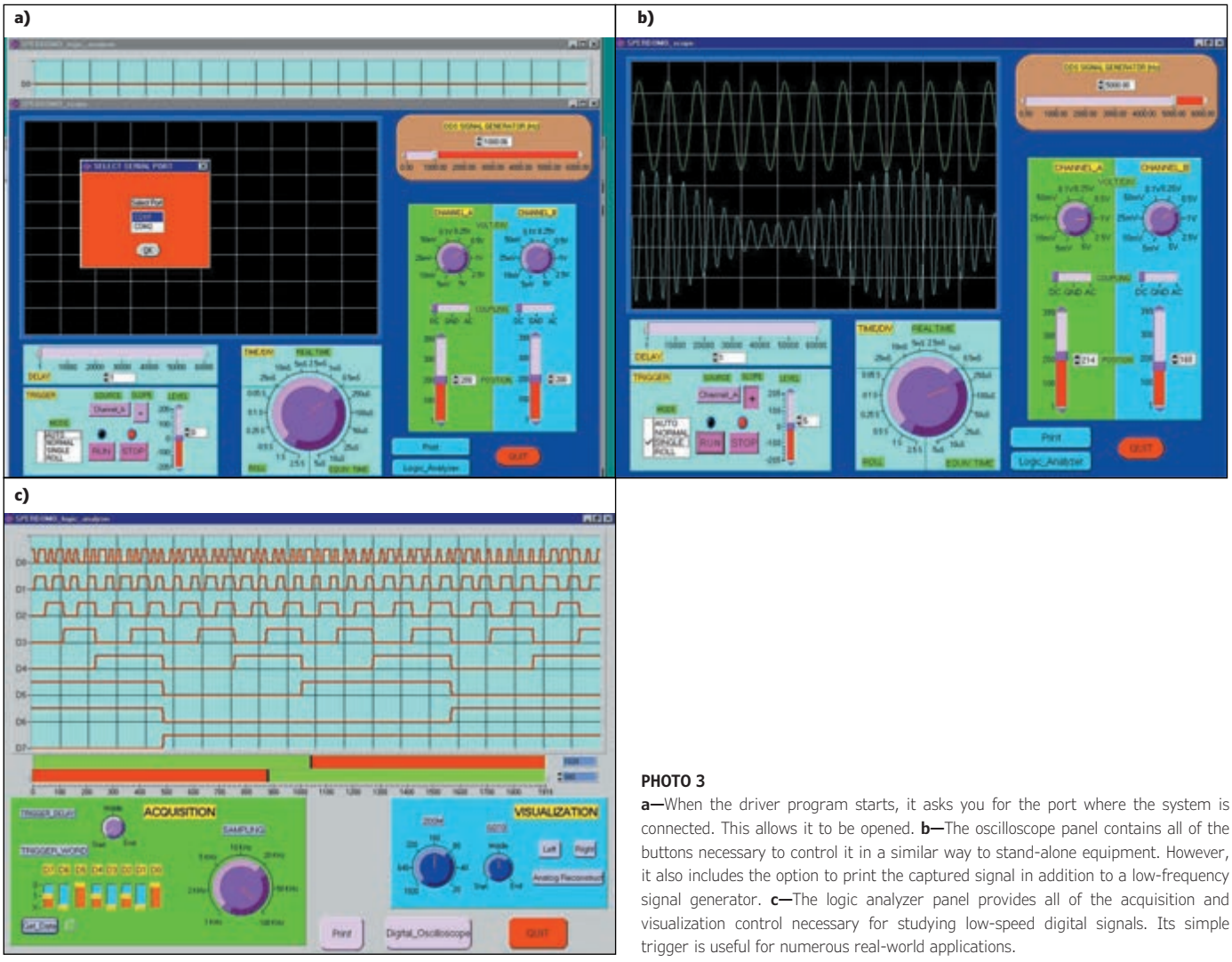


PHOTO 3
a—When the driver program starts, it asks you for the port where the system is connected. This allows it to be opened. **b—**The oscilloscope panel contains all of the buttons necessary to control it in a similar way to stand-alone equipment. However, it also includes the option to print the captured signal in addition to a low-frequency signal generator. **c—**The logic analyzer panel provides all of the acquisition and visualization control necessary for studying low-speed digital signals. Its simple trigger is useful for numerous real-world applications.

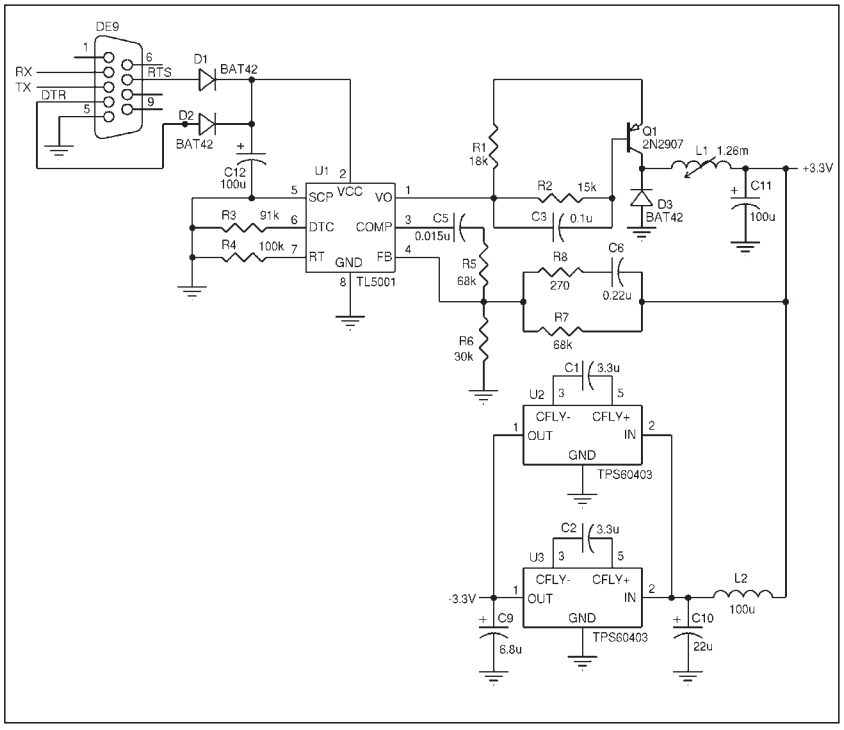


FIGURE 2
Thanks to the selected components' low power, I was able to eliminate the system's independent power. For this project, a DC/DC converter is indispensable.

I included a digital oscilloscope with two channels that have 1-MHz bandwidth, 8 bits of resolution, and 401 words of memory per channel. There are 10 amplitude scales from 5 mV to 5 V per division and 18 timescales from 5 µs to 2.5 s per division. Note that there are four working modes: Auto, Normal, Single, and Roll.

The logic analyzer has eight channels, 1920 words of memory per channel, and sampling from 1 to 100 kS/s. It is trigger-delay selectable between 0, 50, and 100% of memory length.

Looking at **Photo 1**, you see that the system's hardware consists of two separate boards that are attached to each other. Photo 2a shows the tops of the boards, and Photo 2b shows the bottoms. The larger board contains the oscilloscope analog chain: BNC connectors, relays (and circuit controller) for DC-GND-AC in the coupling section, and the digital programmable attenuator/amplifier. The top board contains the DC/DC converter power supply, charge-pump inverter, serial-communication driver, low-pass filter, trigger (real and equivalent time sampling) circuit, channel-trigger selector, and the microcontroller.

a)

$$SIN\_ROM = \text{round}\left\{199 \times \sin\left[\frac{2\pi(0:255)}{256}\right] + 200\right\}$$

b)

$$COSIN\_ROM = \text{round}\left\{199 \times 2\pi\cos\left[\frac{2\pi(0:255)}{256}\right]\right\}$$

c)

$$SINE = SIN\_ROM(Phase\_H) + COSIN\_ROM(Phase\_H) \times Phase\_L$$

FIGURE 3
a—The first equation extends the values from 1 to 399 to cover the dynamic range of the DAC. **b**—Now, the values extend from −1250 to 1250. **c**—Using the software DDS, you can create a sine function.

CONNECTION AND POWER

The RTS and DTR DE9 connector pins feed the system. (I didn't use the hardware handshake.) The Rx(2) and Tx(3) pins are used for communication between the microcontroller and PC. This is achieved with the USART0 and a MAX3221 driver.

First, I tested the RTS and DTR pins' I-V curves in order to know how much power was available. The curves are similar, and they resemble a PMOS device connected to a 12-V supply coming down to 10 V at 10 mA of current consumption and approaching 0 V at 14 mA. So, each pin produces a maximum power of approximately 100 mW (200 mW for both of them).

In order to feed the system with 3.3 V from the serial port, I used a 3.3-V output DC/DC converter (see Figure 2). The controller is based on a TL5001. (In the photos, it's near a radio-IF filter can that contains the converter's coil.) Also, a couple of TPS60403 charge-pump voltage inverters were used in parallel to feed the analog portion of the system with ±3.3 V.

Although I could have bought a 1.2-mH unit for the coil construction, I decided to build it with an old 10.7-MHz radio IF filter core. The coil construction took 180 turns of 4-mils (0.1 mm) diameter wire and gave an adjusted range value from 0.5 to 1.26 mH. The coil's DC resistance is 4.1 W.

The circuit was measured, and it performed well up to a current consumption of 42 mA (a power of 138 mW), which is enough to feed the entire system. The RTS and DTR pins remain at −12 V when the serial port is closed. When you call the PC driver program, the first panel appears (see Photo 3a). When you select the serial port where the system is connected, the port opens (increasing the RTS and DTR voltage to 12 V), configures a protocol with 115,200 bps, 8N1, and reserves a receiving buffer of 2000 bytes. Also, an interrupt for receiving_buffer_full (802 bytes) is prepared to let the main program know that a datastream has arrived. The LabWindows statements include the following two lines of code:

```
OpenComConfig (COMx, " ",
115200, 0, 8, 1, 2000, 30)
```

```
add     FREQ,&PHASE          //Phase accumulate from 0 to 65535
mov.b   &(PHASE+1),TABLE_S   //PHASE_H is used to read tables.
rla     TABLE_S                X2 accesses word tables.
mov     #8000h,&RESLO        //To round the MACS to the 16-bit
                                nearest integer.
mov     SIN(TABLE_S),&RESHI
mov     COSIN(TABLE_S),&MACS
mov.b   &PHASE,TABLE_S       //OP2 is a 16-bit SFR, so go throw
                                a register to translate PHASE_L
                                to OP2.
mov     TABLE_S,&OP2
mov     &RESHI,&TBCCR2       //Got new sample. Update TB2_PWM.

SIN  DW  200,205,210,215,220,224,229,234,239,244,248,253
     DW  258,262,267,272,276,281,285,289,294,298,302,306, ...

COSIN DW  1250, 1250, 1249, 1247, 1244, 1241, 1237, 1232, 1226, 1220
     DW  1213, 1205, 1197, 1187, 1177, 1167, ....
```

LISTING 1
Now that you're familiar with the software DDS, you can generate the sine function in Figure 3c.

```
InstallComCallback(COMx, 15,
802, 0, RxBuffer_full, 0)
```

When the microcontroller receives power, it starts from reset and configures all of the necessary peripherals: the comparator, ADC, TimerA and B, USART0, I/O pins, and the interrupts. Afterwards, it stays in a default state, waits for a PC command, and attends to the TimerB signal generator interrupts because it starts generating a 1-kHz sine wave by default.

THE SIGNAL GENERATOR

The signal generator is intended to provide the signal A × sine(wt), where the amplitude, A, is fixed near VCC/2, and the frequency is programmable with 0.3-Hz (20 kHz/65,536) resolution and up to 6 kHz.

The signal generator's hardware consists of a D/A converter made with TimerB PWM output TB2 (pin 4.2), an external six-pole RLC low-pass passive filter, and a BNC connector. To save more components, the necessary D/A converters are carried out with the PWM of TimerB. A passive filter loads the output transistors and produces distortion in the signal. Active filters are recommended instead. You may download a schematic of the signal generator's hardware from the Circuit Cellar ftp site.

The PWM-D/A converter has a 20-kHz

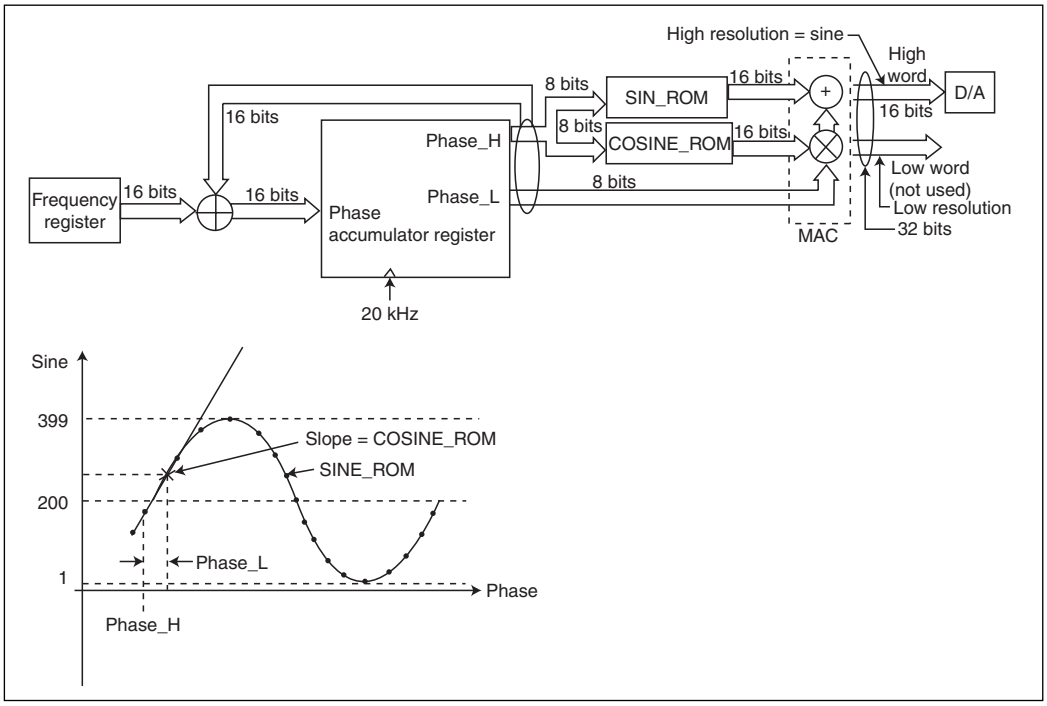


FIGURE 4
The DDS technique for synthesis is a recent development. Today, the IC-form approach has a 1-GHz sampling frequency with phase accumulators of 32 bits or more. I found the performance to be a modest 20 kHz with a 16-bit phase accumulator.

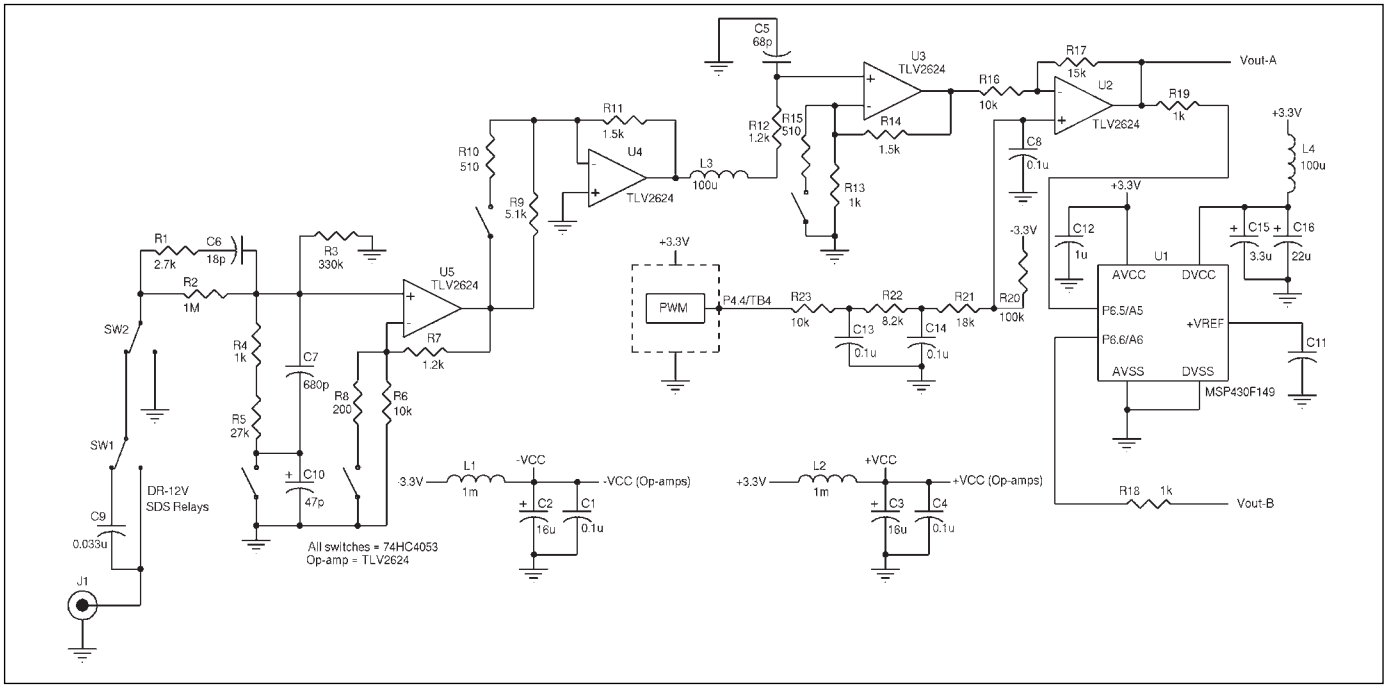


FIGURE 5
The analog conditioning chain must have digital programmable gain capability in order to adjust the voltage range of the input signal to the ADC input voltage range.

sampling frequency. This value is the result of several trade-offs among resolution, time spent attending to its interrupt and peripherals, and the maximum generated frequency. The converter has the following resolution:

$$\frac{8\ \text{MHz}}{20\ \text{kHz}} = 400$$

which is equivalent to 8.64 bits.

As you can see in Photo 1, the filter's inductors are power chokes. Their series resistance was measured, and it changes with the frequency in the following way:

$$R_S = 27 + 0.9e - 3f + 0.5e - 6f^2$$

This behavior results from the losses in the ferromagnetic core. It was taken into account in the filter design as well as the resistance of the PWM (pin 4.2) output MOS transistors. The filter was designed after measuring the coils with a MATLAB-based program that took into account the coil-loss variations with the frequency. You may download graphs of the frequency response from the Circuit Cellar ftp site. The graphs represent the response to the sampling frequency and the passband details.

The distortion of the PWM as a D/A converter is by far the biggest source of spurious signals, mainly because the output MOS transistors must supply the analog current to the passive filter. So, I recommend using an active filter because it won't load the PWM, and it will save you from using bulky coils.

The rest of the signal generator is based on a software digital synthesizer (DDS) composed of a 16-bit frequency register, 16-bit phase accumulator register, and two look-up tables (SIN_ROM and COSIN_ROM). Each of these tables is 256 words long and 16 bits wide. The values are computed in MATLAB using the equation shown in Figure 3a. By using the equation in Figure 3b, the values extend from –1250 to 1250. This equation represents the time derivative (slope) of the SIN_ROM table multiplied by 256. Note that 0:255 (i.e., 0 through 255) is simply a way to create an array of numbers in MATLAB.

Figure 4 depicts a function diagram of the software DDS. It uses the microcontroller's multiply-accumulate capability to generate the sine function in the equation shown in Figure 3c, which is easily carried out with the code in Listing 1.

THE DIGITAL OSCILLOSCOPE

The oscilloscope panel incorporating the signal generator's controls is shown in Photo 3b. The brown box in the upper-right corner is the DDS control, which controls the generated frequency. Everything else pertains to the visualization and control of the digital oscilloscope. As you can see, the panel is visualizing two signals: a 5-kHz sine DDS generated in channel_A (green trace), and a 10-kHz AM modulated signal generated by a commercial generator in channel_B (blue trace). The digital oscilloscope's hardware consists of a configurable analog chain that drives the ADC, RAM, a trigger circuit, and a display.

Each of the digital oscilloscope's channels has a configurable coupling stage (DC, GND, AC) made with two low-power (high-resistance, 1400-W) DR-12V monostable relays from SDS-Relais—a company that's now called Matsushita Electric Works UK. Its pick-up

voltage is 9.6 V, and its dropout voltage is 1.2 V. So, the circuit that drives its coil is a little tricky when you're trying to engage it with only ±3.3 V. To produce a transient response bigger than the available power supply, you must rely on reactive components (i.e., coils, capacitors, or both). In this system, the charge stored in a capacitor is used as a floating battery that's added to the fixed power supply. You may download a diagram of the circuit from the Circuit Cellar ftp site.

When the microcontroller pin changes from a high level to a low level, a pulse that's long enough and close enough to 12 V is applied to the coil to pick it up. Afterwards, it continues applying approximately 3.3 V, which keeps it engaged (neglecting the voltage drop in the Schottky diode and the transistor saturation voltage).

Nearby, there is a digitally controlled attenuator and amplifier around the low-power, high-bandwidth CMOS op-amp (TLV2624) and 74HC4053 multiplexer. Of the 16 possible switch combinations, only 10 are used to obtain 10 different gains (from 30 to 0.03) corresponding to 10 different input ranges (i.e., oscilloscope sensitivity from 5 mV to 5 V per division).

The bandwidth achieved is always better than 1 MHz. The signal path that runs from the BNC connector to the ADC input for one channel is shown in Figure 5. As I calculated its values, I took into account the pin's capacitance, the op-amp frequency response, and the 74HC4053 switch's on resistance (approximately 70 W). The ADC had 12 bits of resolution, but I used only the eight higher bits that were sent to the PC. The internal 1.5-V reference voltage fixes the input range.

In order to control the position of the channel A and B traces in the screen and the offset of the amplifiers, two PWM DACs and a passive low-pass filter—which are based on TimerB PWM outputs TB4 (pin P4.4) and TB3 (P4.3)—are provided (see Figure 5). The amplifiers' outputs go to the micro's ADC inputs—A5 (pin 6.5) and A6 (P6.6)—and to a trigger-channel selector made with a 74LVC2G66, which feeds the MSP430F149 analog comparator connected with positive feedback (Smith trigger). This is achieved in such a way that it has 30-mV hysteresis, or 2% of 1.5 V (the full range of the ADC).

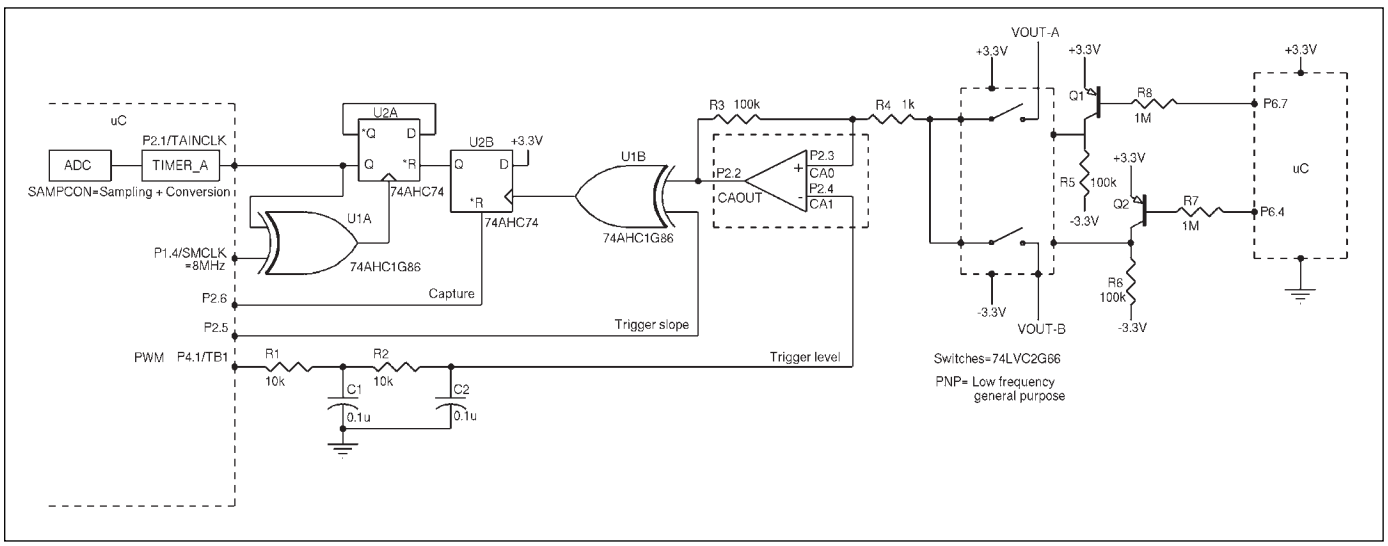
The comparator also receives the output of another PWM DAC and passive low-pass filter based on the TimerB PWM output TB1 (pin P4.1) that establishes the oscilloscope trigger level. After the comparator, the 74AHC1G86 exclusive OR gate is used to select the trigger slope.

Figure 6 depicts this portion of the hardware with the rest of the trigger circuit, which makes possible the sequential equivalent time-sampling technique. It's composed by one 74AHC74 (a couple of D flip-flops) and the 74AHC1G86 exclusive OR gate.

The trigger circuit and TimerA collaborate in order to make the oscilloscope work with this technique. It happens as soon as you select the time bases from 5 to 250 µs per division, and it is transparent. In this way, the oscilloscope bandwidth is only limited by the analog bandwidth and comparator precision, and not by the ADC maximum frequency conversion (Nyquist criteria).

The only requirement to function with this technique is that the input signal must be periodic during the acquisition. For instance, given the faster time base of 5 µs per division, it is necessary to capture 401 samples (400 intervals of 125 ns) to complete 50 µs of the signal (i.e., 5 µs per division × 10 divisions). But the MSP430F149 ADC's maximum sampling frequency is limited to 200 kS/s because of the

FIGURE 6
The trigger is probably the key section in an analog or digital oscilloscope. To start the capture, it must provide a clean and precise point in the signal.



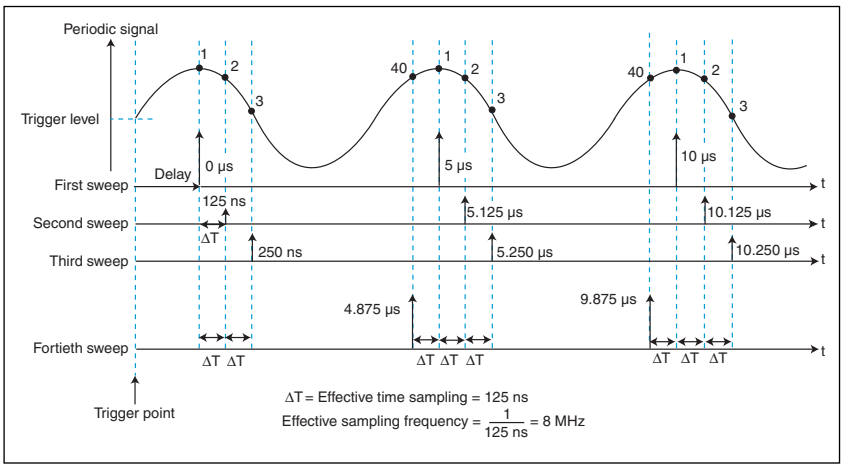


FIGURE 7
If the ADC has a limited conversion speed and its analog bandwidth is higher than the Nyquist criteria enforces, some kind of equivalent time sampling can be applied. This figure explains one of the techniques—sequential time sampling.

5-µs conversion time (per channel).

The signal is periodic, so it is possible to make successive trigger-capture cycles or sweeps capturing only a portion of the signal on each sweep. They would have to be incrementally delayed with respect to the trigger point, as illustrated in Figure 7. TimerA is in charge of the delay. For the 5-µs-per-square time base, the ADC is programmed to acquire 11 successive samples (5 µs apart) on each sweep. Forty successive sweeps, which are incrementally delayed 125 ns, are performed to total 440 samples. Note that only the first 401 are sent to the PC.

Real time covers the time bases from 500 µs to 2.5 s per division, and it implies only one sweep capturing 401 samples per channel in both channels simultaneously (Nyquist criteria applies). In practice, there is a delay of one sample between the two channels because there is only one sample-hold (actually the channels are converted interlaced), but it isn't noticeable.

Equivalent time and real time (depending on the time base that's selected) are the ways the hardware works when you select Auto mode, Normal mode, or Single mode from the PC's oscilloscope mode control. Now, let's take a look at each one.

In Auto mode, an automatic trigger will occur if there is not a trigger within a fixed 0.2-s interval. This fixed time is commanded by the PC if it does not receive the samples it is waiting for from the previous Acquire command. To produce the automatic trigger, the microcontroller changes pin P2.5 (trigger slope) twice in order to assure that the first flip-flop in Figure 6 is set. When the 802 samples arrive (401 + 401), another Acquire command is released.

In Normal mode, a trigger event is necessary to acquire data. Only after all of the 802 samples have arrived does the PC release another Acquire command.

Single mode is similar to Normal mode, but there is one major difference. Basically, after all of the samples have arrived, the PC stops waiting for another user command.

Roll mode is only selected from 0.05 to 2.5 s per division. It is different from the other modes because it doesn't use a trigger event to start acquisition. Instead, it is continuous, and the microcontroller doesn't wait to acquire 401/channel samples before they are sent to the PC.

In Roll mode, a smaller number of samples (depending on the time base) are acquired and sent. For instance, at 2.5 s per division, only four samples per channel are acquired before they are sent to the PC. When the PC receives them, the old samples are shifted to make room for the new ones and are shown on the screen. This produces an effect of picture displacement known as the roll effect. Of course, because the datastream length changes with respect to the other modes, the LabWindows receiving_buffer_full interrupt has to be adapted correspondingly (function InstallComCallback).

THE LOGIC ANALYZER

The logic analyzer panel is shown in Photo 3c. There is no direct access from this panel to the signal generator, but it keeps generating a signal with the frequency previously fixed in the oscilloscope panel. Photo 3c shows only the central 160 samples per channel (zoom applies) of the 1920 samples per channel captured.

The hardware is easily built with a 74AHC244 buffer and a pull-down array of eight 1-MW resistances. The 74AHC244 buffer makes the

circuitcellar.com/ccmaterials

PROJECT FILES

To download the code and additional files, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2003/156.

RESOURCES

Texas Instruments, "MSP430 Bug list," www.ti.com/sc/cgi-bin/buglist.cgi.

———"MSP430x1xx Family User's Guide," SLAU049A, 2001.

SOURCES

MATLAB
MathWorks, Inc. | www.mathworks.com

Monostable relays
Matsushita Electric Works UK | www.matsushita.co.uk

LabWindows/CVI
National Instruments Corp. | www.ni.com

74AHC244 Buffer
Philips Semiconductors | www.semiconductors.philips.com

MSP430F149 Microcontroller
Texas Instruments, Inc. | www.ti.com

system 5-V, TTL-compatible, and is connected to port 5 on the microcontroller. The rest of the logic analyzer (i.e., the sampling frequency, triggering, and trigger delay) is software-based. Also note that it's 8 bits wide with a 1920-KB memory depth and an acquisition frequency range from 1 to 100 kS/s.

The trigger delay is user-selectable, which enables pre-triggering, middle triggering, and post-triggering. Because of the asynchronous sampling of the data, the visualization is only available as a timing diagram.

Concerning the triggering, when you activate the Get_Data control, the PC's main program extracts two bytes—ID (ID7 through ID0) and IDE (IDE7 through IDE0)—from the states of the trinary switches, D7 through D0. D7 through D0 define the trigger word with three possible values (0, 1, and x) bit by bit. After the microcontroller receives the IDE and ID bytes from the PC, the sampled input data is masked (masked_DATA = DATA logical AND with IDE), making zero the don't care bits (option x) selected by the user. This masked input data is compared to the ID byte (masked_DATA is CMP with ID). A match validates the trigger. If there is a match, the trigger delay is counted and the acquisition is completed. Afterwards, the data is sent to the PC. You may download a diagram of this process from the Circuit Cellar ftp site. Note that the process is used for the trigger word depicted in Photo 3c.

The simplest event that can trigger a logic analyzer is the coincidence of a data with a word you have selected. This coincidence must be bit by bit. To define the trigger word, some switches are provided so you can set each bit to 0, 1, and x (don't care).

THE PC-MICRO COMBO

By default, the microcontroller attends to the TimerB signal-generator interrupt every 50 µs (20-kHz sampling frequency), and the subroutine lasts 7.875 µs including the latency time. So, the remainder of the time is available for the received PC commands or the interrupts and commands released by other peripherals.

The PC commands are composed of 3 bytes: two data bytes and one command byte. When the USART0 received data interrupt is attended, the number of data bytes received are counted in order to correctly deposit them in three registers, including word_received (16 bits) and PC_command (8 bits). Back in the main program, the PC_command register is used to make a table-based indexed branch to the routine that serves the intended command.

The reason for accompanying the byte command with two data bytes is self-explanatory. To change the frequency of the signal generator, it is necessary to load a new 16-bit value in the frequency register in Figure 4, and to change the trigger level or change the channel trace position. Another example is that the acquisition command, START_ADQ, is accompanied by a number indicating the time division to program TimerA in order to fix the ADC acquisition frequency. Another number indicates how the samples have to be dealt with (e.g., equivalent time, real time, or roll). For other commands (e.g., TRIG_SLOPE and TRIG_SOURCE), they're unnecessary and filled with dummy data.

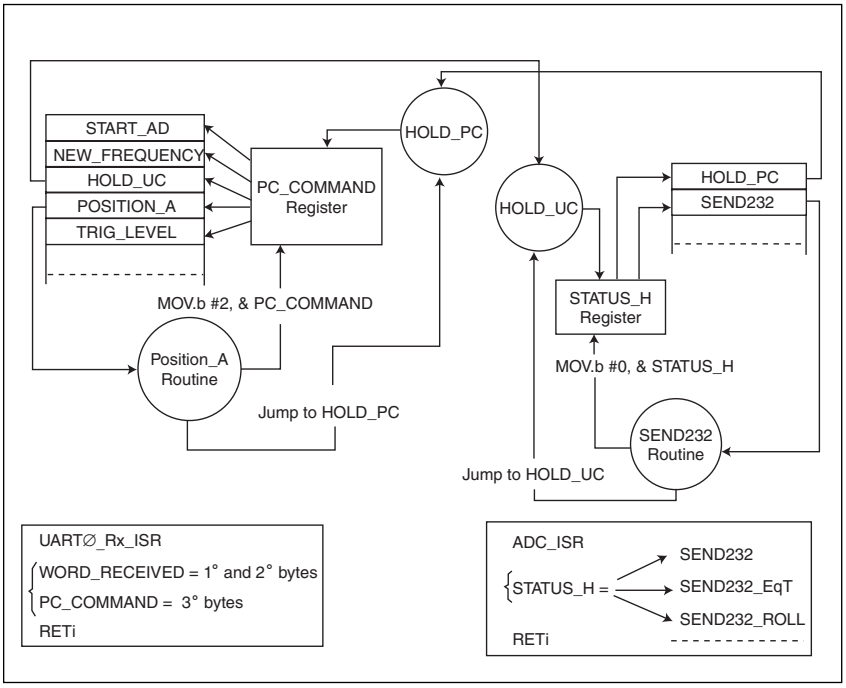


FIGURE 8
Two different registers control the microcontroller's program flow. The PC writes one (PC_COMMAND), and the other (STATUS_H) is written by the peripheral when their interrupts are attended.

Otherwise, there is another microcontroller register called STATUS_H, which keeps track of the peripherals' jobs. For instance, when the ADC routine has loaded 401 samples per channel in the RAM memory, it deposits a peripheral command such as SEND232 in this register. Thus, when back in the main program, the STATUS_H register is used to make a table-based indexed branch to the routine that serves the intended command. (In this example, the acquired data is sent to the computer.)

A helpful flow chart is depicted in Figure 8. You may download several lines of code representing these ideas from the Circuit Cellar ftp site.

Keep the following advice in mind when you're changing from the oscilloscope to the logic analyzer (and vice versa): You can send a command to make the microcontroller return to a default state (waiting for a PC command and attending the TimerB signal-generator interrupts), and stop sending data if it is doing it. Simultaneously, the PC data queue must be cleared.

ABOUT THE AUTHOR

Salvador Perdomo received a degree in Telecommunications Engineering from the Universidad Politécnica de Madrid, Spain. He has lectured at the Universidad de Las Palmas de Gran Canaria, Spain. His interests include analog and digital electronics. You may reach him at sperdomo@det.ulpgc.es.