

designideas

READERS SOLVE DESIGN PROBLEMS

Circuit and software provide accurate recalibration for baseline PIC microcontroller's internal oscillator

Noureddine Benabadji,
University of Sciences and Technology, Oran, Algeria

All of Microchip's (www.microchip.com) baseline PIC microcontrollers have internal 4-MHz oscillators, which are useful for freeing up one or two pins for I/O use and allowing you to build minimal-component-count designs using these devices. You must calibrate the internal oscillator by reading a factory-programmed calibration setting that resides at the last address in the user-program memory and then writing that setting into the microcontroller's oscillation-calibration

register during the application software's initialization of the device. Because the calibration value is unique to each microcontroller, problems arise for time-sensitive applications if you erase or overwrite the last address.

The circuit in **Figure 1** recovers the calibration value by recalibrating against a reference clock, the 4-MHz crystal. The frequency looks for the best calibration value to ensure that the microcontroller's internal oscillator runs within 1% accuracy at 4

DI's Inside

54 Microcontroller moving-dot display interface uses three I/O lines

56 Microcontroller displays multiple chart or oscilloscope-timing ticks

56 Fast-settling synchronous-PWM-DAC filter has almost no ripple

58 Switched-gain op amp serves as phase detector or mixer

► **What are your design problems and solutions? Publish them here and receive \$150! Send your Design Ideas to edndesignideas@reedbusiness.com.**

MHz. You can download the microcontroller's program and a flow chart

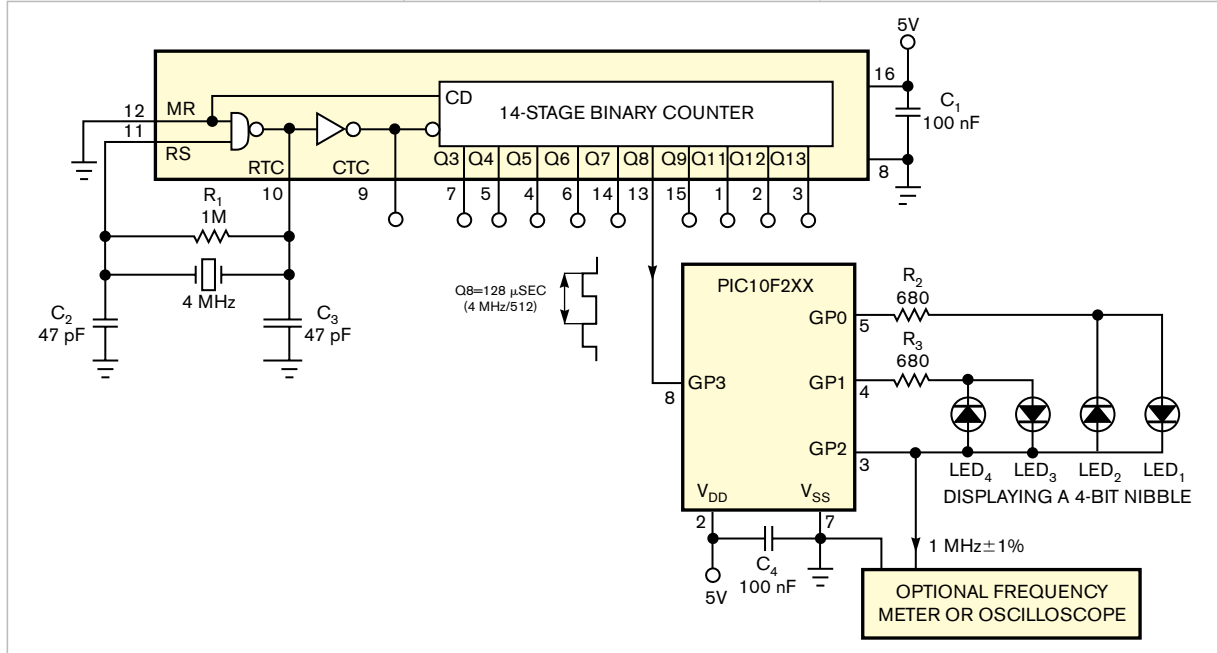


Figure 1 This circuit and an assembly-language program that occupies less than 250 bytes allow you to calibrate a PIC10F2xx microcontroller against a 4-MHz reference clock.

in a compressed zip file from www.edn.com/080501di1.

The baseline PIC microcontroller, which includes the PIC10F, PIC-12C508/509/510, or PIC16F505/506 series, uses its internal timer, Timer 0, to count the number of instruction cycles that execute in one period from


output Q8 of a Fairchild Semiconductor (www.fairchildsemi.com) CD4060 oscillator/divider to the only input, GP3, of the PIC microcontroller. The 4-MHz crystal drives the CD4060, which yields a period of 128 μ sec from the output Q8.

The four LEDs display the two 4-

bits nibbles of the 8-bit oscillation-calibration register's best value. Output GP2 acts as a multiplexing line to drive these LEDs for 8 to 10 sec and then as the oscillator output to yield a 1-MHz signal, which you can measure with a frequency meter or an oscilloscope. **EDN**

Microcontroller moving-dot display interface uses three I/O lines

Abel Raynus, Armatron International, Malden, MA

 The moving-dot display has some benefits over the bar-graph display: It better indicates the location of a detected object in sonar and radar applications; it needs only one LED's current-limiting resistor instead of several; and it provides the same current for all LEDs, thus providing even brightness. When a new design required adding a seven-LED moving-dot display to an 8-bit, low-end microcontroller, a question arose about the corresponding interface. Of course, the most cost-effective approach is to di-

rectly connect the LEDs without any extra parts. But this approach needs seven vacant microcontroller-output pins, which microcontrollers with limited I/Os often cannot afford.

A previous Design Idea describes a one-wire interface that applies only to a bar-graph display, not to a dot display (**Reference 1**). Another tack would be interfacing using serial-to-parallel shift registers or a serial-input Johnson counter. But small microcontrollers often lack a SPI (serial-peripheral interface), and you must use firmware to

re-create it (**Reference 2**). The method in this Design Idea needs three output lines—data, clock, and latch—and requires some firmware and hardware. Exploiting the fact that only one LED in a dot display should light at a time, you can use National Semiconductor's (www.national.com) CD4051 1-to-8 analog demultiplexer (**Figure 1**). This circuit needs three microcontroller outputs, and the firmware is simple and straightforward. The additional benefit is that the microcontroller now does not limit the LED current and voltage; you can choose them independently.

Listing 1, which you can download from the Web version of this Design Idea at www.edn.com/080501di2, provides demo firmware illustrating this design. The demo program automatically moves the lit dot back and forth by incrementing and decrementing a modulo-7 counter. Ideally, any three adjacent microcontroller outputs, such as pA0, pA1, and pA2, are available for the A, B, and C inputs of the CD4051. But, this scenario is not always possible. In this application of a low-end, eight-pin MC68HC908QT1 microcontroller, you can use pins pA2 and pA3 only as inputs. You can easily overcome this problem by programming, as **Listing 1** shows. This Design Idea applies to any small microcontroller because it uses only a standard instruction set. **EDN**

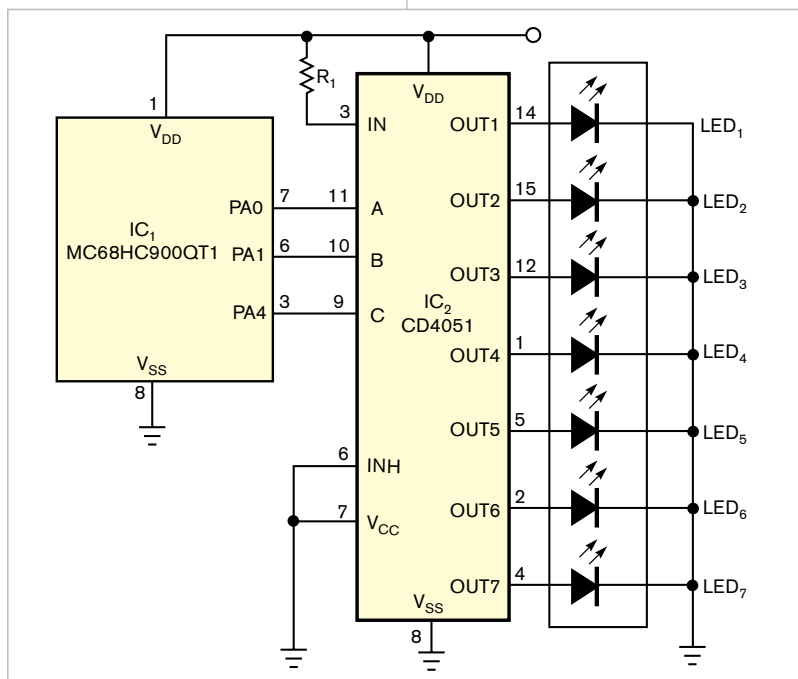


Figure 1 Using the 1-to-8 CD4051 analog demultiplexer you can interface a moving-dot LED display to a low-end microcontroller using just three outputs.

REFERENCES

- 1 Jayapal, R, "Microcontroller's single I/O-port line drives a bar-graph display, *EDN*, July 6, 2006, pg 90, www.edn.com/article/CA6347254.
- 2 Raynus, Abel, "Squeeze extra outputs from a pin-limited microcontroller, *EDN*, Aug 4, 2005, pg 96, www.edn.com/article/CA629311.

Microcontroller displays multiple chart or oscilloscope timing ticks

William Grill, Honeywell, Lenexa, KS

While working with a 10-bit DI-184 module from Dataq to monitor and display vibration-sensor data, I found that, although the chart displays a time index, this time reference is not visible in the saved file. You can add time ticks, representing seconds or minutes, to a chart graphic

by using a simple and inexpensive crystal-based microcontroller to generate a sequence of tags on a dedicated chart channel. **Figure 1** shows a small, eight-pin 12F508 microcontroller from Microchip Technology (www.microchip.com) that provides multiple timing ticks. **Listing 1**, the microcontroller's

program is available in the Web version of this Design Idea at www.edn.com/080501di3. It offers four timing sequences. You can select a timing sequence by strapping pins 4 and 6 (**Table 1**, also at www.edn.com/080501di3).

The 4-MHz crystal maintains a solid instruction-timing reference, and equalized coded branches in the **listing** maintain accurate timing ticks. You can also configure the 12F508 with an internal, 4-MHz RC oscillator. You base the coded loops on a sequence of exactly 25 instructions, and they provide a fundamental, base-reference loop that is exactly 100 instructions. A 16-bit register counter serves as the multiplier to produce the base timing. For use with scopes, you can recode the **listing** with minor changes to use 50 instructions or a 50- μ sec base-timing-tick minimum. The 8-bit registers in the equalized loop provide multipliers to produce the additionally tiered output. The microcontroller uses two output pins, 5 and 7, as a pseudo 2-bit DAC. This configuration generates one of four voltage levels for timing ticks that display continuously, and you can record them along with application data. **EDN**

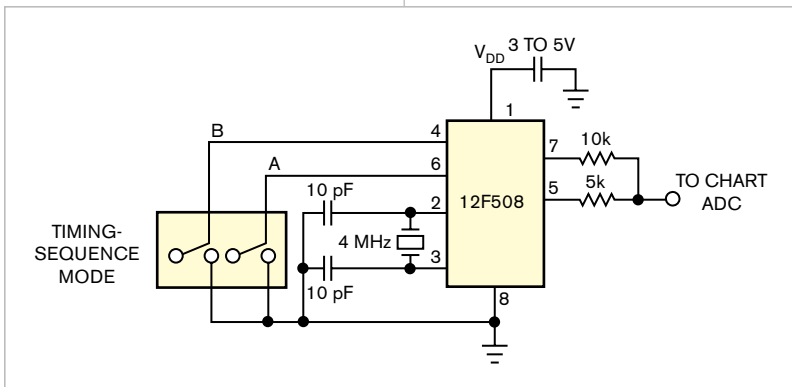


Figure 1 An 8-bit microcontroller with output pins 5 and 7 configured as a 2-bit DAC provides precise timing ticks to annotate captured data.

Fast-settling synchronous-PWM-DAC filter has almost no ripple

W Stephen Woodward, Chapel Hill, NC

An inexpensive way to implement high-resolution digital-to-analog conversion is to combine microcontroller-PWM (pulse-width-modulated) outputs with precision analog-voltage references, CMOS switches, and analog filtering (**Reference 1**). However, PWM-DAC design presents a big design problem: How do you adequately suppress the large ac-ripple component inevitably present in the switch's outputs? The ripple problem becomes especially severe when you use typical 16-bit microcontroller-PWM peripherals for DAC control; such high-resolution PWM functions usually have long cycles because of the large 2^{16} countdown modulus of 16-bit

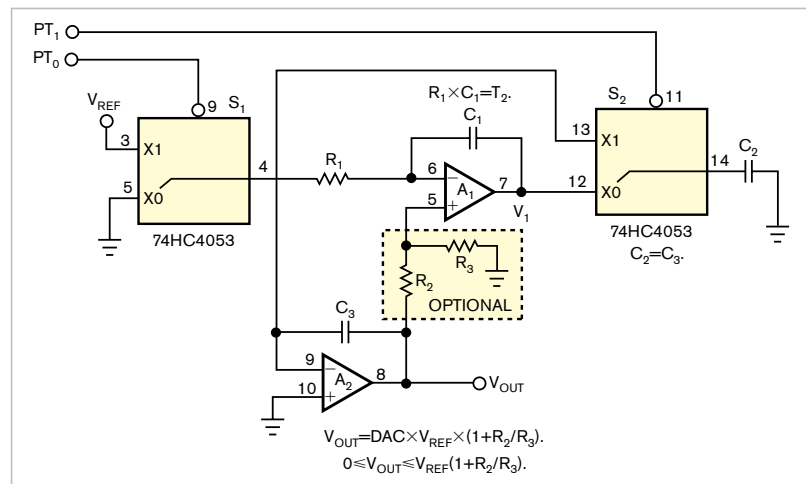


Figure 1 This DAC ripple filter combines a differential integrator, A_1 , with a sample-and-hold amplifier, A_2 , in a feedback loop operating synchronously with the PWM.

timers and comparators. This situation results in ac-frequency components as inconveniently slow as 100 or 200 Hz. With such low ripple frequencies, if you employ enough ordinary analog lowpass filtering to suppress ripple to 16-bit—that is, -96-dB—noise levels, DAC settling can become a full second or more.

The circuit in **Figure 1** avoids most of the problems of lowpass filtering by combining a differential integrator, A_1 , with a sample-and-hold amplifier, A_2 , in a feedback loop operating synchronously with the PWM cycle, T_2 in **Figure 2**. If you make the integrator time constant equal to the PWM cycle time—that is, $R_1 \times C_1 = T_2$ —and, if the sample capacitor, C_2 , is equal to the hold capacitor, C_3 , then the filter can acquire and settle to a new DAC value in exactly one PWM-cycle time. Although this approach hardly makes the

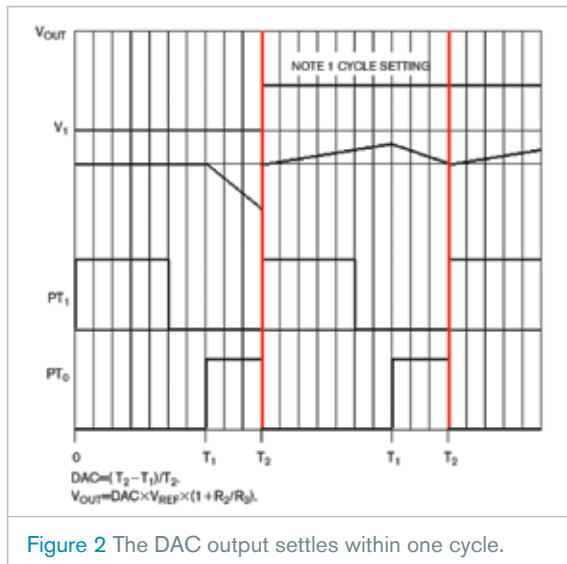


Figure 2 The DAC output settles within one cycle.

resulting DAC exactly “high speed,” 0.01-sec settling is still 100 times better than 1-second settling. Just as important as speed, this improvement in settling time comes without compromising ripple attenuation. Ripple suppression of the synchronous filter

is, in theory, infinite, and the only limit in practice is non-zero-charge injection from S_2 into C_3 . The choice of a low-injected-charge switch for S_2 and an approximately 1- μF capacitance for C_3 can easily result in ripple amplitudes of microvolts.

Optional feedback-voltage divider R_2/R_3 provides flexibility in a DAC-output span with common voltage references. For example, if $R_2 = R_3$, then a 0 to 10V output span will result from a 5V reference. An additional advantage of this method of span adjustment is that output ripple remains independent of

reference amplification. **EDN**

REFERENCE

Woodward, Steve, “Combine two 8-bit outputs to make one 16-bit DAC,” *EDN*, Sept 30, 2004, pg 85, www.edn.com/article/CA454640.

Switched-gain op amp serves as phase detector or mixer

W Bruce Warren, Marietta, GA

Some op amps, such as the AD8041 from Analog Devices (www.analog.com) and the EL5100 from Intersil (www.intersil.com), provide a disable pin, which allows you to parallel the outputs of several op amps for video multiplexing. In addition to this multiplexing, you can also use the disable function to configure the op amp as a phase detector or a frequency mixer. **Figure 1** shows how the disable function can implement a low-frequency phase detector. You can switch the gain of this circuit’s amplifier on and off at the rate of the phase-reference signal. Doing so produces a dc component at the output of the op amp. This component is proportional to the cosine of the phase difference between the phase of the input signal

and the phase of the reference signal.

In the circuit, the output of the op amp is: $V_{OUT}(t) = V_{IN}(t) \times G(t)$, where $V_{IN}(t) = A \cos(\omega_{REF} t + \theta)$, and $G(t)$ is the time-varying gain of the op amp. $G(t)$ is a 50%-duty-cycle square wave that switches from zero to G_0 at the frequency of the phase-reference signal. G_0 is the gain of the op amp when the op amp is enabled. Because $G(t)$ is a time-varying periodic function expand it in a Fourier series: $G(t) = G_0 [1/2 + 2/\pi \{ \cos(\omega_{REF} t) - 1/3 \cos(3\omega_{REF} t) + 1/5 \cos(5\omega_{REF} t) + \dots \}]$.

Multiplying $V_{IN}(t)$ by $G(t)$ and retaining only the dc terms, the dc component of the output is $V_{OUT}(dc) = (AG_0/\pi) \cos(\theta)$.

The EL5100 op amp in **Figure 1** has a 200-MHz unity-gain bandwidth, and

you can turn its output on and off by applying a square wave of at least 0 to 4V to the output-disable terminal, Pin 8. Using the feedback resistances shown and with $G_0 = 3$, the peak output voltage of the phase detector is approximately equal to the peak value of the input signal. The EL5100 has a disable time of 180 nsec and an enable time of 650 nsec, which allows you to gain-switch the device to approximately 250 kHz. At higher frequencies, the gain of the phase detector falls off because the gain-switching no longer has a 50% duty cycle.

The lowpass filter following the op amp extracts the dc component of $V_{OUT}(t)$ and has a 3-dB point at 800 Hz. A 100 Ω resistor in series with the 0.1- μF shunt capacitor limits the phase lag of the filter when the phase detector is inside a PLL (phase-locked loop). The values in **Figure 1** provide a maximum phase lag of approximately 65°. Using 5 and -5V power sup-

plies allows the output swing of the phase detector to be symmetric at approximately 0V. If your design doesn't require this symmetry, you can use a single 5V supply with 2.5V positive offset-biasing of the op amp. In this case, the output swing is symmetric with respect to 2.5V. As with all wide-bandwidth-op-amp circuits, you should take care to connect the power-supply bypass capacitors to ground with short connections and as close to the op amp's power-supply pins as possible to avoid instability.

This same gain-switching scheme also works as a frequency mixer. If the input signal is at frequency ω_s and the reference-square-wave input is at frequency ω_o , the IF output signal is $(\omega_o - \omega_s)$ or $(\omega_o + \omega_s)$. You obtain the desired IF signal by replacing the output lowpass filter in **Figure 1** with a bandpass filter tuned to the desired IF frequency of $\omega_o \pm \omega_s$. If the switching rate for the reference signal is higher than the disable function can provide, then you can use

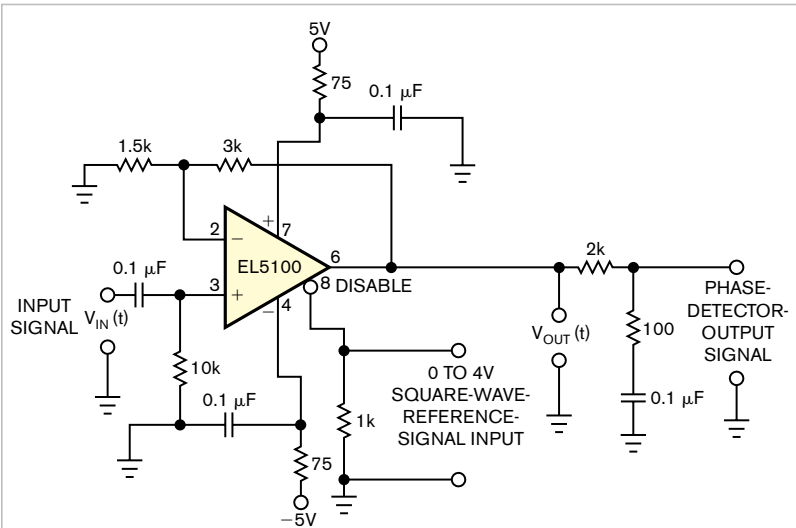


Figure 1 By switching the disable input of the op amp at a reference frequency and lowpass-filtering its output, you can obtain a dc voltage proportional to the phase difference of the switching frequency and the input frequency.

the harmonic mixing using the odd-order harmonics of the reference signal. This approach reduces the gain

of the mixer by a factor of $1/N$, where N is the number of the harmonic you are using. **EDN**