CIRCUIT CELLAR INK®
THE COMPUTER APPLICATIONS JOURNAL

DESIGN CONTEST WINNER

FEATURE ARTICLE

Rick May

A PIC-Based AC Power Meter

Questioning your power bill? Rick shows you how to build a tool to make sure your power bill stays right on target—a portable AC power meter that displays the power delivered to and consumed by your house.

When I watch the old mechanical wattmeter on the house spin, I marvel at just how low tech this device is. No fancy displays, no RF transponders sending telemetry to roving meter-reader trucks (at least not in my neighborhood).

I always wondered why I couldn't go out and buy a small hand-held instrument that I could use to figure out just what was causing that old-fashioned meter outside to spin so fast.

A few years ago, I came across a novel circuit design by Stephen Woodward [1]. He used a quad optoisolator (conventionally a nonlinear device) to generate an analog voltage proportional to the power consumed by a load. It used optos for safety, and, well, it was just neat.

So, off I set to marry some type of micro to that analog front end. I wanted to build a hand-held, portable AC power meter that could display the power delivered to a load. The result: see Photo 1.

By tossing in a little math and some numerical integration, I could also display the energy consumed by a load. For a little added challenge, I used a PIC microcontroller that has no multiply or divide instructions.

SYSTEM DESIGN

I want my power meter to measure AC power, instantaneous and average, 0–1200 W, as well as measure AC energy consumption in kilowatt-hours (or watt-hours). Additionally, I want it to provide digital readout of power or energy, and it should be easy to hook up using standard power receptacles and plugs.

This device is to be packaged as a hand-held instrument, and my budget dictates some ultra-low-cost parts.

I designed the power meter around the Woodward power-measurement circuit. The power source is a 9-V battery, instead of stealing power from the AC line. This setup ensures maximum isolation from the AC line and increases safety.

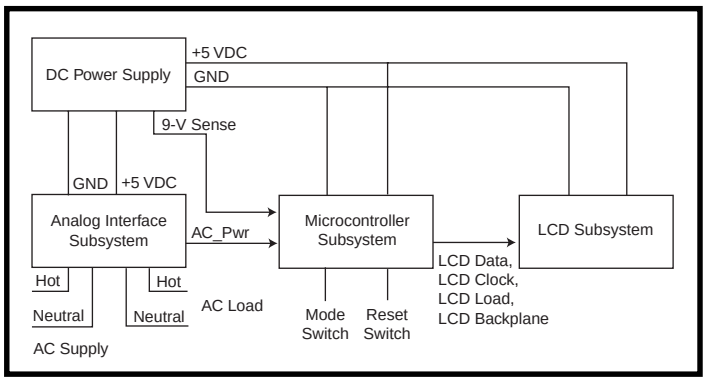

Figure 1—*The power meter is broken down into four subsystems. The user interface is accomplished with just two switches and an LCD.*

I chose a nonmultiplexed LCD

Photo 1—*Here's my dream come true—a hand-held portable AC power meter.*

module because I only need to display numeric data. It also costs less.

The user operates the meter using two momentary switches. The mode switch causes the display to cycle though the different operational modes. Regardless of what operational mode the power meter is in, energy-consumption accumulation still occurs. The reset switch resets the energy-consumption accumulator in any operation mode.

This power meter has four operational modes. The power mode displays the power consumed in watts. By accumulating power over time, the energy mode displays watt-hours or kilowatt-hours. The average power mode displays energy consumed over time in watts. The remaining mode displays the time elapsed since reset in hours, minutes, and seconds.

Because the power meter toggles through these four modes, the user needs to know the current operational mode. Normally, this would be done with annunciator segments in the LCD. However, I want to use low-cost stock parts, so a custom LCD with annunciator segments isn't an option.

The LCD I chose is a four-digit display with three decimal points, a colon, and an arrow. Because I have no real use for the arrow, I decided to use it as the mode indicator.

The colon is used in the time mode, so I only need to discriminate among three modes—power, energy, and average power. I decided that no arrow indicates power mode, a slow-flash arrow means energy mode (1-Hz rate), and a fast-flash arrow is used for the average-power mode (2-Hz rate).

HARDWARE DESIGN

Figure 1 shows the four subsystems of the power meter. The power-supply subsystem supplies DC power to the other subsystems. The analog interface contains Woodward's circuit [1].

The microcontroller and A/D subsystem acquire data and compute power readings that are then displayed by the LCD subsystem. Figure 2 shows how the subsystems link together.

A 9-V battery connects to J6 that feeds a 78L05 regulator. The 5.1-kΩ 1% resistors in a voltage divider network provide the 2.5-V bias voltage for the analog section. They also provide the 9-V sense voltage to the microcontroller for low-battery detection.

A resistor divider network is selected over a 2.5-V reference diode because of cost to provide the 2.5-VDC reference used in the analog section. I chose 5.1-kΩ resistors since I'm already using this value in the analog interface.

Woodward's original circuit used ±15-VDC power supplies and an OP27 precision op-amp. My goal was to modify his circuit using a +5-VDC single supply and a common LM358 op-amp.

The redesign for single-supply operation is straightforward: swap 2.5-V DC bias for ground, ground for –15 V, and +5 V for +15 V. Woodward's circuit indicates power delivered to and from the load, with power delivered to the load being below the 2.5-V bias.

Power readings range between ground and +2.5 V, effectively reducing the ADC resolution to seven bits. Therefore, 128 discrete output values are possible between no load and full scale.

A pot, P1, calibrates the full-scale value to approximately 1200 W nomi-

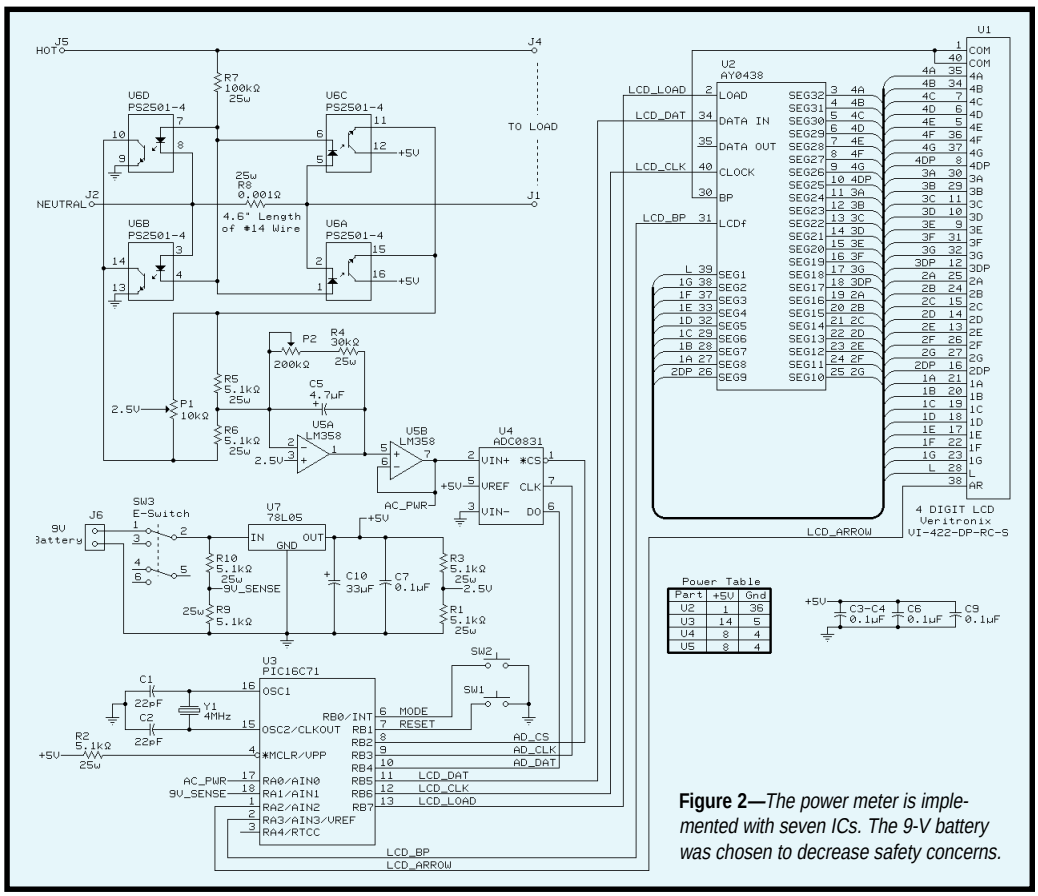

Figure 2—*The power meter is implemented with seven ICs. The 9-V battery was chosen to decrease safety concerns.*

nal. By adjusting the pot to give a full-scale reading of 1280 W, the actual power (in watts) is obtained by:

$$\text{power} = (0x80 - \text{ADvalue}) \times 0x0A$$

So, if *ADvalue* equals 0x76, the power is 100 W.

I initially chose the PIC16C71 because of its low cost and onboard four-channel eight-bit ADC. However, you can save even more by replacing the '16C71 ($12.30) with a separate ADC, National's ADC0831 ($3.29), in combination with the ADC-less PIC16C61 ($6.15).You save $2.86, but you need space for one more 8-pin DIP.

The National ADC0831 is a low-cost 8-bit ADC that has a simple three-wire serial interface (chip select, clock, and data) and a 32-µs conversion time.

Note there are no pull-up resistors on the two momentary switches. Pullups are provided internally by the PIC16C61. Again, you see the 5.1-kΩ 1% resistor used as the MCLR pullup, which means the MCLR pin can be shorted to ground without shorting out the +5-V supply.

This component could be replaced with a 0-Ω jumper for production. Currently, a crystal is used as the microcontroller clock, but you could replace it with a ceramic resonator for more cost savings.

The display subsystem uses a non-multiplexed Varitronix LCD, with a Microchip AY0438 LCD driver. The AY0438 operates up to 32 segments of an LCD, providing a simple three-wire serial interface (data, clock, and load), and it's capable of generating the AC waveforms required to illuminate LCD segments.

For a LCD segment to be on, there must be voltage differential between the segment pin and the backplane. However, this voltage cannot be static (non-time-varying) or the display gets permanently damaged. To drive the LCD correctly, a low-frequency (100 Hz) square wave is applied to the backplane pin.

For a segment to be on, a segment pin must have the inverted backplane signal applied. For a segment to be off, the segment pin must have the in-phase backplane signal applied.

a)

	7	6	5	4	3	2	1	0
PORT_A	N/A	N/A	N/A	X	ay_bp	ay_arr_out	9V_sense	ac_pwr

ac_pwr	Analog input	Analog AC power input (PIC16C71 only)
9V_sense	Analog input	Analog 9-V battery sense (PIC16C71 only)
ay_arr_out		Arrow segment pin of LCD
ay_bp		LCD backplane input pin

b)

	7	6	5	4	3	2	1	0
PORT_B	ay_ld	ay_clk	ay_dat	ad_dat	ad_clk	ad_cs	res_sw	mode_sw

mode_sw	Active low	Indicates mode switch depressed
res_sw	Active low	Indicates reset (or zero) switch depressed
ad_cs	Active low	Chip select to ADC0831
ad_clk	Falling edge	Data valid out of ADC on falling edge
ad_dat		Data output of ADC0831
ay_dat		Data input to AY0438 LCD driver chip
ay_clk	Falling edge	Clock input to AY0438, data is clocked into AY0438 on falling edge
ay_ld	Rising edge	Data is transferred from AY0438 shift register to output latches

Figure 3a—*Here are the port A assignments. With just 13 I/O pins available, allocation is critical.* ***b—****With the port B pin assignments, just three pins are needed for the A/D conversion, but five are required for the LCD controller.*

The AY0438 generates this backplane waveform and the correct segment waveforms without processor involvement. Just hang a capacitor on the LCDf pin to control the onboard oscillator.

Initially, I used the onboard oscillator of the AY0438 to generate the AC waveforms needed and to get the display up and running. However, the AY0438 can only drive 32 segments.

I needed 33 segments to use the arrow segment and all four digits, three decimals, and the colon. With a couple of unused pins on the PIC, I figured it couldn't be that hard to drive the LCD directly.

And, driving the LCD is easy. Just remove the cap from the LCDf pin, then drive this pin (backplane) and the thirty-third segment directly from the PIC. Remember that you can't drive the LCD segment pin alone because the AY0438 oscillator (backplane) would be asynchronous with respect to the PIC.

The software must toggle the backplane and the thirty-third segment every 10 ms. Note that the AY0438 still manages the other 32 segments and provides a serial interface. The AY0438 just gets its backplane frequency reference from the PIC. This is a case where a watchdog timer should be used because display damage can result if the backplane doesn't toggle at least every 10–100 ms.

Register definitions of external interfaces to the PIC are shown in Figure 3.

The PIC16C61 is an 18-pin DIP and only has 13 I/O pins. Obviously, when you're using an I/O-limited micro like the PIC, it's important to choose peripherals with low pin-count interfaces.

Five pins are used for the LCD subsystem. It could have been three if I didn't need the thirty-third segment. Three pins are used for the ADC and two for the mode switches.

SOFTWARE DESIGN

The PIC's internal timer generates a 10-ms timer interrupt for periodic software functions. This works well since I need to service the LCD backplane in the 10–100-Hz range, and it also lets me sample power at a 100-Hz rate.

Figure 4 shows a flowchart of the 10-ms timer ISR. Every 10 ms, the LCD backplane and arrow segment must be toggled, the tic (0.01 s) counter incremented, and power value accumulated into *S*.

Every 250 ms, the fast flash of the arrow segment is done if needed. Every 500 ms, the slow flash of the arrow segment is done if necessary, and the power value computed for display.

Figure 5 illustrates the main-loop processing. Both switches are scanned every time through the main loop, debounced through a 50-ms delay.

Every second, the larger 24-bit energy accumulator, *E*, is updated from the local 16-bit accumulator, *S*. The mode state variable directs execution to the appropriate processing for the four operational modes.

MEASUREMENT CONSIDERATIONS

The raw value from the ADC is read at a 100-Hz rate or every 10 ms and is in the 0–127 range. This value indicates a 10-W increment of power (so, 12 means 120 W). Let's refer to this raw A/D value as having the unit of a ten-watt (i.e., 10 W).

To display a power reading, the main loop reads the current value of power that the ISR writes to, multiplies this value by 10, converts this result to a five-digit BCD word, and then displays the four least significant digits of the result. The multiply requires 16-bit math because 10×127 equals 1270, which doesn't fit into 8 bits.

I found the 16-bit math and BCD conversion routines on the Microchip BBS. The routines are fairly compact in size and require approximately 300 cycles (300 µs at 4 MHz) for an unsigned multiply or divide.

Since the 16-bit values need to be converted to BCD prior to display, and the BCD routines return a five-digit result, a divide by 10 is done just by displaying the upper four digits instead of the lower four digits. This fact is taken into account when designing scaling algorithms.

For energy consumption, a 24-bit accumulator is required to capture a reasonably large amount of energy consumption, given the 100-Hz accumulation rate. To keep the non-reentrant math routines out of the timer ISR, the timer ISR accumulates the power value (0–127) into an intermediate 16-bit accumulator, *S*. The main-loop routine adds *S* to the master 24-bit accumulator, *E*, when the `1_sec_ elapsed` flag is set.

The *S* value is in units of tenwatt-seconds. To accumulate a wide range of energy consumption, the 24-bit *E* accumulator is in units of milliwatt-hours. This allows a maximum value of $2^{24} - 1$ = 16,777,215 mWh or 16.77 kWh.

The conversion from tenwatt-seconds to milliwatt hours is:

$$\text{milliwatt-hour} = \text{tenwatt-second} \times \frac{1\,\text{h}}{3600\,\text{s}} \times \frac{10\,\text{W}}{1\,\text{tenwatt}} \times \frac{1000\,\text{mW}}{1\,\text{W}}$$

$$= \text{tenwatt-second} \times \frac{100}{36}$$

However, since the *S* accumulator represents 100 samples over a 1-s period, we must divide *S* by 100 before this conversion. So, the *S*-to-*E* accumulation calculation (performed only once per second) is:

$$E = E + \frac{S}{36}$$

in milliwatt-hours. Note that this conveniently reduces a multiply and divide operation to a single divide.

To filter transients in power-measurement mode, a 500-ms moving average is implemented. The *S* accumulator is restarted every second as part of the energy-consumption function. At 500 ms after restart, a power value is calculated by:

$$\text{power (watts)} = \frac{S}{50} \times 10 = \frac{S}{5}$$

At 1 s after restart, a power value is calculated by:

$$\text{power (watts)} = \frac{S}{100} \times 10 = \frac{S}{10}$$

To ensure that no-load situations read zero and that noise does not cause false readings, hysteresis is added to cause readings less than 2 (20-W reading) to be treated as zero. This also helps in setting the zero trim pot.

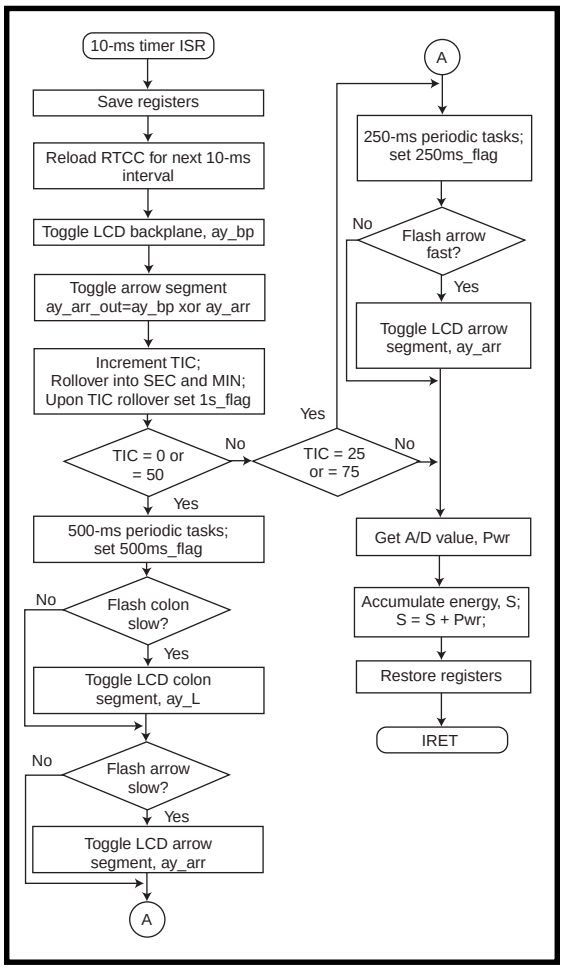

Figure 4—*For every 10-ms period, energy accumulates in S until the `1s_flag` is set.*

DISPLAY CONCERNS

Now that I've discussed how the *E* accumulator is maintained, I want to turn to how the energy accumulation is displayed.

If watt-hour readings are the finest resolution displayed, then the user might have to wait 2 min. to see 0.001 kWh, depending on the load. To solve this problem, I implemented autoranging using three display modes.

Display mode d1 displays readings in the range of 0.000–9.999 Wh. Once 10 Wh are accumulated, display mode automatically switches to d2 mode.

The d2 mode displays readings in the range of 0.010–9.999 kWh. Once 10 kWh are accumulated, the display mode automatically switches to display mode d3, which displays readings in the range 10.00–16.77 kWh.

Display mode d1 is implemented by converting the low-order 16 bits of *E* (milliwatt-hours) to BCD, then displaying the four least significant digits with the decimal point three places to the left (i.e., 0.000). The move of the decimal point effects a multiply by 1000, causing watt-hours to be displayed.

Display mode d2 requires display of kilowatt-hours, and has a least significant digit of watt-hours. Divide *E* (in milliwatt-hours) by 1000 to get the result in watt-hours. This is implemented by using the upper 16 bits of *E*, effecting a divide by 256 operation on *E*.

Recall that canned math routines support 16-bit math, not 24-bit math. So, by taking the upper 16 bits of *E* and performing a divide by 4, a divide by 1024 is

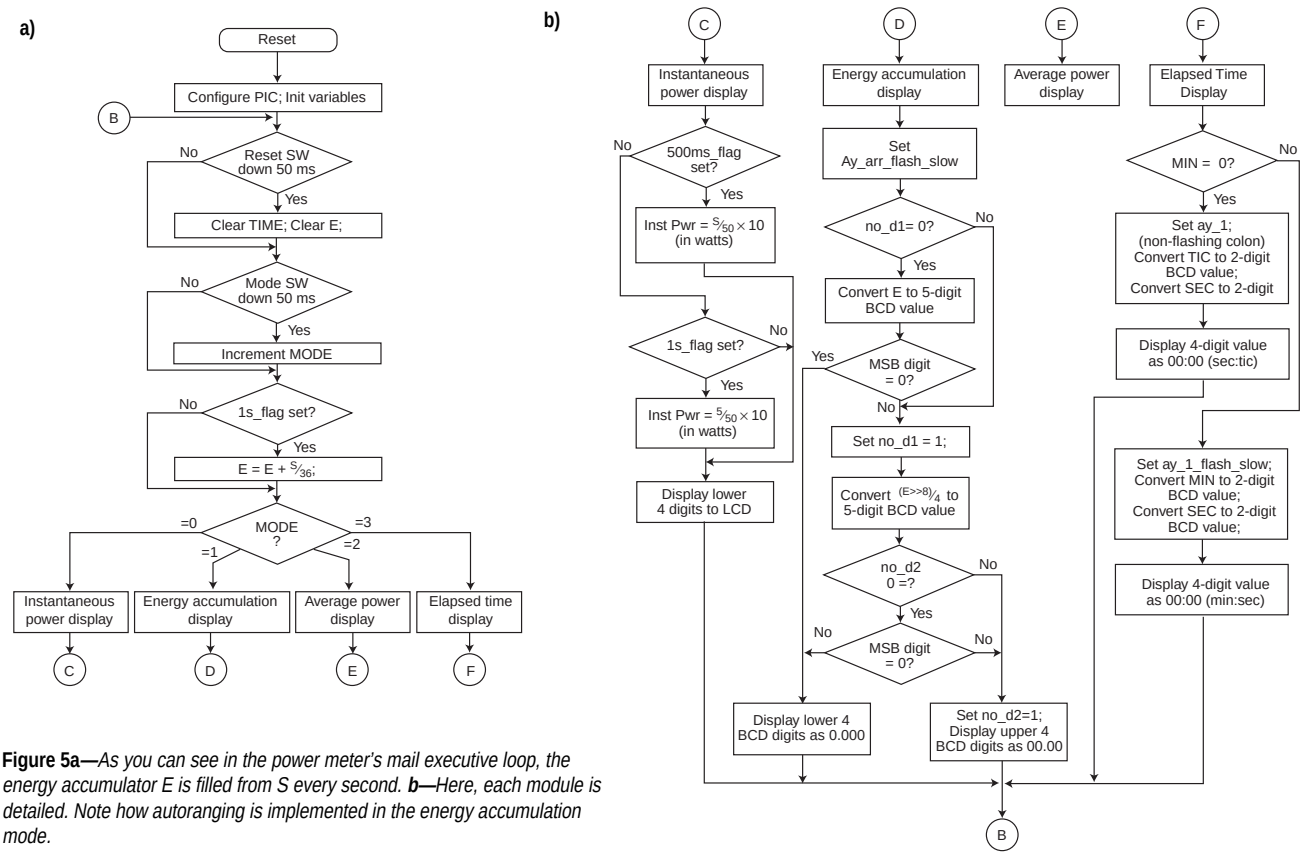

Figure 5a—*As you can see in the power meter's mail executive loop, the energy accumulator E is filled from S every second.* ***b—****Here, each module is detailed. Note how autoranging is implemented in the energy accumulation mode.*

implemented, which approximates the correct divide by 1000. This result is converted to BCD, and then the four least significant digits are displayed with the decimal point set three places to the left.

Display mode d3 also requires display in kilowatt-hours. The only difference between the d2 and d3 modes is that d3 needs an additional divide by 10 to provide a reading in the range (10.00–16.77). By taking advantage of the fact that the BCD conversion routine returns a five-digit result, a divide by 10 is done just by displaying the four most significant digits and placing the decimal point two places to the left.

The LCD services are designed such that the LCD routines expect data where the BCD math routine deposits its result. Although I don't elaborate on these routines in detail here, I mention them because they cooperate with the PIC math routines well.

MY TOP PIC

The PIC software occupies 722 of 1024 words of program memory. I haven't yet implemented average power measurement or low-battery detection. And, I did the initial calibration using incandescent light bulbs as loads.

The results appear accurate within 10 W, as Woodward's article indicates. Calibration using the two trim pots was fairly easy. For easier calibration, a multi-turn pot could replace the single-turn trim pot.

Some possible enhancements include use of a higher resolution ADC for more accuracy, removal of pots altogether, larger energy consumption accumulator, and autocalibration.

You just saw how a simple PIC with Microchip's math routines can be made to do significant computation such as numerical integration. You can use this project as starting point for any instrumentation-type project.

And maybe now I can figure out just what's making my wattmeter spin so fast. ■

Rick May is a principal design engineer at Raytheon Systems. He currently designs embedded software for a Navy communications system. You may reach him at rmay@televault.com.

SOFTWARE

Source code for this article can be downloaded from the Circuit Cellar Web site.

REFERENCE

[1] W.S. Woodward, "Optical isolator computes watts," *Electronic Design*, 102–103, October 14, 1994.

SOURCES

PIC16C71, PIC16C61, AY0438
Microchip
(602) 786-7668
Fax: (602) 786-7277
www.microchip.com

ADC0831
National Semiconductor Corp.
(800) 272-9959
(408) 721-5000
Fax: (408) 721-2233
www.national.com

LCD
Varitronix
(213) 738-8700
Fax: (213) 738-5340
www.varitronix.com

©Circuit Cellar INK, the Computer Applications Journal. Reprinted by permission. For subscription information, call (860) 875-2199 or subscribe ©circellar.com