Port J Bit 1 to be an output. But now that you know C, it’s easier to read the systems software manual to get the job done. Port J is now configured. Ta-da!

CODE & BIT MANIPULATION

Next let’s take a look at the code to manipulate Port J bits 0, 1, and 2 to control the SPI to the driver LED and then read the results on bits 4, 5, 6, and 7. I started by making routines for each output bit.

Listing 2 includes the routines for setting the SPI clock bit high and low. I extracted all this so that I just call RockerClkHi() or RockerClkLo(). It may seem like a lot of work and additional overhead, but when we get to where we use these routines that code will read much better. Note that we could have used defines something like this:

#define RockerClkHi() GPIOPinWrite(GPIO_PORTJ_BASE, RK_CLK, RK_CLK)

This is should be just a simple text substitution. I have not tested this define construct. Each compiler might perform in a manner other that you expected. I see many of the engineers that I work with using this technique. And now it's time for you to start thinking about your own designing style.

I wrote routines for each output bit and one for a delay. If you look at the data chip for the STMicroelectronics STP16CP05MTR sink driver, you’ll see some timing requirements such as maximum clock frequency of 30 MHz. In order to meet all these requirements, we need to insert delays into the code. I defined a routine to do just that (see Listing 3).

I just put in a busy loop for the first implementation of this routine. It's something to keep the CPU occupied just wasting clock cycles. In the example, I incremented the variable i 10,000 times. That provided about 1 ms delay, which was where I wanted to be. This is a very poor way to implement a delay loop, but I had code running on Day 1. When you start using the compiler's optimization levels, this code could get compiled out for run speed. And that's probably not what you'd expect. You’ve seen me implement this code with timer interrupts and state machines. But for now, let’s just use this delay.

Where in the world did I discover routines such as GPIOPinWrite()? It’s all in the peripheral driver library user's guide. With the Stellaris family of CPUs, TI has the StellarisWare software suite. Go to TI's website to search for and download the latest copy. Realize that it's about 400 pages of information. In this library, you'll find descriptions for routines that work with all peripheral devices in the chip. But wait, there's more. Some of the Stellaris CPUs come with the peripheral library in ROM. When in ROM, these routines do not use any of the flash memory. You can start debugging with the routines in flash memory. Single step through the code and use break points since you have the source. Then, link with the ROM version as your code gets tested.

TURN LOOSE THE RTOS

The code for the entire interface to the keyboards is available on the Circuit Cellar FTP site (ReadTheSwitches.c). Look to see how I implemented the timing in Figure 1. There is, of course, more to getting the board up and running. We need to set up the clock and at least blink an LED. For now, get the kit out if you've got one, download the system files, or contact a distributor and get an evaluation board. Look at Freescale's ColdFire, TI's Stellaris, or any other that is a complete monolithic solution.

There's a whole lot more that comes with any of these new highly integrated CPUs. In the second part of this series, I'll describe how to get a program up and running and then turn loose the RTOS.

George Martin (gmm50@att.net) began his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and co-founded a design and manufacturing firm (www.embedded-designer.com). His designs typically include servo-motion control, graphical input and output, data acquisition, and remote control systems. George is a charter member of the Garcia Design Works Team. He is currently working on a mobile communications system that announces highway info. He is also a nationally ranked revolver shooter.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2010/244.

RESOURCES


SOURCES

ColdFire Microcontroller
Freescale Semiconductor, Inc. | www.freescale.com

STP16CP05 Constant current LED sink driver
STMicroelectronics | www.st.com

Stellaris EKK-LM3S9B96 Evaluation Kit
Texas Instruments, Inc. | www.ti.com
Recharging Portable Devices

A DIY Power Adapter Design

When traveling with multiple portable devices, having the ability to recharge quickly and easily is key. But why buy AC adaptors when you can design your own all-in-one solution? Here's how.

Our constant use of portable devices means frequent recharging. Most products come with an AC charger. For some devices, car charger adaptors are available as well. Ten years ago, I needed to stay in touch while spending a week at Boy Scouts of America (BSA) camp (“Sharing Technology with Mother Nature: Out of State with an Internet-Compatible Cell Phone,” Circuit Cellar 125, 2000). My Motorola StarTAC flip phone had a serial connection, and I brought a laptop with modem [remember those?] so I could get and send e-mail through the phone. I took a used motorcycle battery with a 12-V car socket attached so I could plug in both my phone and laptop and keep them charged.

This July, I went to the BSA’s 100th anniversary National Jamboree, which was held, for the last time, at Fort AP Hill in Virginia. This time, I had a laptop, a Droid, a still camera, and a video camera. Sure, I had AC adaptors for each piece of equipment, but I would’ve also needed four separate DC adaptors. Instead, I brought a “one size fits all” adaptor that I built at my workbench.

PROGRAMMABLE REGULATOR

I doubt there are any circuit builders who are unfamiliar with the three-terminal regulator [see Figure 1a]. Using one is a no-brainer. You apply an unregulated voltage between the input and ground terminals, and you get any [lower] regulated voltage between the output and ground terminals. Even if you’ve used a fixed voltage regulator, you...

---

Figure 1a—Here’s the standard 78xx fixed output linear regulator that’s used just about everywhere. It’s easy to use and comes in many varieties, including different package styles for low- and high-current applications, and plenty of fixed output voltages to choose from. b—The LM1/2317 adjustable linear regulator is similar to the standard fixed voltage regulator, but it can be set to your choice of regulated voltages from 1.2 up to 37 V (depending on your supply voltage) based on two external resistors.
may not be familiar with its counterpart, the adjustable three-terminal regulator. You can think of the adjustable regulator as a 1.2-V fixed output regulator when the adjustment terminal is connected directly to ground. While the adjustable regulator requires two additional resistors to set the output voltage to something higher than 1.2 V, that's a pretty cheap price to pay for its flexibility.

To increase the output voltage, you add a resistor voltage divider network between the output terminal and ground with the divider junction connected to the adjustment terminal of the adjustable regulator, like the LM317 in Figure 1b. The upper resistor between the output and adjustment terminals creates a current due to the 1.2 V across it. This current also flows through the lower resistor (along with a small bias current provided by the adjustment terminal) creating a voltage across the lower resistor raising the adjustment terminal above ground and the output 1.2 V above that. So, this voltage across the lower resistor, plus the 1.2V across the upper resistor, equals the new regulated voltage.

The upper resistor's value is chosen to give the LM317 a minimum load of about 5 mA. You can see by the formula for calculating resistor values for a required VOUT that with a 5-mA load through the resistor divider, the 50 µA bias current will have very little effect:

\[ V_{\text{OUT}} = 1.2V + \left( \frac{1.2V \times R_2}{R_1} \right) + 50 \mu A \times R_3 \]

A simple calculation will provide a good start for determining R2, the upper resistor: 1.2 V/5 mA = 240 Ω. If you are designing for a 5-V VOUT, the bottom resistor will need to drop 3.8 V (i.e., 5 V - 1.2 V). The resistor will need to be 760 Ω (i.e., 3.8 V/5 mA).

**PROGRAMMABLE RESISTOR**

At the heart of the project is a programmable resistor. My initial thought was to use a digital potentiometer. But further investigation revealed that the pot was not as isolated from the control circuitry as you might think. The potential on any connections must remain within the supply voltage (usually around 5 V).

The circuit I developed has voltages that exceed twice that. So, I couldn't use such devices.

I now can say that I've read the fine print. However, this seemed like such an ideal match for this design, that I saw just what I wanted to see and jumped ahead designing with it. It's a fairly straightforward design. A microcontroller is used to read a couple of miniature rotary BCD switches. These set the desired voltage: one switch for the integer volt value and one switch for the tenths of a volt.

An internal ADC makes it possible to read both battery voltage and regulated voltage. What better way to dis-
3 PORT INTERFACE
RS-485 to Ethernet Converter

Only
$170

RS-485 to Ethernet Converter

Powerful feature
- Protocol converter RS485 between Ethernet
- Offer TCP/IP Communication to Devices with RS485 I/F

<table>
<thead>
<tr>
<th>Specification</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>TCP, UDP, DHCP, ICMP, IPv4, ARP, IGMP, PPPoE, Ethernet, Auto MDI/MDIX, 10/100 Base-TX Auto negotiation (Full/half Duplex)</td>
</tr>
<tr>
<td>Serial</td>
<td>RS485 3 Ports, 1,200~115,200 bps, Terminal block I/F Type</td>
</tr>
<tr>
<td>Control program</td>
<td>IP Address &amp; port setting, serial condition configuration, Data transmit Monitoring</td>
</tr>
<tr>
<td>Accessory</td>
<td>Power adapter 9V 1500mA, LAN cable</td>
</tr>
<tr>
<td>Etc</td>
<td>- DIP Switch (485 Baud Rate setting) - LED: Power, Network, 485 Port transmission signal</td>
</tr>
</tbody>
</table>

www.icbank.com
MP3P DIY KIT, Do it yourself

(Include Firmware Full source Code, Schematic)

**myPIC**

- **$160** qty 100
- **$220** qty 1

**myWave (MP3 DIY KIT SD card Interface)**

- **Only**
- **$150** qty 100
- **$200** qty 1

**myAudio (MP3 DIY KIT IDE)**

- **Only**
- **$180** qty 100
- **$220** qty 1

---

**Powerful feature**

- MP3 Encoding, Real time decoding (320kbps)
- Free charge MPLAB C-Compiler student-edition apply
- Spectrum Analyzer
- Application: Focusing for evaluation based on PIC
- Offer full source code, schematic

**Specification**

- Microchip dsPIC3F2556GPF710 / 16-bit, 40MIPS DSC
- VLSI Solution VS1033 MP3 CODEC
- NXP UDA1330 Stereo Audio DAC
- Texas Instrument TPA6110A2 Headphone Amp (150mW)
- 320x240 TFT LCD
- Touch screen
- SD/SDHC/MMC Card
- External extension port (UART, SPI, I2C, I2S)

**myWave (MP3 DIY KIT SD card Interface)**

- Play, MP3 Information, Reward, forward, Vol+/-
- Focusing for MP3 Player
- SD Card interface
- Power: battery
- Offer full source code, schematic

**Item**

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td>Atmel ATmega128L</td>
</tr>
<tr>
<td>MP3 Decoder</td>
<td>VS1002 / VS1003 (WMA)</td>
</tr>
<tr>
<td>IDE Interface</td>
<td>Standard IDE type HDD (2.5&quot;, 3.5&quot;)</td>
</tr>
<tr>
<td>Power</td>
<td>12V, 1.5A</td>
</tr>
<tr>
<td>LCD</td>
<td>128 x 64 Graphic LCD</td>
</tr>
<tr>
<td>Etc</td>
<td>Firmware download/update with AVR ISP connector</td>
</tr>
</tbody>
</table>

**myAudio (MP3 DIY KIT IDE)**

- Play, MP3 Information, Reward, forward, Vol+/-
- Focusing for full MP3 Player (Without case)
- IDE Interface
- Power: Adapter
- Offer full source code, schematic

---

Youngsan Cho / Senior Account Manager / Sales & Marketing ICbank Inc.
Tel: 82-707-019-2024, Fax: 82-2-2340-1000, Cell: 82-17-238-2078, Email: youngsan@icbank.com
18F Ace High-End 2, 222-14 Guro Digital Valley, Guro-dong, Seoul, 152-060, Korea
Website: http://www.icbank.com/
play these values than by a small LCD? I have a 1 x 8 display that will work well. Figure 2 shows my original design using a digital potentiometer. Can this circuit be saved by a device that does not exceed its maximum ratings?

CADMIUM SULFIDE

My first introduction to cadmium sulfide (CdS) cells came around 40 years ago when I was working for Electronic Music Labs. A new synthesizer being developed was going to use a toggle switch for selecting one of two audio signals. The design’s switches were creating audible “pops” when switching between signals. A quiet switch was developed using a combination CdS cell and incandescent bulb prepackaged together. I forget what it was called at the time. An SPDT switch was wired to alternate power to the bulbs of two of these devices. The relatively long time constants of the incandescent bulb and their effect on the resistance change of each CdS cell produces a nice quiet transition between the signals.

The CdS cell has very high resistance (greater than 1 MΩ) when the sensor is in darkness and this is considerably reduced as light is allowed to fall on the sensing surface. As you might guess, when the device has a lot of light and the resistance is small, current flow may become an issue. The physical size of the device has a relationship to its power rating. The largest I purchased have a maximum rating of approximately 0.5 W at about 10 mm in diameter. With 10 V across a resistance of 200 Ω would produce a current of 10 V/200 Ω, or 50 mA. The power would be 10 V x 50 mA, or 0.5 W.

Similar to solar cells, a CdS cell uses photon energy to affect electron movement. However, while a solar cell creates a current via photons giving electrons the energy needed to cross its PN junction, the CdS cell has no junction and the N-type material becomes more conductive due to the increased electron energy thanks to the photon. The CdS cell is essentially a light dependant resistor (LDR). By placing an LDR across the lower leg of the regulator’s resistor divider network, you get a way to change the resistance of this leg and thus the regulator’s output voltage.

To use the LDR in this circuit, you need to control its light source. The micro has a PWM peripheral, so I powered an LED from the PWM’s output pin to achieve complete control over its intensity. A series resistor will limit the maximum intensity [at 100% PWM] of the LED, which will keep the LDR from entering a region that exceeds its maximum power dissipation (smallest resistance).

DIGITAL REGULATOR

In the future, you might find voltage regulators that contain their own microcontrollers; but for now, you’ll have to design your own. My original design (using the digital pot) was to be table-driven—that is, pick your voltage using the BCD switches and look up [or calculate] the appropriate resistance to set the bottom resistor of the regulator’s divider network. This new design won’t calculate anything, it will be self-adjusting. If you compare the actual output voltage with the user setting [BCD switches], you can call for more light [higher PWM] if the actual voltage is too high and less light [lower PWM] if the output voltage is too low.

The key to this design is making an LED/LDR module. I began using an LED light source to see how each of

Photo 1—The cadmium sulphide cell, or light-dependant Resistor (LDR), was a saving grace when coupled with a red LED.

Photo 2—I’ve dialed in “5” and “0” on the BCD switches, and the microcontroller is measuring and displaying the output voltage. It must compare this voltage with the BCD setting and adjust its PWM output to whatever duty cycle is necessary to drive the connected LED with an intensity level that will keep the LDR at the required resistance, and to produce a voltage that matches the BCD setting.
the LDRs I had purchased reacted to this source of illumination. The spectral range of the LDR is amazingly close to the human eye (approximately 400 to 700 nm). The red LED I had handy will reduce all of the LDRs values down to less than 1 kΩ with approximately 10 mA through the LED. I chose the best alternative as a compromise between lower resistance and the higher wattage (Advanced Photonix PDV-P7002).

It is important to package the LDR and LED in a light-proof container to prevent any stray light from creating an uncontrolled variable in the photons-to-resistance conversion. A short piece of shrink tubing created a stable environment for these two devices holding each in proper alignment (see Photo 1). Now let’s see how this all fits together.

**IT’S ALL ROUTINE(S)**

This design is nothing but routines. Read the user input BCD switch settings. Display status on the LCD. Take analog-to-digital samples (Timer 1 Interrupt). Determine if the PWM requires any adjustment (A/D conversion complete Interrupt). The first two routines happen as time allows, while the final two are handled on an as required basis. The main loop handles reading the user input from the switches and outputting status to the LCD as the lowest of priorities.

Notice that the parallel LCD uses a 4-bit data bus. Also note that this bus is shared with the two BCD switches. The commons for each BCD switch are tied to their own output pin on the microcontroller and the BCD binary switch pins are connected through diodes to the data bus. In order to read to and write from the LCD, the data bus must not be affected by any other device connected to it. Wire-ORing each switch with diodes prevents any data line which is being driven low from affecting another data line presently being shorted together through a switch, as in D0 and D1 when a switch is in position 3 (b’0011’). Connecting each switch’s common to an output pin allows the switch to be effectively removed from the circuit whenever a logic high is placed on the common. This way the three devices share the same data bus. Care should be taken to only enable one device at a time or else active interaction will result in bogus data as well as potential device damage.

**TAKING YOUR INPUT**

I will be treating the two user input switches as the number of tenths of a volt between 0 and 100 tenths [100 tenths = 10 V]. Selection is a byte variable whose value is 10 times the upper digit switch reading plus the lower digit switch reading. While storing the value as two BCD digits might make more sense, I want this value to be in the same format as the A/D values. (You’ll see what I mean shortly.) In the configuration used, a switch’s position is read by writing a low to its common pin, and reading this effect on its binary pins through the data bus on the lower nibble of PORTB. The value read from PORTB
is complimented and then ANDeD with 0x0F to mask out the upper nibble. While the upper limit (10 V or 100) is just within reach, you know that the lowest output of the regulator is 1.2 V. So, unless you have an LDR that can reach 0 Ω with unlimited power dissipation, the lower values (say, 0 to 30) will be illegal. (I'll let the LCD tell that story.)

Before I explain the display routine for the LCD, I'd like to take a short detour here and interject the A/D calculations as they directly relate to the aforementioned user input switch settings. Having the single byte value SELECTION (from the BCD switch) and OUTVOLTS (from the A/D conversion and calculation) make it easy to determine what must be done. They should be identical for this application.

The status (supply voltage and regulated voltage) can actually be greater than the microcontroller's ADC can handle, so these voltages will be divided (in this case) by three before measuring them. This means that a full-scale conversion will actually mean 15 V and not the ADC's reference of 5 V. While the ADC is capable of 10-bit conversion, I use only the 8 MSBs to keep it simple. With a 5-V reference, each bit will represent 5 V/256, or 0.01953 V. With this application's divider, that would be three times 0.01953 V, or 0.0586 V. I want to eliminate fractional math and only need to show tenths of a volt using a byte variable. Since my design calls for a full-scale input of a maximum of 15 V and I want this to read as a value of 150 (tenths of a volt), I scale everything up by a factor of 256. Therefore, if I multiply the conversion value by 150 and divide it by 256, I get (for full scale) 255 x 150/256 = 38.250/256 = 149 tenths, or 14.9 V. This application requires a simple 8-bit x 8-bit integer multiply for the first part. Choosing a factor of 256 makes all this work out nicely because I need to use only the upper byte of the 16-bit multiply result to divide it by 256!

DISPLAYING RESULTS
There are only two things of importance to this project: the supply voltage and the regulated output. A small three-digit LED display would be adequate. I like using LCDs when possible because I can display more than just numbers. The small NH0-0108, an eight-character single-line LCD has just what I need. If you've been following my column, you know that scrolling long messages on an LCD really removes what at first seems like a limitation of smaller displays.

Besides showing the column and issue number on a power-on screen, I have only two messages to display: "Bat=xx.x" and "Out=xx.x" [see Photo 2]. Well, that's not really true, as there are a couple of other messages I added to indicate that your choice on the BCD input switches are illegal. These messages occur in place of the output voltage message if the output cannot be regulated to the required request.

I chose a display time of 1 s for each of the two messages. The variables Next time for next display change and Seconds [running seconds counter] are compared to determine when to increment the State variable (which determines the message to display) and recalculate a new Next. While this might seem like a bit of overkill for this application, it lends itself well to easily adding additional states for more messaging capabilities.

Each display message consists of a four-character ROM string to LCD routine (i.e., "Bat") to wipe out the previous display's first four characters and leave the cursor positioned for the individual character display of the appropriate value in the format: digit, digit, decimal point, digit. I used an integer divide routine to help turn the byte variables BatVolts and OutVolts into hundreds, tens, and units digits, symbolizing tens, units, and tenths by the placement of a decimal point in the display format. Both the integer multiply and divide routines used here require less than two dozen instructions each.

PROGRAM INTERRUPTED
Two related interrupts are used in this application. The first is a timer, which provides two functions. It is initialized to interrupt an overflow
every 10 ms. The Count variable is used to increment the Seconds variable every 100 interrupts. The timer also acts to control the initiation of A/D conversions. That’s where all the real work goes down. Because the LDR (remember that device) is relatively slow with respect to a microcontroller’s execution speed, it can actually be counterproductive to change the controlling illumination faster than it can respond to the control signal. This timer interrupt can be used to slow down the update rate by delaying A/D conversions.

I tested the LDR/LED module by pulsing the LED with a 1-Hz output. With the LDR connected between VCC and a 1-kΩ resistor to ground, a scope showed that the LDR’s resistance falls in roughly 20 ms with illumination, but it takes 100 ms for the resistance to rise back up from the lack of the light. With the ADC’s present configuration, it can complete a conversion on a new channel every 120 μs. By using the 10-ms timer to begin a pair of conversions they [and the adjustment routine that follows the conversions] won’t over run the response time of the LDR.

By varying the PWM’s pulse width value Percent, you can effectively control the illumination of the LED from full off to full on. Percent’s value is used to set the PWM’s duty cycle and is determined by comparing Selection [user’s switch setting] to OUTVOLTS (the measured output). If Selection = OUTVOLTS, then no change to Percent is necessary. If Selection < OUTVOLTS, then Percent is decremented, which reduces the light intensity, thereby raising the resistance of the LDR, the voltage across it, and the overall regulated output. If Selection > OUTVOLTS, then Percent is incremented, which increases the light intensity, thereby lowering the LDR’s resistance, the voltage across it, and the overall regulated output.

Earlier I mentioned displaying additional messages. The measured output voltage is displayed unless this voltage doesn’t match what the you have “dialed in” on the BCD switches. If there is an attempt to decrement Percent below its minimum value, the PWM will already be full off and the output will have reached its maximum voltage. The alternate message “Choose <” is displayed. If there is an attempt to increment Percent above its maximum value, the PWM will already be full on and the output will have reached its minimum voltage. The alternate message “Choose >” is displayed.

SWITCHING REGULATORS
While I’ve demonstrated this technique on a linear regulator, there are switching regulators that can be substituted. These use the same style of voltage divider network to set the output voltage. They require additional external components and create their own noise source as they rely on switching the supply voltage on and off. However, they do have advantages. They can boost the output voltage above the supply and have a high efficiency at all voltages, unlike linear regulators that make heat (input/output voltage difference times the current supplied). If you are interested, here are a few switchers you might want to check out: Maxim Integrated Products’ MAX5096, Linear Technology’s LT3430, and National Semiconductor’s LM2576 and LM2585.

PORTABLE POWER
I can now keep all of my electronic devices charged while I’m out in the field. If by chance I begin to tax the 12-V gel cell I use to run this project, I’ll need to remember to connect my portable solar cell battery charger during the daylight hours to keep the battery ready to do its job. I can also plug this project into the cigarette lighter—ah, excuse me, auxiliary power outlet—in any vehicle so I can charge on the go. Let’s see now: phone, camera, laptop, camcorder, gel cell, project, and solar charger. That’s it. Is there any room left for a tent?

Jeff Bachiochi (pronounced BAH-key-AH-key) has been writing for Circuit Cellar since 1988. His background includes product design and manufacturing. You can reach him at jeff.bachiochi@imaginethatnow.com or at www.imaginethatnow.com.

SOURCES
PIC16F737 Microcontroller
Microchip Technology, Inc. | www.microchip.com

LM117/217/317 Regulator
National Semiconductor Corp. | www.national.com

NEED-TO-KNOW INFO
Knowledge is power. In the computer applications industry, informed engineers and programmers don’t just survive, they thrive and excel. For more need-to-know information about topics covered in Jeff Bachiochi’s Issue 244 article, the Circuit Cellar editorial staff highly recommends the following content:

Portable Power
A Power Supply for Embedded Applications
by Jason Wu, Kiran Kanukurthy, & David Andersen
Circuit Cellar 193, 2006
This design team built an inductively charged power supply for embedded apps. The portable system provides 100-mA, 3.3-V continuous power. Topics: Power Supply, Portable Power, Inductive Charging

Programmable Power
Build a Simple USB DAC
by Yoshiyasu Takefuji
Circuit Cellar 213, 2008
Yoshiyasu describes the construction of a simple USB DAC around an ATTiny45 and a MAX517. You can use it as a programmable power supply. Topics: Programmable Power, USB, DAC, Protocol Stack

Go to Circuit Cellar’s webshop to find these articles and more: www.cc-webshop.com
Heat Harvester
A Look at Thermal Energy Production

Green is the new black, and the Silicon Wizards are doing their part with chips that do more for less. But maybe there's another answer. Instead of just using less energy, why not use all the energy, and there's a lot of it, that otherwise goes to waste?

"Engine Bay Cuisine" sub-culture (well at least a few Wiki entries and webpages) for those cooks that really want to get under the hood. But unless you're a Julia Child wannabe on a roadtrip, I dare-say most of the heat is wasted.

But it doesn't have to be. In a 2004 presentation titled "The Effects of an Exhaust Thermoelectric Generator of a GM Sierra Pickup Truck," Alexander Kushch, Madhav Karri, Brian Helenbrook, and Clayton J. Richter described a prototype thermoelectric generator design for a standard pickup truck (see Figure 1). Extracting "free" heat energy from the exhaust pipe and coolant, the generator easily delivers upwards of 100 W at freeway speeds. The extra energy translates directly to improved mileage by reducing the alternator load. As fuel prices rise, so does the payoff from the fuel savings, potentially reducing payback time to a matter of months. Even if electric cars render internal combustion moot, don't forget all the surplus heat energy generated by friction. Now we're talking about every motor, shaft, bearing, and gear, virtually anything with moving parts.

There's plenty of heat for the taking, but it's the taking that's the trick. Read on to see why thermal energy harvesting is a truly hot topic.

**MY GENERATION**

MicroFelt is an outfit that's figured out how to use IC-like processes to create arrays of tiny...