Logic Design and Power Switching (p. 43) • An Introduction to Microstrips (p. 64)

www.circuitcellar.com

CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

#223 February 2009

WIRELESS COMMUNICATIONS

A Scalable Wireless Module Control System

New Wireless Gadgets at Your Fingertips

Move Data With the FAT File Format

Processor Modification Explained

Tips for Solar Panel Evaluation

0 74470 75349 0 02>
$5.95 U.S. ($6.95 Canada)

SERIAL TO ETHERNET SOLUTIONS

Simple Ethernet connectivity for serial devices

Works out of the box - no programming is required

Customize to suit any application with low-cost development kit

Features:

10/100 Ethernet

TCP/UDP mode

DHCP/Static IP Support

Data rates up to 921.6kbps

Web-based configuration

Device P/N: SB70-100CR
Kit P/N: NNDK-SB70-KIT

SB70

$69 *Qty. 100*

2-port serial-to-Ethernet server with TTL support

Device P/N: SB72-EX-100CR
Kit P/N: NNDK-SB72EX-KIT

SB72EX

$139 *Qty. 100*

2-port serial-to-Ethernet server with RS-232 & RS-485/422 support

Device P/N: PK70EX-232CR
Kit P/N: NNDK-PK70EX232-KIT

PK70EX232

$269 *Qty. 100*

4-port serial-to-Ethernet server with RS-232 support

Need a custom solution?

NetBurner **Serial to Ethernet Development Kits** are available to customize any aspect of operation including web pages, data filtering, or custom network applications. All kits include platform hardware, ANSI C/C++ compiler, TCP/IP stack, web server, e-mail protocols, RTOS, flash file system, Eclipse IDE, debugger, cables and power supply.

NetBurner
Networking in One Day!

Information and Sales | sales@netburner.com
Web | www.netburner.com
Telephone | 1-800-695-6828

freescale™
Alliance Member

5 Competitive Advantages

Overseas Manufacturing

Imagineering, Inc. enjoys the reputation of being one of the most experienced & successful offshore PCB suppliers.

CAM USA

Our Illinois based DFM office has eight fully staffed CAD / CAM stations. Within hours of receipt of new Gerber files, our highly experienced DFM engineers conduct thorough and precise analyses.

Quick-Turn Production

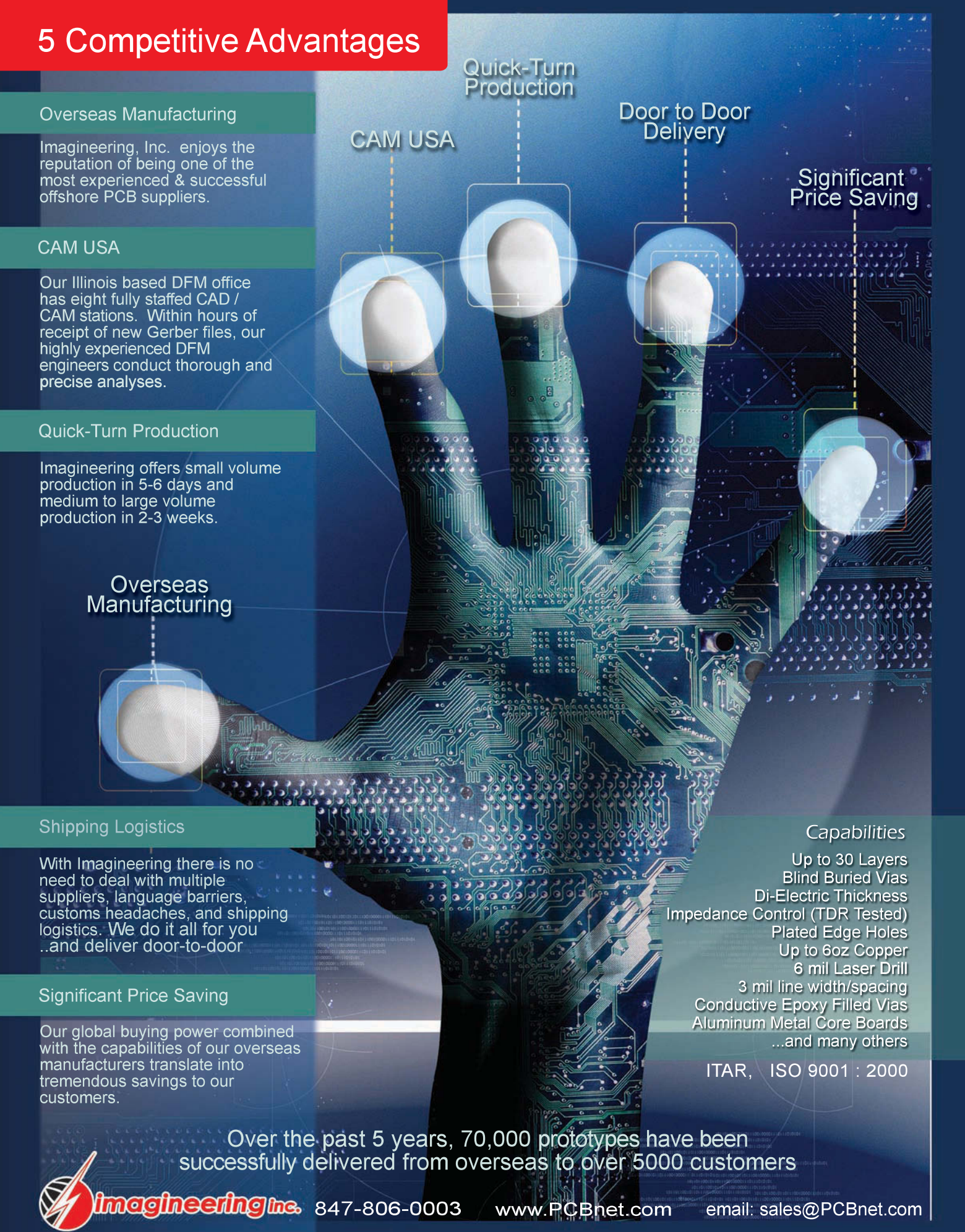
Imagineering offers small volume production in 5-6 days and medium to large volume production in 2-3 weeks.

Shipping Logistics

With Imagineering there is no need to deal with multiple suppliers, language barriers, customs headaches, and shipping logistics. We do it all for you ..and deliver door-to-door

Significant Price Saving

Our global buying power combined with the capabilities of our overseas manufacturers translate into tremendous savings to our customers.



Capabilities

Up to 30 Layers
Blind Buried Vias
Di-Electric Thickness
Impedance Control (TDR Tested)
Plated Edge Holes
Up to 6oz Copper
6 mil Laser Drill
3 mil line width/spacing
Conductive Epoxy Filled Vias
Aluminum Metal Core Boards
...and many others

ITAR, ISO 9001 : 2000

Over the past 5 years, 70,000 prototypes have been successfully delivered from overseas to over 5000 customers

imagineering inc. 847-806-0003 www.PCBnet.com email: sales@PCBnet.com

23 YEARS IN BUSINESS...AND STILL GOING STRONG

Experience a Faster Way to Search for Components!

Jameco Electronics' new catalog and enhanced Jameco.com website are two tools that are designed to work together to give electronic professionals faster access to the hottest components in the industry.

Color coded references throughout the catalog assist you in analyzing a wide range of brand choices (from franchise sourced to house brands to factory overruns) offering you more pricing options than you'll see anywhere else. Web codes allow you to quickly jump from catalog to website to view additional specifications, application notes, photos and more products. You'll find exactly what you're looking for without having to wade through hundreds of thousands of products on a complicated website or wielding a twenty pound catalog.

With a flip of the page or a click of the mouse, you have all the tools you need at your fingertips. Reach for Jameco first and order your catalog today.

Order your web-enabled catalog today!

JAMECO®
ELECTRONICS

Call 1-800-831-4242 or Visit www.Jameco.com/Speed for the ride of your life!

With compilers that cut power consumption

Your MCU can go green!

HI-TECH C® PRO
ANSI C Compilers

featuring Omniscient Code Generation™ (OCG)

OCG is a new whole-program compilation technology that cuts power consumption by reducing interrupt latency and increasing speed while simultaneously reducing code size. It's a win-win for your project and the environment.

HI-TECH C PRO compilers: available for 8051, Cypress® PSoC® Programmable System-on-Chip, and Microchip PIC10/12/16/18/32 MCUs.

Denser code, better performance: www.htsoft.com/ocg

HI-TECH SOFTWARE

OCG

www.htsoft.com • sales@htsoft.com • +1 800 735 5715

TASK MANAGER

PV Technology, Microstrips, and More

In the February 2008 issue, we ran the last article in Steve Ciarcia's three-part series about the design and installation of his photovoltaic (PV) system. That series marked an important point in this magazine's history. We made it clear that we planned to focus on, and cultivate articles about, intelligent embedded technology solutions to energy-related problems. We challenged you to use the technologies introduced by *Circuit Cellar* authors and advertisers in your own embedded systems, whether that meant designing an energy-efficient, stand-alone system or developing MCU-based components to be used in larger "green" systems. This winter, we continue in this vein with Ed Nisley's article, "Solar Measurements" (p. 24). If you plan to design a PV system, large or small, you should first make an effort to understand Ed's techniques for evaluating panels. He focuses on the topics of PV technology, instrumentation, and circuit modeling.

Many discussions about "energy" or "green" topics end up at the important subject of water—the scarcity of water, water conservation systems, and the positive and negative effects of various forms of water usage. One thing is for sure: designing a system to monitor and control an area's irrigation system is an important endeavor. In "Wireless Module Control," Tom Kibalo describes his recently updated microcontroller-based irrigation control system (p. 14).

As usual, we also present articles about several other interesting subjects. On page 30, Larry Standage introduces the topic of processor modification. You can use the flexibility of soft-core processors to your advantage.

The second installment in Steve Hendrix's series about power switching begins on page 43. This month, he describes practical solid-state power switching arrangements.

For an introduction to some exciting new wireless technologies, turn to Tom Cantrell's article on page 50. These wireless gadgets—or *wadgets*, as Tom calls them—are sure to work well in your next embedded communications project.

On page 57, Jeff Bachiochi continues his series about using SD memory cards with embedded applications. He covers everything from implementing the FAT file format to move data to adding new commands and using ring buffers.

Do you know how microstrips are used in PBC designs? In "Microstrip Techniques," Robert Lacoste introduces the topic of microstrips and describes microstripping techniques that you will find useful for many of your upcoming design projects (p. 64).

cj@circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

MANAGING EDITOR
C. J. Abate

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Jeff Bachiochi
Ingo Cyliax
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Gary Bodley
Ken Davidson
David Tweed

ASSOCIATE EDITOR
Jesse Smolin

CHIEF FINANCIAL OFFICER
Jeannette Ciarcia

MEDIA CONSULTANT
Dan Rodrigues

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNERS
Grace Chen
Carey Penney

STAFF ENGINEER
John Gorsky

ADVERTISING

860.875.2199 • Fax: 860.871.0411 • www.circuitcellar.com/advertise

PUBLISHER
Sean Donnelly
Direct: 860.872.3064, Cell: 860.930.4326, E-mail: sean@circuitcellar.com

ADVERTISING REPRESENTATIVE
Shannon Barraclough
Direct: 860.872.3064, E-mail: shannon@circuitcellar.com

ADVERTISING COORDINATOR
Valerie Luster
E-mail: val@circuitcellar.com

Cover photography by Chris Rakoczy—Rakoczy Photography
www.rakoczyphoto.com

PRINTED IN THE UNITED STATES

CONTACTS

SUBSCRIPTIONS
Information: www.circuitcellar.com/subscribe, E-mail: subscribe@circuitcellar.com
Subscribe: 800.269.6301, www.circuitcellar.com/subscribe, Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650
Address Changes/Problems: E-mail: subscribe@circuitcellar.com

GENERAL INFORMATION
860.875.2199, Fax: 860.871.0411, E-mail: info@circuitcellar.com
Editorial Office: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: editor@circuitcellar.com
New Products: New Products, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: newproducts@circuitcellar.com

AUTHORIZED REPRINTS INFORMATION
860.875.2199, E-mail: reprints@circuitcellar.com

AUTHORS
Authors' e-mail addresses (when available) are included at the end of each article.

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Vernon, CT 06066. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $23.95, Canada/Mexico $34.95, all other countries $49.95. Two-year (24 issues) subscription rate USA and possessions $43.95, Canada/Mexico $59.95, all other countries $85.** All subscription orders payable in U.S. funds only via Visa, MasterCard, international postal money order, or check drawn on U.S. bank. **Direct subscription orders and subscription-related questions to Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650 or call 800.269.6301.**

Postmaster: Send address changes to Circuit Cellar, Circulation Dept., P.O. Box 5650, Hanover, NH 03755 5650.

Capacitive Touch Sensing In a Flexible, Single-chip Solution

Capacitive touch interfaces provide an excellent way to add low-cost, reliable and stylish buttons into your design. Microchip Technology's mTouch™ Sensing Solution includes comprehensive development kits and a free diagnostic tool to make implementation easy and fast. Our free source code can be seamlessly integrated with your existing firmware on a single PIC® microcontroller – eliminating the need for additional controllers.

THE mTouch SENSING SOLUTION FEATURES:

- FREE license libraries and source code
- A FREE diagnostic tool
- Integration with 8- and 16-bit PIC microcontrollers
- Easy expansion, with support from 6 to 100 pins
- Low-power operation

GET STARTED IN 3 EASY STEPS

1. Visit the mTouch Sensing Solutions design center at www.microchip.com/mTouch
2. Download FREE libraries and source code
3. For a limited time, save 20% off a variety of touch sensing development tools when you purchase from www.microchipDIRECT.com and use coupon code **mTouch4U**.

Microcontrollers

Digital Signal Controllers

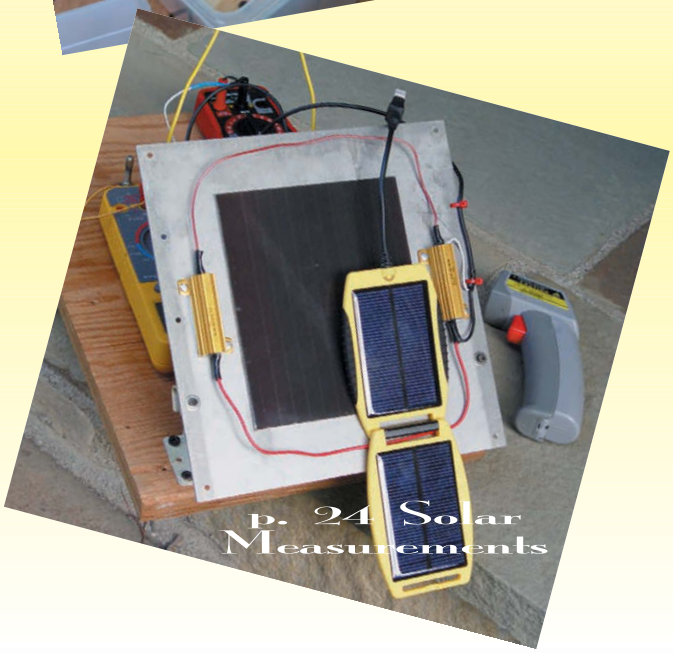
Analog

Serial EEPROMS

MICROCHIP

INSIDE ISSUE 223

February 2009 • Wireless Communications



Hammer Down Your Power Consumption with picoPower™!

THE Performance Choice of Lowest-Power Microcontrollers

Performance and power consumption have always been key elements in the development of AVR® microcontrollers. Today's increasing use of battery and signal line powered applications makes power consumption criteria more important than ever. To meet the tough requirements of modern microcontrollers, Atmel® has combined more than ten years of low power research and development into picoPower technology.

picoPower enables tinyAVR®, megaAVR® and XMEGA™ microcontrollers to achieve the industry's lowest power consumption. Why be satisfied with microamps when you can have nanoamps? With Atmel MCUs today's embedded designers get systems using a mere 650 nA running a real-time clock (RTC) and only 100 nA in sleep mode. Combined with several other innovative techniques, picoPower microcontrollers help you reduce your applications power consumption without compromising system performance!

Visit our website to learn how picoPower can help you hammer down the power consumption of your next designs. PLUS, get a chance to apply for **a free AVR design kit!**

http://www.atmel.com/picopower/

Everywhere You Are®

RF QUADRATURE MODULATOR

The new **CMX993** is a highly integrated, general-purpose, RF quadrature modulator, offering 100 MHz to 1 GHz operation and excellent wideband noise performance. Additional features include gain control and uncommitted differential amplifiers. Targeted to meet the challenging requirements of wireless data, the product's I/Q architecture supports a wide range of modulation types. Various integrated, selectable functions are offered to maintain performance across multiple modulations and bandwidths.

The quadrature modulator provides translation from baseband I/Q signals to a modulated RF signal. The wideband inputs can be driven single-ended or differentially for optimum performance. The device offers two integrated and matched double-balanced mixers, driven from a buffered quadrature-split local oscillator. The "LO" frequency is divided by either two or four, with the mixers forming an I/Q vector modulator with programmable gain stages, offering 30 dB of gain control in 2.5-dB steps.

A digital control interface, C-BUS (SPI-compatible interface), allows gain control, as well as the power management of individual internal blocks for optimum system performance. The C-BUS interface operates from its own power supply, enabling the CMX993 to be interfaced to baseband devices of differing voltages.

The CMX993 operates at 3.3 V (1.8 V I/O) and comes in a 48-pin VQFN package. It costs **$5** in 1,000-piece quantities. An evaluation kit, the EV9930, costs **$87**.

CML Microcircuits
www.cmlmicro.com

ULTRA-COMPACT, LOW-COST NETWORKING MODULE FAMILY

The **MiniCore** is a series of easy-to-use, ultra-compact, low-profile, low-cost networking modules. The MiniCore series is optimized for real-time control, communications, and networking applications, such as energy management and intelligent building automation. Smaller than a packet of sugar, it offers you extreme flexibility to easily place wired or wireless network connectivity anywhere on a motherboard. It also features 32 GPIO and a Rabbit Semiconductor 5000 microprocessor.

Only 0.1" high, the MiniCore family provides a rich embedded feature set on an ultra-compact mini PCI Express form factor. The family includes the pin-compatible and interchangeable wired RCM5700 and the soon-to-be-released Wi-Fi RCM5600W. Pin-compatible versions that support ZigBee and USB are also in development. Systems developers can easily interchange MiniCore family products to tailor connectivity for any given application. The RCM5700 features a 50-MHz Rabbit 5000 microprocessor, on-chip 10/100 Ethernet, 1 MB of serial flash memory, 36 digital I/O ports, and five serial ports.

Low-cost development kits are available featuring all of the software and tools necessary to make development easy. The RCM5700 development kit includes the RCM5700 module, an interface board with standoffs and connectors, a prototyping board with standoffs and connectors, a USB cable, and a Dynamic C CD-ROM with complete development documentation for a promotional price of **$49 ($99 MSRP)**. Deluxe kits, including an additional universal AC adapter, digital I/O, and serial communication accessory boards with sample programs and CAT 5/6 Ethernet cable are available at a promotional price of **$99 ($199 MSRP)**. The RCM5700 costs **$24.99** in 2,500-plus volume orders.

Rabbit Semiconductor, Inc.
www.rabbit.com

NEW PRODUCT NEWS

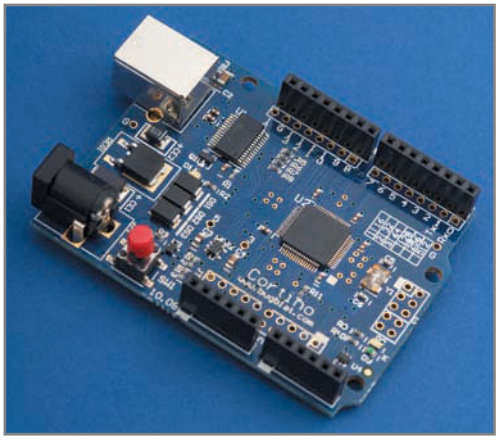
Edited by John Gorsky

DEVELOPMENT SYSTEM FOR ARM CORTEX-M

The Cortino is an entry-level development platform for the ARM Cortex-M processor in the popular Arduino format and pinout. The Cortino is USB powered and controlled, so all you need to get started on ARM Cortex development is a Cortino, a USB cable, and a compiler.

The USB hookup is naturally quick—the Cortex processor drives the link at 3 MBps and it has crucial advantages for a development system. You can control the reset mode of the Cortex, rewrite its flash memory, and drive its serial debug port.

The on-board MCU is an STM32 32-bit ARM Cortex with 32 to 512 KB of program flash memory, 6 to 64 KB of RAM, up to five USARTs, three SPIs, two I^2Cs, six 16-bit timer/counters, three multi-input 12-bit ADCs, one two-channel 12-bit DAC, a multi-channel DMA controller, two watchdogs, 5-V tolerant I/O, and up to 72-MHz operation.



The Arduino format adds multiple expansion headers with 3.3 and 5 V available, extends the built-in USB debug interface with an ARM 2 × 10 pin JTAG connector, and adds flexible reset circuitry and programmable LEDs. The Cortino will operate in both stand-alone mode and as a USB peripheral.

The Cortino starts at $39.99.

Bugblat
www.bugblat.com

LEVEL-TRANSLATING ADC DRIVER

The AD8275 is a level-translating ADC driver that simplifies signal conditioning for high-voltage industrial and instrumentation designs.

You can now use a single component to attenuate, level-shift, and drive low-voltage ADCs in high-voltage applications, reducing part count and board space by simplifying the signal conditioning complexities that often occur in data acquisition, data logging, and other high-performance industrial and instrumentation equipment. By translating ±10-V signals to 0- to 4-V input levels, the 15-MHz AD8275 is especially suited for driving higher-speed 16-bit SAR converters.

The AD8275 0.2 gain-difference amplifier includes internal, matched, precision laser-trimmed resistors that achieve 0.002% gain error and 1 ppm/°C (max) gain drift. Delivering low distortion and a fast 450-ns settling time to 0.001%, the amplifier is ideal for system engineers designing applications where quick, accurate data capture is required. In addition to eliminating high-voltage power supplies, discrete resistor dividers, driver amplifiers, and other signal conditioning circuitry previously required to drive SAR ADCs, the AD8275 saves space because of its small footprint.

The AD8275 features a wide input voltage range of –35 to 40 V and rail-to-rail outputs, making an easy-to-use building block. Single-supply operation reduces the power consumption of the amplifier and helps protect the ADC from overdrive conditions. The AD8275 can be used as an analog front end, or it can follow buffers to level-translate high voltages to a voltage range accepted by the ADC.

The AD8275 costs $1.60 each in 1,000-piece quantities.

Analog Devices, Inc.
www.analog.com

NPN

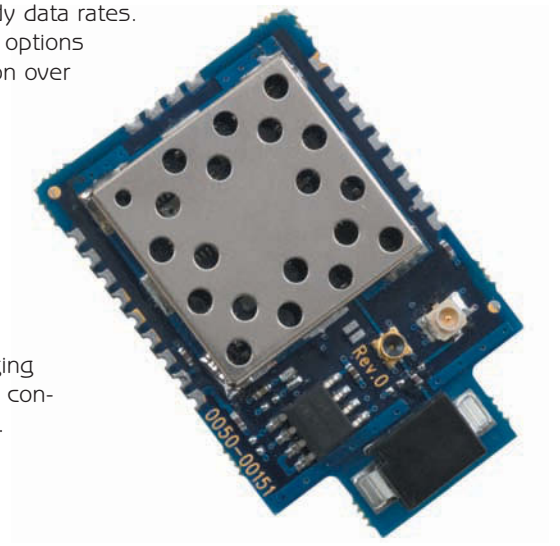
RF MODULES PUSH THE LIMITS

The LT2510 is a fifth-generation 2.4-GHz FHSS module that sets the standard for industrial RF communication. Based on its established proprietary FlexRF technology and operating in the globally available 2.4-GHz spectrum, the LT2510 is optimized to outperform conventional wireless standards.

Embedded with a robust server-client protocol, the LT2510 permits each module to communicate with any other in-range module for true peer-to-peer operation. Out-of-range modules can be reached via a meshing topology. The configuration and test software enables OEMs to structure and optimize networks to suit their application.

Enhanced API commands provide packet-routing control and network intelligence. With its field-proven FHSS air interface protocol, the LT2510 rejects RF noise, excels against multipath scenarios, allows for colocated systems, and provides an extremely reliable communication link.

With over 150 kbps throughput in half-duplex mode, the LT2510 delivers speedy data rates. In addition, variable output power options (up to 1 W) enable communication over distances that are not achievable with competing technologies. A range of ultra-low-power modes, plus low TX/RX power consumption, make the LT2510 ideal for power-restrictive or battery-operated applications. The SMT package is well-suited for space-constrained designs and is available in pick-and-place packaging for volume manufacturing. Please contact Laird Technologies for pricing.



Laird Technologies, Inc.
www.lairdtech.com

UNDER-BOARD HEATER IS PERFECT FOR PCB REPAIRS

The WHP200 is an under-board heater designed to speed up repair of small multilayer PCBs. It is simple to operate and offers easy temperature adjustment.

An infrared plate provides fast, precisely controlled heating, ensuring an even substrate temperature. The set temperature and actual temperature are digitally displayed for easy process control. A PCB board holder is also included, so the WHP200 is ready to work, right out of the box.

Although small in size, the WHP200 is powerful. It's perfect for repairing the PCBs from a wide range of today's consumer products, including digital cameras, mobile phones, MP3 players, and PDAs.

The WHP200 costs $670.

Cooper Tools
www.coopertools.com

NPN

SHARC & Blackfin PROGRAMMER

The dspFlash Sharc & Blackfin Programmer is a production programmer that allows for on-board flash memory programming for flash memory connected to Analog Devices's Sharc and Blackfin DSPs. The dspFlash programmer uses a 12-MHz JTAG connection to obtain programming speeds of over 400 kbps for parallel flash memories and over 50 kbps for serial flash memories. It is capable of programming all standard ADI ldr formats, so all you need is a standard ADI JTAG connector on your target.

The dspFlash deluxe further expands the functionality of the dspFlash by increasing the internal memory to 128 Mb and adding support for programming OTP memory on the newest ADI processors. It can also run on-board diagnostic programs, which can be generated by you, allowing for full customization.

The dspFlash operates in conjunction with a stand-alone Windows program that is included with the hardware. Two replaceable ribbon cables that plug into the latching header are included. The software runs under Windows 2000, Windows XP, or Vista with at least 256 MB of RAM.

The dspFlash costs $500. The dspFlash deluxe costs $750.

Danville Signal Processing, Inc.
www.danvillesignal.com

STELLARIS BRUSHED DC MOTOR CONTROLLER MODULE AND RDK

The Brushed DC Motor Controller Module (MDL-BDC) and Reference Design Kit (RDK-BDC) are now available worldwide. The MDL-BDC provides variable speed control for 12-V brush DC motors at up to 40-A continuous current. It provides two options for motor speed control, an industry-standard R-C servo type (PWM) interface, and a CAN interface. The module also features a status LED that indicates run, direction, and fault conditions, along with a rich set of control options and sensor interfaces, including a motor brake/coast selector, forward and reverse limit switch inputs, analog inputs, and quadrature encoder interfaces. The module uses highly optimized motion control software and a powerful 32-bit Stellaris LM352616 microcontroller to implement multiple motion-control algorithms including open-loop speed control, as well as closed-loop control of speed, position, or motor current. The module is securely enclosed in industrial plastic and is cooled by a self-contained, temperature-controlled fan.

Packaged to demonstrate the simultaneous motor control and connectivity capabilities of the MDL-BDC module, the RDK-BDC reference design kit ships with a Mabuchi Motor 12-V brush DC motor and a separate CAN console, based on the EK-LM352965 CAN evaluation board, which demonstrates the ability to control and monitor the MDL-BDC in real time over the integrated CAN network. The RDK-BDC also includes all of the cables needed to connect the components in the kit, a universal input wall power supply, a JTAG interface cable for debugging user code modifications to the MDL-BDC module, and a CD containing all of the documentation and design information.

The MDL-BDC costs $109 and the RDK-BDC costs $219.

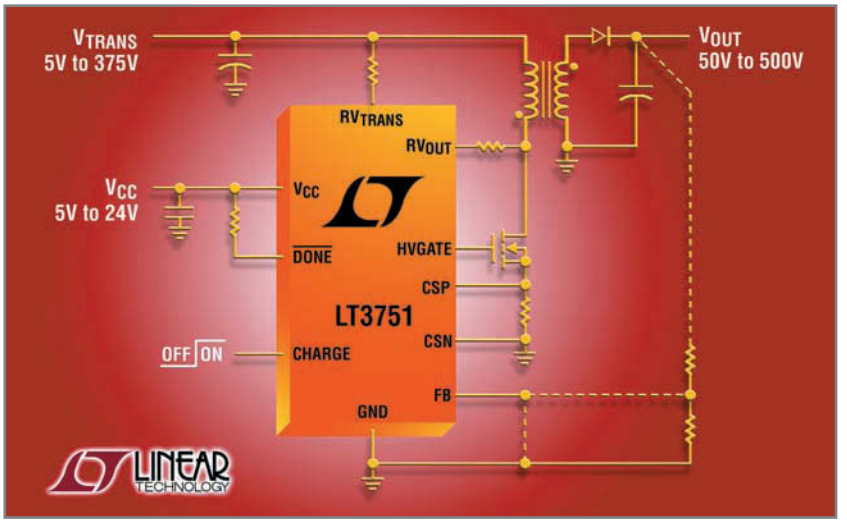
Luminary Micro, Inc.
www.luminarymmicro.com

NPN

FLYBACK CONTROLLER CHARGES ANY SIZE CAPACITOR

The LT3751 flyback controller is designed to rapidly charge large capacitors to voltages as high as 1,000 V. The LT3751 drives an external N-Channel MOSFET and can charge a 1,000-µF capacitor to 500 V in less than 1 s, making it ideal for applications such as professional photoflash systems, RF security, inventory control systems, and specialized high-voltage power supplies. The LT3751 can be configured for primary-side output voltage sensing without needing an optocoupler. For lower noise and tighter regulation applications, a resistor divider network from output voltage can be used to regulate the output, making it well-suited for high-voltage power supply requirements.

Boundary mode control minimizes transition losses and delivers up to 88% efficiency. The V_{CC} input voltage ranges from 5 to 24 V, enabling it to operate from a variety of power sources and the V_{TRANS} supply voltage range is dependent on the choice of external components with the LT3751's internal shunt regulator. A low 106-mV differential current sense threshold accurately limits its peak switch current. Additional features include an on-board selectable 5.6- or 10.5-V powerful 2-A N-channel MOSFET gate driver, a CHARGE pin that enables you to initiate a new charge cycle and remote shutdown, along with undervoltage and overvoltage lockout on the V_{TRANS} and V_{CC} input voltages. The DONE pin indicates when the capacitor has reached its programmed voltage and has stopped charging.



The LT3751 comes in extended and industrial temperature ranges and starts at $3.65 in 1,000-piece quantities.

Linear Technology Corp.
www.linear.com

NPN

TEMPERATURE SENSOR WITH INTEGRATED EEPROM

The new **CAT34TS02** combines a 12-bit (plus sign bit) digital output temperature sensor with 2 Kb of serial presence detect (SPD) EEPROM for DDR3 applications in high-speed PCs and laptops, graphics cards, servers, telecom equipment and base stations, environmental control systems, and industrial process control equipment.

The CAT34TS02 is compliant to JEDEC specification JC42.4, providing ±1°C temperature sensing accuracy from 75° to 95°C and ±3°C accuracy from 20° to 100°C, across the device supply voltage range of 3 to 3.6 V. It measures and records the system temperature approximately 10 times per second, with readings compared to three trigger limits stored in the internal memory registers. Readings can be retrieved by the host system via the I²C/SMBus interface and over- and under-limit conditions are signaled on the open-drain EVENT pin.

For general-purpose temperature sensor applications, integrating the temperature sensor with nonvolatile EEPROM eliminates the need for external memory to store sensor and threshold configurations.

Additionally, the CAT34TS02 nonvolatile memory retains custom-configured sensor and threshold values at power-up resulting in a simple, space-saving, and cost-effective solution.

The integrated 2-Kb SPD EEPROM is internally organized as 16 pages of 16 bytes each, for a total of 256 bytes. The EEPROM features a 16-byte page write buffer, supports both the Standard (100 kHz) and Fast (400 kHz) I²C protocols, and features permanent and reversible software write protection, defined as standard for all DDR3 DIMM modules.

The CAT34TS02 temperature sensor with EEPROM costs **$0.75** each in 10,000-unit quantities.

ON Semiconductor
www.onsemi.com

NPN

by Tom Kibalo

Wireless Module Control

An MCU-Based Irrigation Control System

Tom describes a PIC-based prototype design for a wireless water control system for EarthBoxes, which are gardening boxes with water reservoirs and fertilizer strips for cultivating plants and vegetables. The scalable design can accommodate various multiple-box configurations.

In this article, I will describe a prototype for a wireless water control system for EarthBoxes (see Photo 1). The design is scalable and can accommodate small to large quantities of EarthBoxes.

I first described EarthBoxes in "Java-Based EarthBox Watering System" (*Circuit Cellar* 201, 2007). EarthBoxes are advanced gardening boxes with self-contained water reservoirs and fertilizer strips that create a near-perfect environment for growing plants and vegetables in a compact 2.5′ long by 15″ wide growing area (see Figure 1). Because of its compact, self-contained configuration, the EarthBox enables productive gardening in small spaces.

My family and I are strong proponents of the EarthBox. We've used it successfully for several years now to grow vegetables. As a side note, because of its productivity and low maintenance, the EarthBox is beginning to make serious headway as an alternative in some large commercial-scale farming activities.

When using EarthBoxes, you must make sure the internal water reservoirs are replenished on a daily basis, especially during the height of the growing season. A manual watering regimen could be my "back to nature" mantra; however, I get a lot more satisfaction using an automation process where microcontrollers (especially ones that I have designed) do the work. My articles on this subject document my attempts to

Photo 1—This shows all of the major components of the watering system. There are two sites in the current design: Site A and Site B. (This is indicated on the graphics display.)

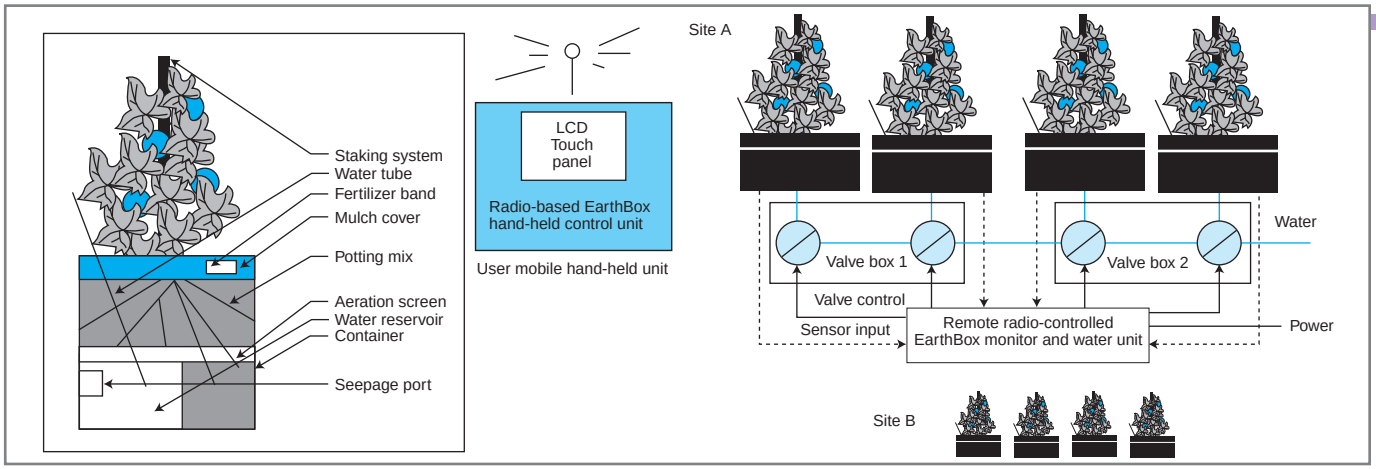


Figure 1—This is a layout for the EarthBox and control system.

automate that process. In one sense, the design I cover in this article can be viewed as "second generation"—that is, it is an updated version of the first design that features new technologies.

IMPLEMENTATION OVERVIEW

A representation of this system is shown in Figure 1. This configuration should not be conscribed as a design limitation, rather it is configured for my gardening needs. The design covers two sites that are on the south side of a large deck in my back yard. Each site is separated by about 50′ to 70′ to allow for gardening maneuverability and harvesting. The system, as implemented, can water eight EarthBoxes. I found that six to eight EarthBoxes can easily meet the fresh vegetable needs of my immediate family (six people).

Let's review the major components. The first is the portable hand-held control unit (on the left-hand side). It is used to remotely determine the level of the water reservoir within each EarthBox and to initiate, under your control, a watering action for that EarthBox, if necessary. Another component is the remote-controlled monitor and water unit. There is one for every four EarthBoxes. This component is controlled by the hand-held unit and continuously monitors the water level for each of its associated EarthBoxes. It also controls the water valves, which are used to supply water to the reservoirs in the EarthBoxes. The final component is the valve box. The boxes are a useful way to organize distributed valves using a manifold system to help minimize water hose connections. Both the hand-held and remote units are Microchip Technology PIC24F-based microcontroller systems. Communication is accomplished with MaxStream (Digi International) 802.15 ZigBee modules.

The hand-held control unit uses a Multilabs 160 × 80 monochrome touchscreen LCD. This approach offers a flexible way of dealing with the operator interface with virtual touch buttons and graphic and text displays. The product is easy to configure (via a vendor command set over a TTL serial port) and reasonably priced. Newer versions have greater pixel capability and support controlled backlighting. Photo 2 shows the user interface constructed for the present design. One of my design goals was simple representation and straightforward operation. I wanted a novice user, with some reasonable notion of the overall system configuration, to be able to effectively operate the system. In other words, I did not want the watering responsibility all the time!

The main screen (defaults at power-up) enables you to select the site of interest, determine the reservoir water status, and initiate watering, if necessary. Once the site is selected, the handheld presents the site's EarthBox water level and valve activation status. The status can be refreshed at any

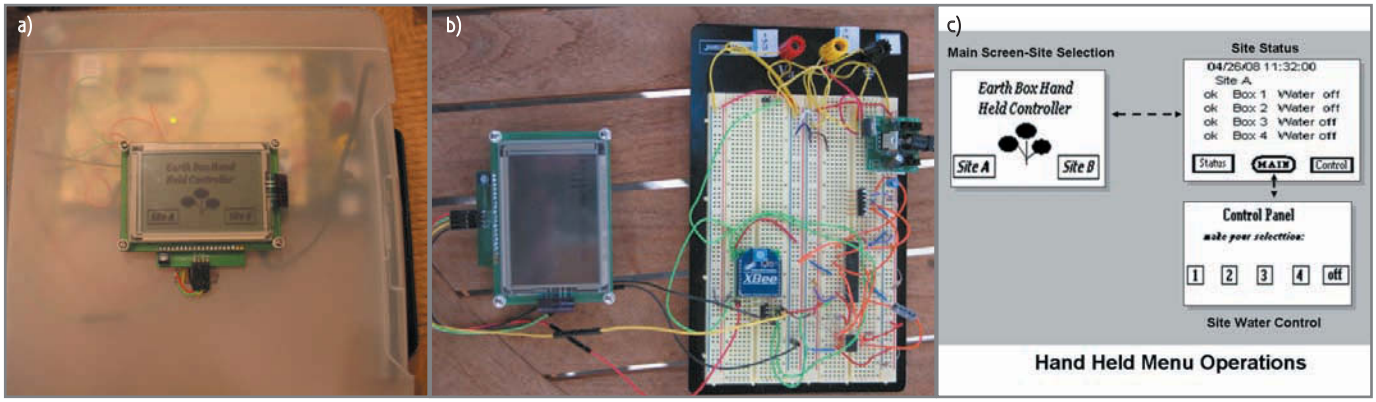


Photo 2—Take a look at a composite hand-held assembled in a box (**a**), a breadboard (**b**), and graphic display menu (**c**).

time by pressing the status button. You can then return to the main screen to select another site. To provide audio feedback for button depressions against the touchscreen, a simple 3.3-V buzzer is incorporated. From the status panel, you can enter the control panel by pressing the control button. At the control panel, you can initiate a watering. Here, only one box valve can be activated at a time by selecting the appropriate button or all of the box valves can be turned off simultaneously. Once a box valve button is selected, the control panel automatically exits to the status panel where all status information for that site is updated automatically.

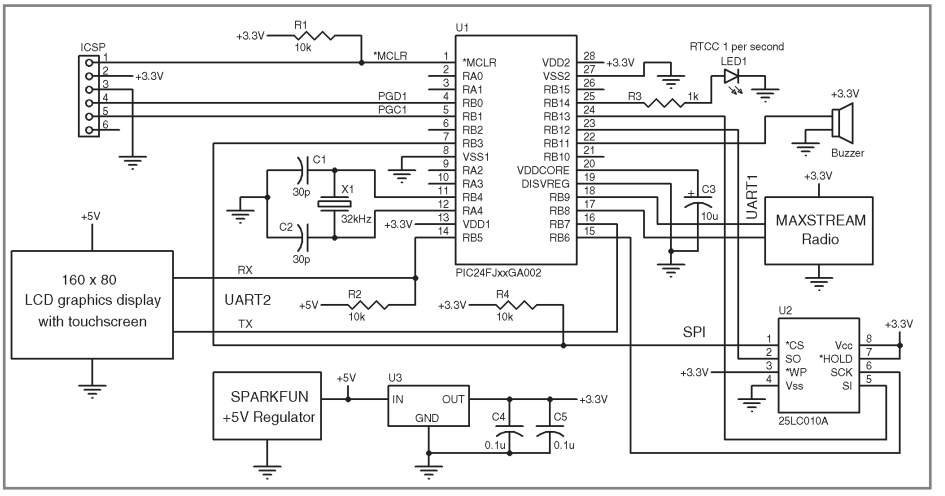


Figure 2—This is the PIC24F64GA002-based hand-held control unit.

MCU & TOOLS

The PIC24F is a 16-bit, general-purpose, 3.3-V based microcontroller. Microchip conveniently offers the chip in a 28-pin skinny DIP package (PIC24F64GA002), which allows for easy assembly in a solderless breadboard setup. The processor has ample program flash memory (64 KB), 8 KB of RAM, and 16-MIPS operation with no need for an external oscillator, low-power operation, and a host of on-chip peripherals. In fact, it contains more internal peripherals than there are pins available. Microchip has gotten around this problem by introducing a peripheral pin select feature, which enables you to map up to 16 (pins RP1 to RP16) of the 28 pins to internal peripherals. The days when designers had to face up to fixed-pin configuration for peripherals are gone. This was a handy feature in this design, where in one case, two UARTs are required for the hand-held unit (one for the touch panel and the other for the MaxStream radio). While in the monitoring and control unit, a single UART is still required for the radio, but all other pins are needed for ADC inputs or relay control. The total available peripheral capability of the PIC24F64GA002 is impressive: 10 channels, 10 bits, 500-kbps ADC, dual UARTS, dual SPI, dual I^2C, dual analog comparators, a parallel master port, five 16-bit timers, five capture modules, five compare/PWM modules, a real-time clock calendar RTCC with an alarm function, as well as just 19 digital I/O. In this design, I hardly scratched the surface but there is good use of the digital I/O, the UARTs, a SPI, ADC channels, and RTCC.

The PIC24F64GA002 supports on-board debug and in-circuit programming using Microchip in-circuit serial programming (ICSP). This interface requires a simple six-pin interface that is easily accommodated with simple six-pin 0.1″ headers on the solderless breadboard. The ICSP appears as a common interface on all of the prototype breadboards for obvious reasons.

Microchip also supplies the MPLAB integrated development environment (IDE) and C30 compiler (the C compiler for this part) for free. (C was used in these designs, but MPLAB also supports assembly.) To complete the picture for debug and programming, Microchip recently introduced a low-cost version

of a real-time debugger/programmer that works with PIC24F microcontrollers. The PICkit2 Debug Express costs less than $50 for this design implementation.

The entire tool set was used. The PICkit2 plugs nicely into the on-board ICSP, which enables quick insertion for program updates and debug without any prototype disassembly. When I assembled the ICSP on the breadboards, I ensured that the PICkit2 could be easily mechanically inserted and removed.

HAND-HELD UNIT

This design is a mixed 5- and 3.3-V design (see Figure 2). The graphics display touchscreen requires 5 V. This voltage is derived from a SparkFun Electronics regulator adapter board. The 3.3 V is derived from a separate 3.3-V three-pin regulator connected to the output of the SparkFun board. Fortunately for 5-V digital I/O, the PIC24F, which is a 3.3-V part, can accommodate this need. Any nonanalog peripheral pin of the PIC24F can be configured as open-drain digital output that with a pull-up resistor to 5 V, can drive 5-V logic without issue. UART2 TX, connected to an LCD, is configured in this matter using C code:

```
__ODB5 =1; //set RB5 to be open drain
```

All input pins on the PIC24F64GA002 are 5-V tolerant. In this matter, the Multilab serial output, a 5-V output, is simply connected to the UART2 RX from the LCD. UART1 TX/RX is connected to the MaxStream ZigBee radio. The radio module uses a 3.3-V serial interface so no additional level shifting is needed. I directly plugged the radio module into the solderless breadboard with a carrier module that has 0.1″ spacing (from SparkFun Electronics). The PIC24F64GA002 real-time clock calendar requires external circuitry, which is accomplished by a simple 32-kHz clock crystal and dual 30-pF capacitors. The clock can be used to activate any needed automatic time-based operations on part of the hand-held application (i.e., polled remote

Listing 1—The code performs a peripheral pin select function for the PIC24F64GA002, configuring the RP pin outside on-chip package to be set to different peripherals contained within the chip. In this case, for inputs, RP9 is configured for UART1 receive. RP13 is configured for SPI port 1 data in. RP7 is configured for UART2 receive. For outputs, RP8 is configured for UART1 transmit. RP5 is configured for UART2 transmit. RP12 is configured for SPI port 1 data out, and RP6 is configured for SPI port 1 clock out.

```
__builtin_write_OSCCONL(OSCCON & 0xbf); // unlock
//input handheld
    RPINR18bits.U1RXR =9;       //make RP9 UART1 RX
    RPINR20bits.SDI1R=13;       //make RP13 SDI1
    RPINR19bits.U2RXR=7;        //make RP7 UART2 RX
//output handheld
    RPOR4bitsRP8R =3;           //make RP8 UART1 TX
    RPOR2bits.RP5R =5;          //make RP5 UART2 TX
    RPOR6bits.RP12R=7;          //make RP12 SDO1
    RPOR3bits.RP6R=8            //make RP6R SCK1
__builtin_write_OSCCONL(OSCCON | 0x40);  //lock
```

Listing 2—The code does a peripheral pin select function for the PIC24F64GA002, configuring the RP pin outside on-chip package to be set to different peripherals contained within the chip. In this case, for inputs, RP9 is configured for UART1 receive and RP7 is configured for SPI port 1 data in. For outputs, RP8 is configured for UART1 transmit. RP5 is configured for SPI port 1 data out. RP6 is configured for SPI port 1 clock out.

```
builtin_write_OSCCONL(OSCCON & 0xbf);// unlock
//input monitor and control
        RPINR18bits.U1RXR =9;         //make RP9 UART1 RX
        RPINR20bits.SDI1R=7;          //make RP7 SDI1
//output monitor and control
        RPOR4bitsRP8R =3;             //make RP8 UART1 TX
        RPOR2bits.RP5R =7;            //make RP5 SDO1
        RPOR3bits.RP6R=8              //make RP6R SCK1
__builtin_write_OSCCONL(OSCCON | 0x40);      //lock
```

status updates or data logging, if necessary). There is also a 1-KB EEPROM interface supported by SPI1. The EEPROM is needed to establish application-specific power-up parameters. Finally, a RadioShack 3.3-V buzzer provides audio feedback for the graphics touchscreen. The list of RP pins and C code to modify them is in Listing 1. Note that a macro code is used to unlock and lock the RP pin configuration using a special-purpose register OSCCON.

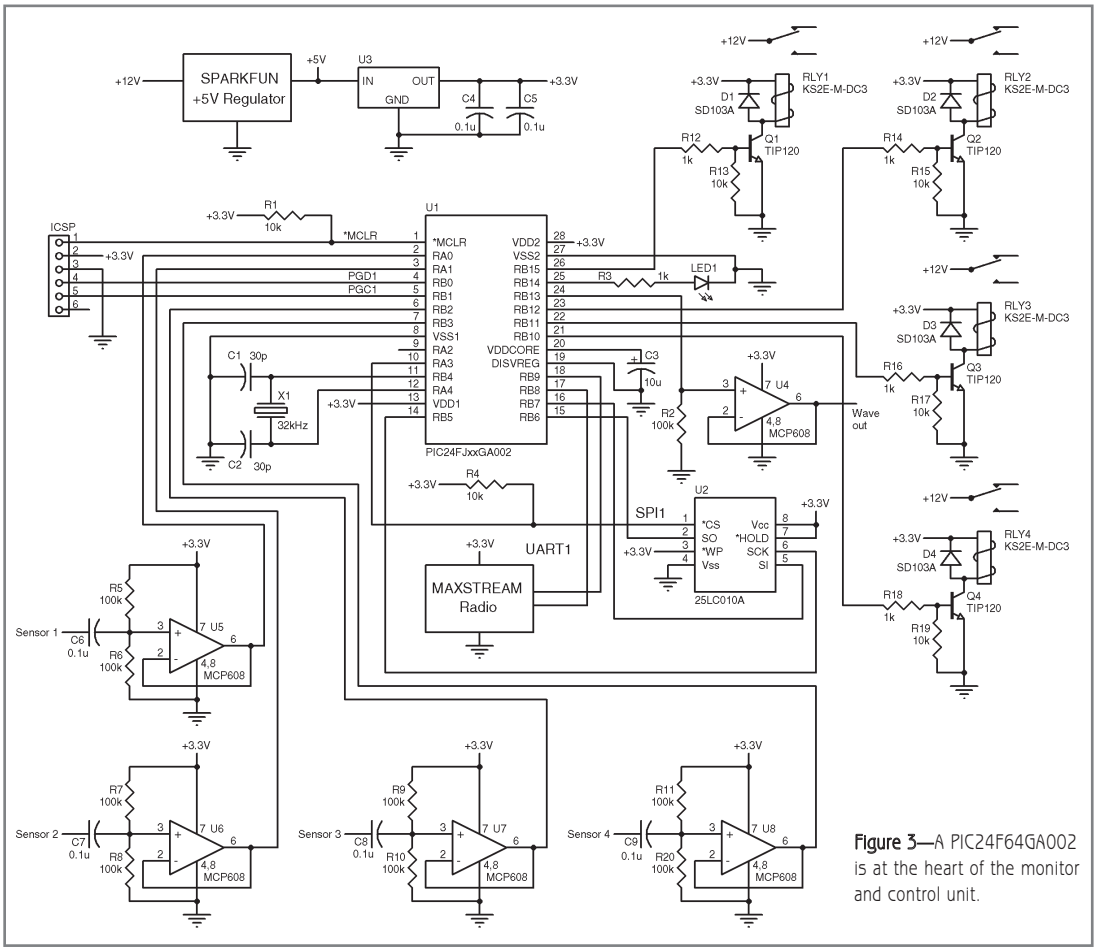


Figure 3—A PIC24F64GA002 is at the heart of the monitor and control unit.

Photo 2b shows the hand-held control unit in a makeshift case. The main screen is active. It also shows the hand-held unit disassembled with major components highlighted. If it is not clear by now, I am keen on minimal assembly efforts and rapid prototyping techniques. In these early stages of design, it is prudent to use techniques that can accommodate quick modification and changes. Once you are through these early prototype validation stages, a more permanent configuration can be employed based on market value and customer needs.

MONITOR AND CONTROL UNIT

This unit is exclusively 3.3 V with the exception of 12 V used in relay-switched control of water valves (see Figure 3). UART1 TX/RX is connected to the MaxStream ZigBee radio, identical to the hand-held. I tried to employ as much commonality as I could across the two designs to maximize hardware and software reuse. I also used the PIC24F real-time clock calendar. Here the clock can activate necessary automatic time-based operations on part of the monitor application (i.e., unsolicited status updates every predefined time). Finally, the 1-KB EEPROM is used, for the same reasons as the hand-held, to establish application-specific power-up parameters.

The use of the ADC across four unique channels for individual monitoring of each sensor input is new. This use of ADC is critical to the "wet test." The Microchip MP608 op-amp is used exclusively for all sensor interfaces. Each MP608 is configured as a unity gain follower. The part is low-power and has sufficient gain-bandwidth product to handle the sensor's 10-kHz input waveform. It also has rail-to-rail operation for maximum output swing. Each MP608 is capacitively coupled to the water medium. The input capacitor is then connected to a resistor bias network to ensure that the incoming level is properly DC-shifted to reduce clipping before ADC conversion. A single MP608 is used to drive the water media from the P24F-generated 10-kHz square wave. Again, a unity gain follower configuration is used. The sensor probes are close to each other in the water medium (less

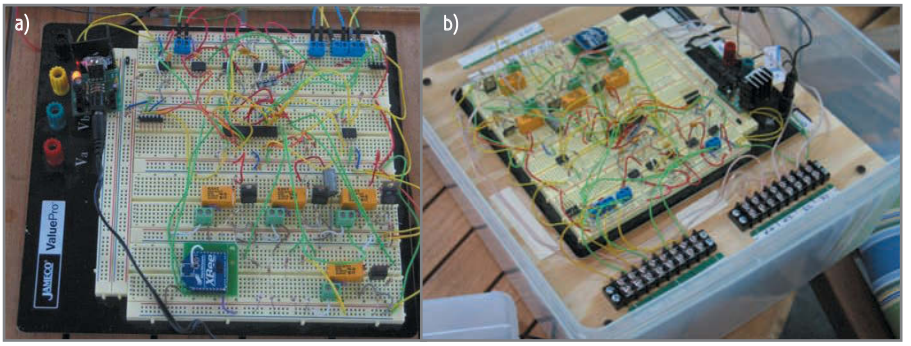


Photo 3—These are composite assemblies of the PIC24F64GA002 monitor and control unit.

than 0.5″ separation), so no gain is required.

The use of PIC24F digital I/O with interface electronics to switch power to the water valves using relays is also new. The relays used are DPDT capable of switching up to 2 A at 24 VDC. The coils can be activated with just 3.3 V. The 12 V is connected to one side of the DPDT wiper of each relay with the normally open side of the relay connected to the power input to the water valve. To drive the relay, each PIC24F digital I/O uses a TP120 Darlington pair transistor with its collector connected to one side of the coil while the other is connected to the 3.3-V supply.

Listing 2 includes the list of RP pins and the actual C code to modify them with the monitor and control unit. Photo 3 shows the disassembled monitor and control breadboard.

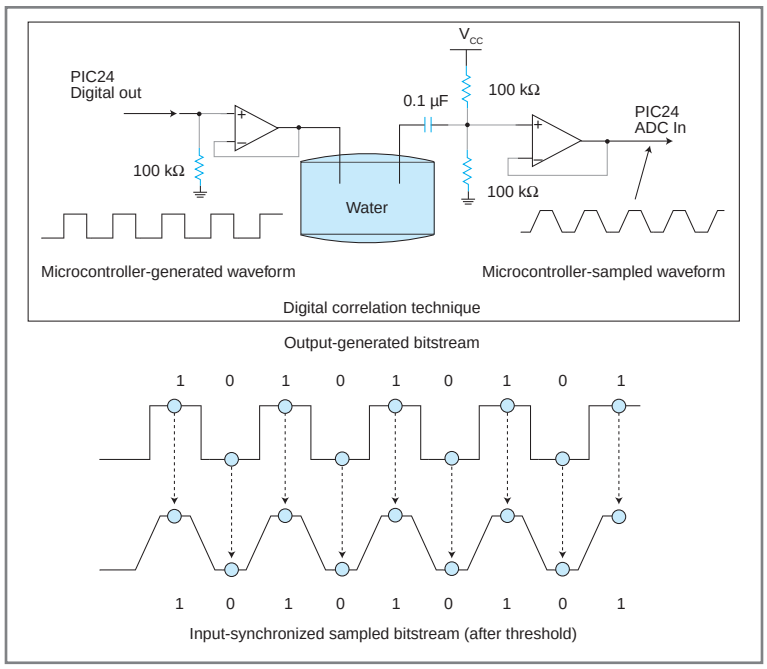


Figure 4—This is a monitor and control wet test diagram showing an example of sensor I/O and a notional correlation scheme.

THE WET TEST

The key to ensuring successful operation is what I call the "wet test." This test is conducted at every sensor in the monitor and control unit to determine if water is present (see Figure 4).

During the wet test, the PIC24F64GA002 continuously generates a 10-kHz 50% duty cycle square wave doing nothing other than bit-banging output pins. The output square wave is then applied to an MCP608 3.3-V, single-supply, full-rail op-amp configured as a voltage follower. The op-amp then sources the signal to all of the reservoirs within the site associated with the monitor unit. On the other side of each reservoir is the sensor pick-up for that signal. A replica of the signal should appear on this side of the reservoir if water (as an electrical conductor) is present. Each pick-up feeds its own MCP608 op-amp that is capacitively coupled to the source and DC-biased to center the mid-voltage point of the original square wave to avoid clipping. The recovered signal, a copy of the original square wave, is somewhat "rounded" in its edges (because of inherent low-pass filtering) and then applied to an ADC channel of the PIC24F64GA002. Here's where the magic happens. The PIC24F64GA002 is used to generate the output waveform level. This level is first stabilized and then ADC sampled. This synchronization with ADC sampling is done in lockstep with all output waveform changes. Because you know the signal at origin and you can synchronically sample it as it crosses the water media, you should know the answer a priori. In essence, you perform a "digital correlation." If there is a failure in comparison between input and output at any time, this constitutes a lack of water between the output source and the input sensor. The only other implementation concern is to mechanically set the source and sensor probes within the water medium at the right level in the reservoir to establish a "full

condition."

A couple of other points need to be made. The input samples through the ADC channel are 10-bit unsigned magnitude values. They need to be threshold set to determine the one or zero representation. The PIC24F can accomplish all of these activities in its 16-MIPS operation and sustain a 10-kHz waveform digital correlation process. The 10 kHz, its 3.3-V swing, and the probe material (steel), tend to minimize any potential for corrosion build-up at the probes. Finally, the entire wet test runs over several hundred cycles at 10 kHz for each sensor to ensure that a wet or unwet condition is consistently stable before a determination is made and reported.

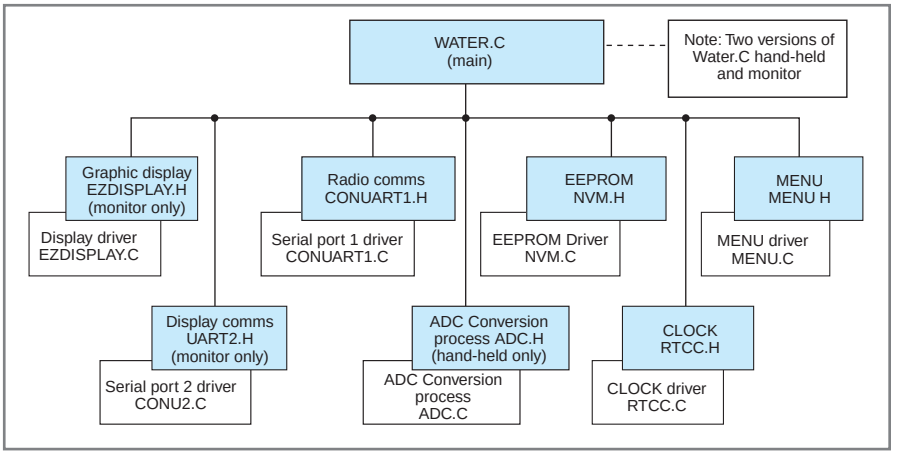


Figure 5—PIC24F software is commonly organized in this fashion. This is how it is implemented in both hand-held and monitor and control units.

DEPLOYED SYSTEM

The valve box is shown in Photo 4. The solenoid irrigation valve's PVC manifold constructions are also in the box. For each manifold, a garden hose must be connected for the water supply (see Photo 1).

I mounted a 0.5" garden hose adapter to the water entry for each manifold. The solenoid relay I chose was a Lawn Genie 54048 0.75" in-line valve, which is available at most hardware stores. I used crushed white rock to weigh down the remote box.

OPERATING SOFTWARE

No picture of a microcontroller-based design would be complete without an overview and description of the software. As I mentioned earlier, the design is written in C code using Microchip's C30 compiler. The design is multifile and the file structure is used commonly across all of the designs. What is unique to the design is constrained and captured in the WATER.C file. This is also the main function file for C. Again, WATER.C is different for the hand-held and the monitoring and control unit with all of the other files being identical.

Photo 4—This valve box shows the manifold of two water valves, associated hoses, and control and sense wires.

The file structure captures interfaces via their .H and the actual functional code in their .C manifestations. There are seven file sets all together (see Figure 5). The EZDISPLAY file set is a driver set to communicate with a Multilab graphics display in its predefined manufacturer's protocol. This file set is active only in the hand-held unit; it is reliant on UART2 for its physical connectivity. The support code needed for its operation is UART2.H and CONU2.C. This set is a basic console with 9,600 bps, 8 bits, no parity, 1 stop bit initialization, and input and output routines for exercising UART2. This is required for EZDISPLAY.

The CONUART1 file set is a basic console with 9,600 bps, 8 bits, no parity, 1 stop bit, initializing, and input and output for exercising UART1, which is connected to the MaxStream radio. This code enables UART1 for interrupts when a character is received over radio and made available to the PIC24F64GA002.

The ADC file set is the basic initializing and input driver for the PIC24F ADC. It is active only in the monitor and control unit. The EEPROM file set is the basic driver to read and write from the 25LC101A 1-KB EEPROM using SPI1. The RTCC file set is a basic driver for initialization, reading, writing to the PIC24F64GA002's clock, and setting the alarm. This code optionally enables the RTCC alarm to interrupt.

The MENU file set is a low-level console-based menu system that can be used to enter proper date/time arguments for clock, read/write ASCII to EEPROM, bit control, and exercising relays. This functions as a minitrapdoor to exercise the system using the MaxStream X-CTU terminal emulation code over a radio link. The receive character interrupt service is active in both units. The RTCC interrupt is also active in both designs with a TBD function for hand-held and an autostatus reporting capability for the monitor and control on alarm.

Figure 6 is indicative of the WATER.C code in the hand-held unit. An executive routine drives the graphic display. During operator direction within the constraint of the graphic

menu presentation, are the available buttons that the operator uses to operate the system. Note that the status display is a template initially. It requires incoming status messages to be received from the remote monitor and control units to complete the presentation. Characters are captured by interrupt from the radio, but they are processed for operational context by software one character at a time.

Figure 7 is indicative of the WATER.C code within the monitor and unit. An executive routine continuously performs the wet test on all the sensors while checking the radio communication for incoming characters between tests. After receiving a command from the hand-held, the monitor and control checks for its address condition. The address condition is valid only for a single command. The unit must be addressed for each new command.

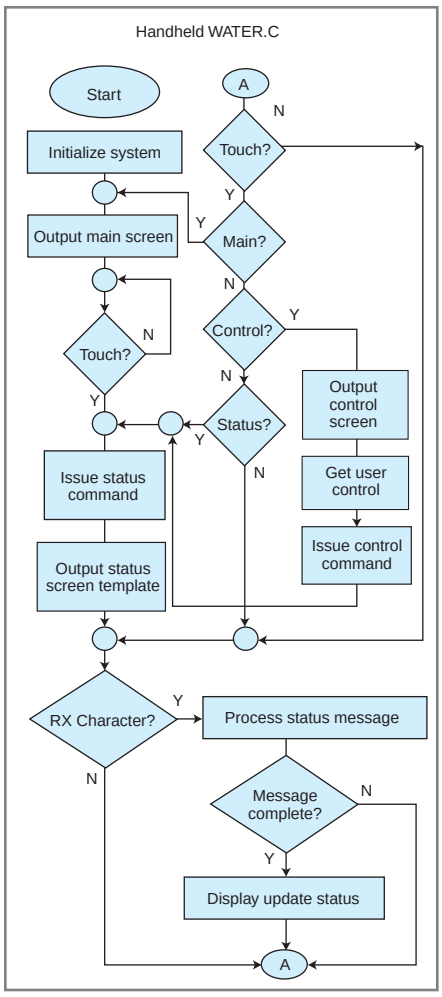


Figure 6—These are different executive flows within WATER.C, as configured for the hand-held and the monitor and control units.

MESH OR PEER-TO-PEER?

The radios are based on MaxStream OEM RF embedded modules that are compliant with the 802.15.4 standard and operate within the 2.4-GHz band. The MaxStream XBee type is used in this design. They have an outdoor range of up to 300′ and data rates of 250 kbps. The modems interface to a microcontroller through a 3.3-V logic-level asynchronous serial port. By default, these modules operate in a "transparent mode." When operating in this mode, the modules act as a serial line replacement. All UART data received through the DI pin is queued for transmission. When RF data is received, the data is sent out of the DO pin. To modify or read RF module parameters, the module must first enter into Command mode—a state in which incoming characters are interpreted as commands. This is achieved by issuing a Command mode

sequence, which looks like a Hayes modem command sequence. The manufacturer supplies PC-based software designated X-CTU that enables you to test and configure modules that are connected to an adapter board that uses a serial port or USB. Out of the box it is trivial to set up and test a point-to-point wireless network. I have expanded on this to quickly configure a peer-to-peer network. For my purposes, each module at the transparent level uses a unique address one character. In this case, it is a simple designator ASCII X for site A and ASCII Z for site B. I was able to do this because there is never an X or Z ASCII character used in any other communications within the system. In fact, most communication is restricted to messages with ASCII numbers, a slash, or a colon. The hand-held uses this address to designate which module a command is intended for. The addressed module responds with an echo of its address and results of the command operation. I tried to keep things simple because most commands are simply S (status), R (relay), and M (menu). The Menu command is not used during normal operation; it's used only for setups using the X-CTU in terminal mode (thereby allowing me full PC keyboard and screen) to set the clock, configure the alarm, and modify the EEPROM.

The peer-to-peer configuration works well for my small-to-medium-sized system; however, for a larger system, a mesh will be required. I tried MaxStream XBEE_B modules that support mesh. The latencies of communication are longer than the peer to peer, but the range and extensibility of mesh shouldn't be underestimated because you may have several EarthBox sites spread over a wide area. In the mesh configuration, one of the units must function as a coordinator, while the others are router/end devices. The coordinator must be on first to be able to coordinate all of the

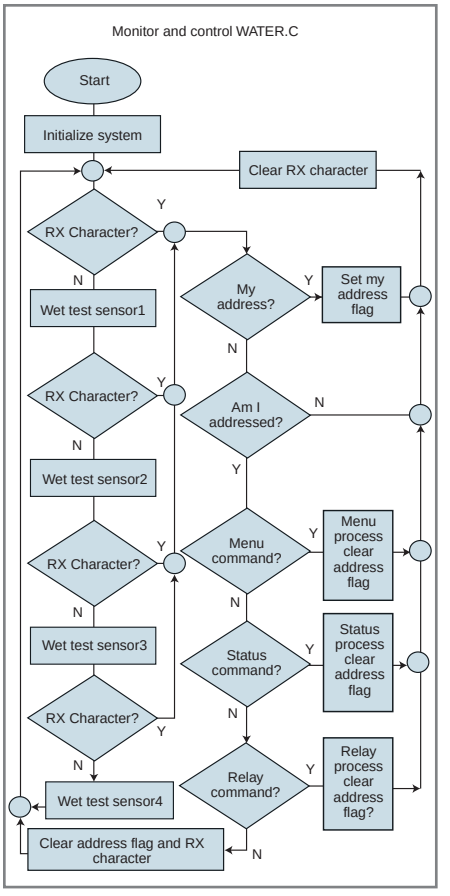


Figure 7—These are different executive flows within WATER.C, as configured for the hand-held and the monitor and control units.

other routers or end devices within the mesh.

START ASSEMBLING

I assembled a comprehensive watering system using a PIC24F64GA002 and commercial radio electronics. The system, as configured, represents an initial prototype that I can easily modify for greater range and capability. In small configurations, peer-to-peer radio setups are sufficient; however, as the system expands, the need for MESH becomes necessary.

Tom Kibalo (tkibalo@gmail.com) holds an M.S. in electrical engineering. He has been a practicing engineer for over 36 years in the Washington, D.C., area. In addition to engineering, Tom is an adjunct faculty member and technical advisor at a local community college, where he teaches courses on microcontroller technology. He is also a Microchip academic partner. Tom enjoys reading, biking, gardening, and bonsai.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/223.

SOURCES

Darlington drivers, diodes, LEDs, relays, and solder
All Electronics Corp. | www.allelectroincs.com

XBEE Radio modules
Digi International, Inc. (distributor) | www.digi.com

EarthBox Gardening system
EarthBox | www.earthbox.com

Capacitors, clock crystal, resistors, and solderless breadboard
Jameco Electronics | www.jameco.com

54048 In-line valve
Lawn Genie | www.lawngeniestore.com

MCP608 Op-amp, PIC24F64GA002 microcontroller, and PICkit2 Debug Express
Microchip Technology, Inc. | www.microchip.com

Graphic LCD
Multilabs | www.multilabs.net

Regulators board and XBEE carrier adapter
SparkFun Electronics | www.sparkfun.com

Solar Measurements

Techniques for Evaluating Panels

Is designing a photovoltaic system in your future? Ed describes how to model and measure the characteristics of a small panel. In doing so, he covers photovoltaic fundamentals, circuit modeling, and instrumentation.

ABOVE THE GROUND PLANE

by Ed Nisley

Long ago, my buddy Eks observed: "If solar power is so good, why is there winter?" His excellent engineering judgment notwithstanding, photovoltaic (PV) panels appear everywhere these days: dark plates facing the sun from buildings, backyards, and even utility poles.

Common performance numbers concentrate on the big picture with sweeping assumptions that might well not apply in a specific location. Although everybody seems to know the "kilowatt per square meter" value of total insolation, few people can relate that to a system's actual, usable output power.

In this column, I'll start from the other end by modeling and measuring the characteristics of a small panel. You can use similar techniques to evaluate larger panels, perhaps to verify a manufacturer's claims or to characterize a panel's actual performance in your system. After you work through the details, you should have a much better understanding of how the big picture looks from your standpoint.

MODELING

Figure 1 presents the classic circuit model of a PV cell: a current source in parallel with a diode. The component values correspond roughly to one cell of the panel appearing in Photo 1. Unlike larger panels made from discrete slices from silicon wafers, this panel is a flat glass sheet with deposited cells separated by barely visible vertical stripes.

D1, the 1N914 diode in the model, has a forward voltage and maximum current reasonably close to those of the panel's cells, but it's obviously unsuited for modeling high-power panels. Achieving a better match may require tweaking a Spice "ideal diode" model, rather than using a standard part.

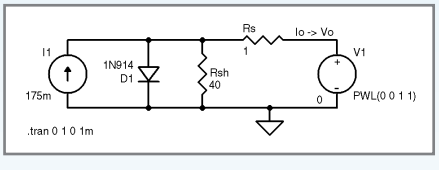


Figure 1—The simplest circuit model for a photovoltaic panel represents the DC response, although a 1N914 diode probably isn't a good match for a real panel. The piecewise linear voltage source on the right provides a variable load.

Current source I1 models the cell's photocurrent, which varies in a surprisingly linear manner with incident illumination. The nominal value is essentially the cell's I_{SC}, the short-circuit current produced with 0 V across the output terminals.

The diode's forward voltage controls V_{OC}, the cell's open-circuit voltage. Unlike I_{SC}, V_{OC} doesn't vary much with illumination: even diffuse sunlight can punch the cell's voltage close to its nominal full-sun value. Even with little power (and, thus, current) behind V_{OC} at dawn or dusk, inadvertent contact with a high-voltage panel's output can flatten you right out.

Shunt resistor R_{SH} adjusts the slope of the current-voltage curve near the short-circuit current limit, while series resistor R_S affects the slope near the open-circuit voltage. You can calculate those values from measurements or use an eyeballometric curve-fitting process. Both resistors represent parasitic losses within the panel that depend largely on its chemistry and construction.

The family of curves in Figure 2 simulates several values of R_S along with the output of the actual panel in Photo 1 scaled to a single-cell V_{OC}. The R_S = 1.5-Ω curve seems to be a reasonable match, although further fiddling with R_{SH} and D1 might improve the fit.

A silicon-based PV cell has a V_{OC} near 500 to 700 mV and an I_{SC} that, for a given technology, varies linearly with the cell's surface area. Larger panels use cells wired in series for higher output voltage and in parallel for higher output current.

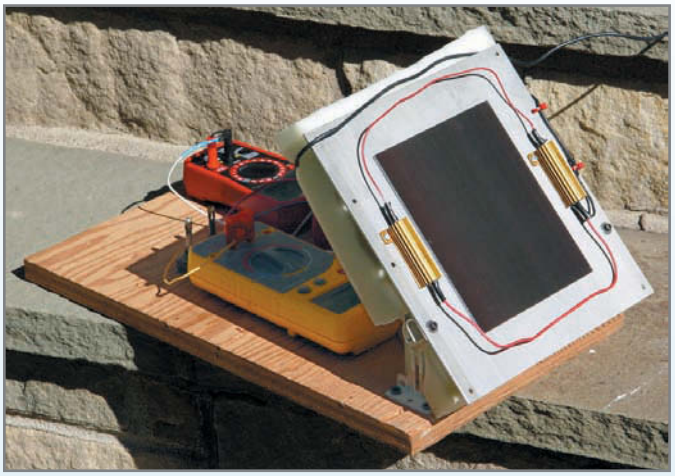


Photo 1—Measuring the performance of a photovoltaic panel requires little more than a handful of multimeters and some sunlight. The two power resistors can boost the panel's temperature to show thermal effects on its output.

Just as in the real world, modeling multiple-cell panels involves little more than copying and pasting single-cell models in series or parallel. However, if parts of your real-world panel can be shaded from the sun, then your panel model must incorporate bypass diodes and vary the current sources across the array.

You can create a more complex piecewise-linear model of each cell with either a mathematical black-box table or curve-fitted combinations of diodes and resistors. Spice simulations may run slowly with a pure look-up table source, but figuring the values of stacked diode-resistor pairs rapidly turns into a fierce mathematical exercise that isn't relevant here.

Before you spend a great deal of effort modeling a specific cell or panel, however, remember that real-world performance depends strongly on illumination and temperature. It's generally better to verify that your design works with an approximately correct source than to get distracted by the third or fourth decimal place of an incomplete model.

Remember that the standard model represents only the DC response, which may turn out to be a serious omission for switching power converters or grid-connected systems. We'll investigate that later.

TEST SETUP

It's relatively simple to measure the fundamental DC properties of a PV panel: record the output voltage and current as functions of a variable load. Getting well-calibrated values requires a known illumination, but we civilians can slide by with the standard sun found above any backyard lab. It should come as no surprise that all values vary nonlinearly with temperature, so recording the panel temperature is vital.

For obvious reasons, I was unwilling to buy a commercial PV panel for this column, but I had the 11-cell panel in Photo 1 in my parts heap. It came from a surplus supplier with only two specifications: the 5″ × 6″ physical size and a 7.4 V at 170 mA rating. The rear surface has aluminum contacts covered with transparent plastic insulation that must be scraped off to make contact. Soldering anything to aluminum is difficult, particularly atop a glass substrate, so I had to find another connection method.

The panel's low-current rating suggested simply gluing connectors to it with carbon-based wire glue, but the resistance turned out to be 15 to 20 Ω per contact. I popped those off and discovered that the glue left an impervious, completely insulating layer atop the aluminum. Perhaps there's a strange chemical reaction going on?

After several other fruitless experiments, I folded copper fabric adhesive tape to produce a metallic connection on both sides, clamped it under a 0.1875″ quick-disconnect tab, and epoxied the entire affair to the panel. Photo 2 shows the resulting joint, which is surprisingly durable and has a resistance well below the resolution of any of my meters.

I machined two openings for the contacts in an aluminum sheet, spread a thin layer of Bondo body filler on the panel, and squished it in place. The aluminum backing adds mechanical rigidity, provides a convenient heatsink, and simplifies mounting the entire affair to a scrap of plywood. While Bondo isn't a true thermal epoxy, it's good enough for this application.

I also epoxied a small brass tube to the center of the aluminum sheet and inserted a thermocouple to measure the temperature. The reading from the rear of the aluminum sheet is within a few degrees Fahrenheit of the value reported by my noncontact IR thermometer for the panel's front surface in full sun.

The two 50-Ω power resistors on either side of the panel enable me to raise its temperature well above ambient, which isn't something you want in a normal application. I also tucked a foam sheet behind the panel to reduce the effect of breezes.

The instrumentation consists of three multimeters attached to the plywood base, one apiece for panel voltage, current, and temperature. Each of my multimeters has a different current-sensing resistor, so I picked the meter with a 1-Ω resistor for the panel-current job.

My parts heap also disgorged a pair of big panel-mount potentiometers that I glued to the plywood; they're barely visible to the left of the yellow meter in Photo 1. This *definitely*

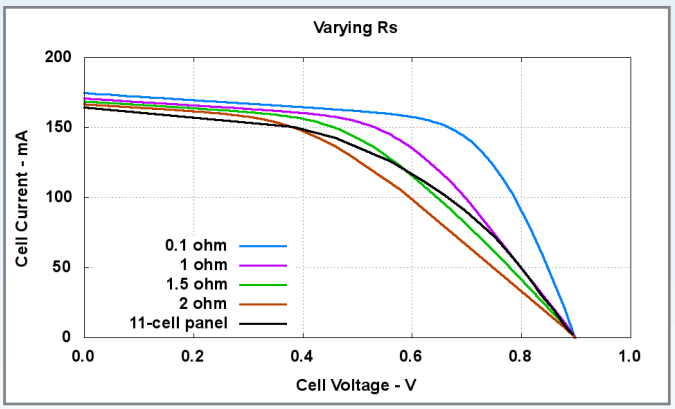


Figure 2—Tweaking R_S can nudge the Spice simulation close to the actual panel characteristics. Even if they're not an exact match, it might be close enough given the usual real-world variations. The black curve is from the panel in Photo 1, scaled to a single-cell V_{OC}.

won't work for higher-power panels, as pots have a very low power dissipation rating. On the other hand, this panel is good for 1 W, tops, and those pots were well past their best-used-by date. Later on, I'll show you how to build a variable load the right way, but this will suffice for now.

I wired a 3-kΩ potentiometer in series with a 300-Ω potentiometer, both connected as variable resistors, to get enough resolution over the entire current range. The panel's V_{OC} was 8.3 V in full sun, but 2.5 mA through the potentiometers at maximum resistance dropped it to 8 V; I disconnected the potentiometers to get the true V_{OC}. The larger potentiometer covered the range up to 25 mA and the smaller one went to nearly I_{SC}. I shorted the potentiometers to measure the output current limited by the meter's 1-Ω internal resistance.

With all of that wired up, I sat on the front steps of our house, aimed the panel directly at the sun, and started recording data.

POWER PRODUCTION

The purple curve arcing downward from the upper left of Figure 3 represents the panel's performance at about 80°F, before it warmed up in the sun. The data points are about 10 mA apart up to 100 mA, then every 25 mA above that, so there are relatively few points in the left half of the curve.

Although the I_{SC} and V_{OC} points are easy to measure, the panel delivers no power to the load at either point. The humped purple curve plots the panel's power output calculated by multiplying the panel's terminal voltage and current at each point. This clearly shows that the maximum power output occurs near 5 V at about 125 mA.

That's just over 600 mW, far less than you'd estimate by simply multiplying the only two numbers in the panel's specs: 7.4 V and 170 mA. Those are evidently the panel's nominal V_{OC} and I_{SC}, not its peak output.

The other curves in Figure 3 show the effect of temperature. The panel stabilized at about 105°F in full sun with a breezy ambient of 75°F, producing the blue curve. Applying 24 V to the power resistors boosted the panel to 138°F to produce the green curve.

Both V_{OC} and I_{SC} change with temperature, although the details differ. As with all such things, careful measurements trump preconceptions. Because the voltage decreases with increasing temperature, the maximum output power also decreases: unfortunately, hot panels produce less power than cold panels.

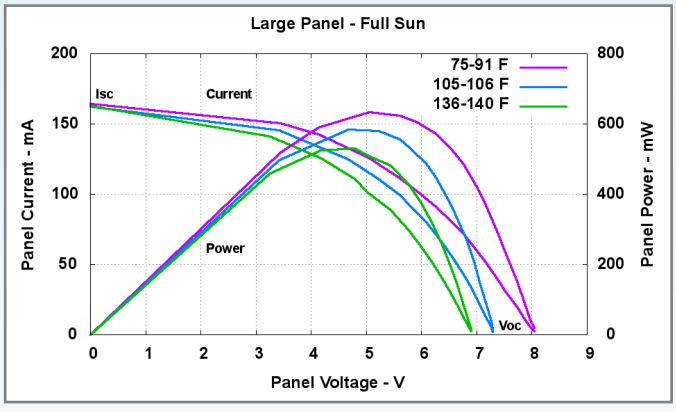


Figure 3—Increasing temperature decreases the panel's open-circuit voltage, but changes the short-circuit current only slightly. The maximum power point occurs at lower voltages and currents as the panel heats up.

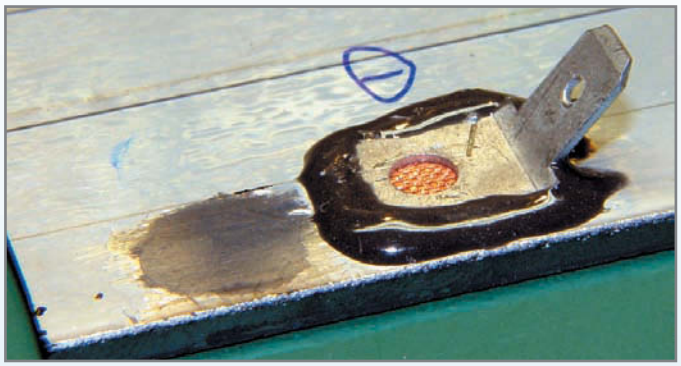


Photo 2—I attached quick-disconnect terminals using copper fabric tape and epoxy. The smudged area to the left shows the adhesive remnant from a surprisingly high-resistance wire glue installation.

Panels produce their highest voltage when they're coldest, a situation that generally occurs just before dawn. If you're working with multiple panels in series, the early dawn output voltage may be *much* higher than you expect from the nominal rating.

The green power curve in Figure 3 shows that the panel produces 0.5 W (4.8 V and 110 mA) at the sort of temperature you'd expect from a weatherproof installation. To extract that power from the panel, however, the load must draw 110 mA at 4.8 V, not whatever voltage or current might be required for, say, optimal battery charging or an instrument's power supply.

SIMPLE BATTERY CHARGING

Old-school PV battery chargers simply soaked up whatever current the panel could produce at whatever terminal voltage the batteries supported. As you've seen in my previous columns, proper battery charging requires more than random trickle charging, so a dumb charger (or none at all) trades off complexity for battery lifetime and performance.

This panel's output suggests it could charge a four-cell NiMH battery pack. A fully charged cell runs about 1.5 V, so the final charge current at 6 V would be 60 mA in full sun at 138°F. To prevent overcharging, the cells should have a capacity of about an order of magnitude larger. Contemporary low-end AAA NiMH cells run about 750 mA•h, with some claiming upwards of 1000 mA•h.

With a quartet of 750-mA•h cells discharged to the usual 900 mV/cell endpoint, the panel will produce 120 mA at 3.6 V and charge the pack at an initial C/6 rate.

Unfortunately, that rate won't last while the battery pack voltage increases and it certainly doesn't apply all day unless the panel tracks the sun. In fact, this solar panel would take two or three days to fully charge a four-cell AAA pack, which is probably not what you were expecting.

Because the panel's output is fully devoted to charging the cells, there's no power "left over" to run anything else. You could swap a pair of four-cell packs between the solar charger

and your gadget, but only if a pack lasts two or three days in normal use. A few cloudy days can put an end to that schedule, though, because there's just not enough energy available.

But it gets worse.

EFFICIENCY

That kilowatt per square meter of solar energy that ever ybody quotes should come with a few asterisks: the utterly absorbing surface must be perpendicular to the sunlight. A square meter of solar PV panel doesn't produce anything close to a kilowatt.

The primary problem is the conversion efficiency: not all photons falling on the panel turn into electrical current. A good rule of thumb for packaged panels seems to be around 10% to 15%, with higher efficiencies priced exponentially higher.

The panel in Photo 1 measures about 0.02 m^2 and should absorb about 20 W when perpendicular to the full sun. Figure 3 shows that it produces about 0.6 W at best, for an overall efficiency around 3%. Perhaps it was a surplus panel for good reason.

I had no problem tracking the sun with my test setup, but that doesn't work for larger installations. Obviously, if the panel isn't perpendicular to the sun, it won't absorb as much energy and its output will drop. Check the references and run the numbers for your location; you may lose 10% to 30% of the incoming energy.

I also own a Power traveller Powermonkey solar eXplorer,

Photo 3—I draped the Powermonkey solar panel over my test setup for full-sun measurements. Notice the lack of shadows due to a passing cloud: good for photography, bad for power!

primarily because I chanced upon a $100 rebate that brought the price down to my level. The basic Power monkey consists of a 2.2-A•h lithium batter y inside a small plastic case; an assortment of connectors that match many common phones, PDAs, cameras, and other electronic gadgets; and a wall-wart charger.

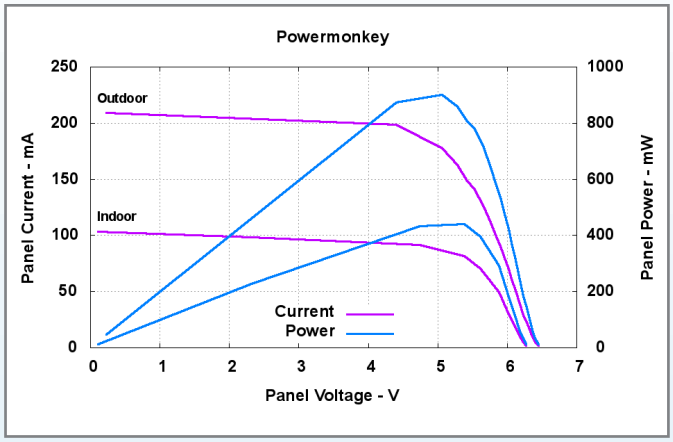


Figure 4—The Powermonkey panel has a much higher efficiency, producing more power from a smaller area. Putting it on a table behind two layers of window glass cuts its power output in half.

An internal 5-V DC/DC converter produces a constant output voltage that's compatible with most gadgets, so, in theory, the Powermonkey can recharge your gadget when you're far from an AC outlet. Even if it can't produce a full charge, you may be able to make one last phone call.

The "Solar" option adds the folding solar panel in Photo 3, which is supposed to charge the Powermonkey battery from ordinary sunlight. I had my doubts about this from the start and some measurements confirmed my suspicions.

The "Outdoor" traces in Figure 4 show the output of the Powermonkey panel during a day with thin, high clouds. I have no way to measure the actual insolation, but it was a typically sunny day in these parts. The peak power output is 900 mW with the panel perpendicular to the sun. Moving the panel indoors, behind two panes of 1950s-era glass, cut the power output in half.

You can see where this is going. The Powermonkey can produce 2.2 A•h at 5 V, for a total of 11 W•h. Assuming 75% charging efficiency, you must inject 900 mW for 16 h to reach full charge. If the panel is inside your window, then plan on over 32 h. If you only get 5 h of usable sunlight per day, then a full charge will take three or six days.

The instruction manual alludes to this situation: "6 hrs of sunlight will charge powermonkey-eXplorer by one third."

On the other hand, the Powermonkey solar panel is far more efficient than my larger surplus panel. With an active area of about 0.007 m^2, the panel should absorb 7 W in full sunlight. It actually produces 900 mW for an efficiency of nearly 13%.

That's pretty good, but you need more than four times the panel area to recharge the Powermonkey in a day. Perhaps it's no surprise that they named their larger model *Powergorilla*?

CONTACT RELEASE

Logging data by hand makes perfect sense when you're first experimenting with a gadget, but it gets really tedious really quickly. In my next column, I'll take a look at the analog circuitry required to automate the process, along

with a bit of firmware to get rid of the paperwork, too.

Ed Nisley is an EE and author in Poughkeepsie, NY. Contact him at ed.nisley@ieee.org with "Circuit Cellar" in the subject to avoid spam filters.

PROJECT FILES

To download the additional file, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/223.

RESOURCES

J. Bachiochi, "Collecting Solar Energy," *Circuit Cellar* 209, 2007.

R. Campbell, "A Circuit-based Photovoltaic Array Model for Power System Studies," www.ee.washington.edu/research/sesame/publication/Conference/2007/Campbell_PWL_PV_Model_NAPS2007.pdf.

K. Chapman, "Solar Panel Monitor," *Circuit Cellar* 185, 2005.

S. Ciarcia, "Solar-Powering the Circuit Cellar," *Circuit Cellar* 209, 210, and 211, 2007.

T. Kerekes, "High Frequency Analysis and Modelling of Transformerless PV Inverter Systems," www.iet.aau.dk/~tak/personal/phd_proj_descr.htm.

C. Landau, "Optimum Operation of Solar Panels," www.macslab.com/optsolar.html.

Linear Technology Corp., "SwitcherCAD III," www.linear.com/designtools/software/switchercad.jsp.

G. Martin, "Living and Working Off the Grid," *Circuit Cellar* 216 and 218, 2008.

New York State Energy Research and Development Authority, www.powernaturally.org/default.asp.

E. Nisley, "Battery Capacity: Charge," *Circuit Cellar* 201, 2008.

———, "Battery Capacity: Discharge," *Circuit Cellar* 199, 2007.

———, "Real-World NiMH Charging," *Circuit Cellar* 221, 2008.

Wikipedia, "Maximum power point tracking," http://en.wikipedia.org/wiki/Maximum_power_point_tracker.

SOURCES

Surplus solar panels
All Electronics Corp., www.allelectronics.com
The Electronics Goldmine, www.goldmine-elec.com

Wire glue Anders Products | www.wireglue.us

Bondo body filler
Bondo Corp. | www.3m.com/us/auto_marine_aero/bondo/catalog_browsec9d6.html?parNbr=26

gnuplot gnuplot | www.gnuplot.info

Powermonkey solar eXplorer
Powertraveller | https://powertraveller.com/iwantsome/primatepower/powermonkey-explorer/

by Larry Standage

Processor Modification

Put Soft-Core Processors to Work

Want to change your processor? No problem. Larry explains how to use the flexibility of soft-core processors to make the processor you want.

For a while now, selecting a microcontroller for an embedded design has been a process of deciding which offerings yield the features you want and need. Having been involved in selecting processors for embedded devices, I know the frustration of trying to figure out which controllers have sufficient memory and peripherals to get the job done. Sometimes you need one more serial port or a few more I/O lines. Worse yet, the other option is selecting a processor with too many features, most of which your project doesn't require.

System on Chip (SoC) devices offer some flexibility in that you can exchange the features of the controller and peripherals. But you are usually tied to the IP cores that the vendor offers you. And forget about changing the peripherals or even the processor itself! One of the greatest benefits of programmable logic is the ability to configure the device to do what you need and want. With offerings such as those at www.opencores.org, you can have the processor and peripherals you want and modify them for your application. The genesis of

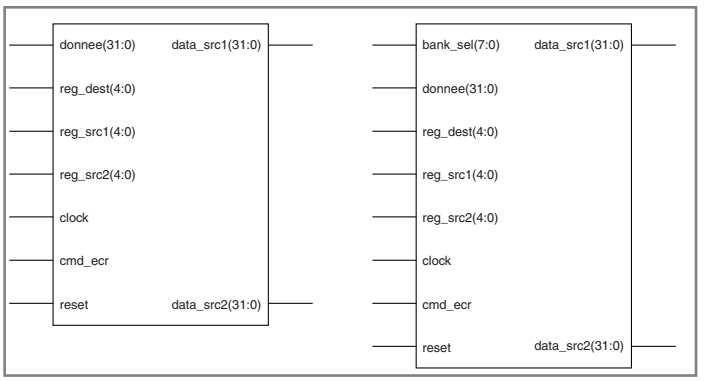


Figure 1—These are changes to the register bank (banc) module, with the addition of the bank_sel control lines. Everything else remains the same functionally.

this idea came from studying real-time operating systems. When you study RTOS behavior, especially context switching, it quickly becomes apparent just how much time it takes to make a context switch from one task to another. In a RISC-type processor, the time it takes to save and restore the registers becomes a significant factor in that context switch time. I asked myself, "How could that time be reduced?"

Suppose you could have a set of registers in your processor for each task in your software. I'm not talking about just reserving registers 1–5 for task one, 6–10 for task two, and so on. I am talking about having a full set of general-purpose registers for each task. One instruction is all that would be needed to go from one set of registers to another. Wouldn't that save a lot of time?

With that idea in mind, I set out to find a processor that I could alter and give that kind of feature. I wanted to be able to give a processor an arbitrary number of register banks and to use one instruction to switch those registers around.

Listing 1—The addition of the num_banks parameter enables you to specify how many register banks to create. By giving a default of 1, you don't have to specify this parameter, and 1 will be created. The bank_sel input is added to control which bank is in use.

```
entity banc is
generic ( num_banks : natural := 1 );
port (
clock : in bus1;
reset : in bus1;

-- Register addresses to read
reg_src1 : in bus5;
reg_src2 : in bus5;
bank_sel : in bus8;

<-- snip -->
);
end banc;
```

FINDING THE PROCESSOR

The toughest part of this project was selecting an appropriate soft-core processor to modify. I knew the ideal candidate would have certain characteristics: an easily obtainable, modifiable source (I'm more familiar with VHDL, so this is the language I was leaning toward); a debug interface to enable me to easily examine the state of the processor and step through code to see if the desired changes take place; a competent software toolset, such as GCC; and an instruction set, which would enable me to add instructions without causing problems for other parts of the processor.

I originally wanted to modify the Xilinx MicroBlaze soft-core processor, because I had been using one at work and I was familiar with it. I could have used the GCC toolchain and the JTAG debug interface to monitor the program execution. However, while Xilinx purportedly offers the source for the MicroBlaze for $5,000 (which in itself was far too expensive), when I contacted Xilinx it appeared that it did not actually sell or give away the source code.

Next, I turned to www.opencores.org to find a core. At the site, source code for IP cores is available under the GNU public license (GPL) and the limited GNU public license (LGPL). There are a variety of cores that can be modified and have freely available sources. Some of the processor cores are still in the early stages of development, but several are mature and even in use in commercial products.

One processor core is the OpenRISC 1200. The OpenRISC is planned as a series of processors, and the 1200 is one of the first real implementations of the specification. The OpenRISC 1200 includes many

Listing 2—The lines added here create a num_banks x 31 array of registers, initialize those registers to 0 on reset, read the data from the appropriate bank and register, and write data to the appropriate bank and register.

```
architecture rtl of banc is
  -- The register bank
  type tab_reg is array (0 to num_banks - 1, 1 to 31) of bus32;
<-- snip -->

  signal bank_selected : integer range 0 to num_banks - 1;
begin
<-- snip -->

  bank_selected <= to_integer(unsigned(bank_sel));
  data_src1 <= (others => '0') when adr_src1=0 else
    registers(bank_selected, adr_src1);
  data_src2 <= (others => '0') when adr_src2=0 else
    registers(bank_selected, adr_src2);

  process(clock)
  begin
    if clock = '1' and clock'event then
      if reset = '1' then
        for j in 0 to num_banks - 1 loop
          for i in 1 to 31 loop
            registers(j,i) <= (others => '0');
          end loop;
        end loop;
      elsif cmd_ecr = '1' and adr_dest /= 0 then
        -- The data is saved
        registers(bank_selected, adr_dest) <= donnee;
      end if;
    end if;
  end process;

end rtl;
```

of the same features as the MicroBlaze, including GCC support for software development, a debug interface that can go over JTAG, and a Wishbone bus master support, which enable it to communicate with many different devices that also support Wishbone with minimal effort.

The OpenRISC 1200 is written in Verilog, which is not a language I am familiar with. Finding the portions of the source code that needed to be modified was difficult, but it was not the hardest challenge. Building a working system for my development board was the biggest challenge. After downloading instructions for building a system using the Wishbone bus and following the steps, I had a system that could be built with Xilinx ISE tools. However, when I tried to connect a debugger to the core to download a program, I ran into problems. The gdb debugger kept complaining that I needed either to enable or to include a debug module (even though it was already included), and that it wasn't talking to the processor.

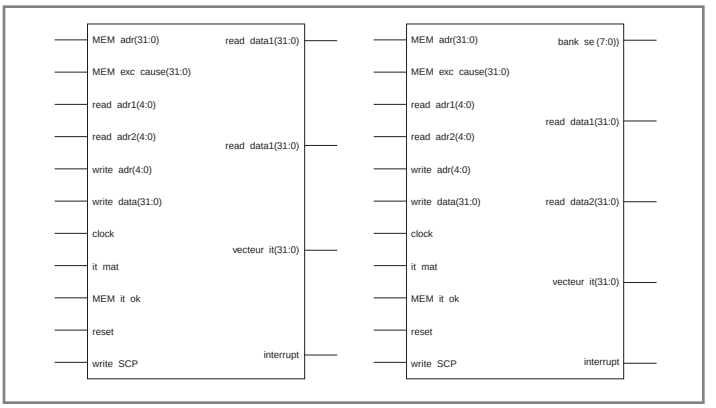


Figure 2—These are the changes to the system coprocessor (syscop) module. The register bank control line (`bank_sel`) is the only addition here.

Instead of sorting out the problem, I abandoned the OpenRISC 1200 because I couldn't afford to spend more time trying to figure out what I was doing wrong. After the problems with the OpenRISC 1200, I was leery of searching for another core on www.opencores.org. However, after another search, I found the miniMIPS project.

The miniMIPS started out as a school project by a group of French students, and they submitted the results to www.opencores.org. Based on the MIPS I processor, it is a 32-bit Von Neumann architecture, with a five-stage pipeline (instruction extraction, instruction decoding, execution, memory access, and update registers). The processor normally holds 32 32-bit registers; but one, R0, is always read as 0 and cannot be written to.

The processor has a simple interface and is written in VHDL. It does not have GCC support as far as I can tell. The processor comes with little to no documentation, nor does the assembler the students built. There isn't a debug interface, so I wasn't

	USBee ZX	USBee AX-Standard	USBee AX-Plus	USBee AX-Pro	USBee DX
Oscilloscope		✔	✔	✔	✔
Logic Analyzer	✔	✔	✔	✔	✔
Mixed Signal Osilloscope		✔	✔	✔	✔
Digital Signal Generator	✔		✔	✔	✔
Digital Voltmeter		✔	✔	✔	✔
Bus Data Extractors					✔
USB (Low and Full Speed) Decoder	✔		✔	✔	✔
I2C Decoder	✔		✔	✔	✔
SPI Decoder	✔		✔	✔	✔
Async Decoder	✔		✔	✔	✔
CAN Decoder					✔
I2S Decoder					✔
1-Wire Decoder					✔
SM Bus Decoder					✔
PS/2 Decoder					✔
Parallel Decoder					✔
Serial Decoder					✔
Click and Drag Bus Decoding					✔
Data Logger	✔			✔	✔
Frequency Counter	✔			✔	✔
Remote Controller	✔			✔	✔
PWM Controller	✔			✔	✔
Frequency Generator	✔			✔	✔
I2C Controller	✔			✔	✔
Pulse Counter	✔			✔	✔
USBee Toolbuilder Source Code	✔			✔	✔
Digital Channels	8	8	8	8	16
Maximum Sample Rate (Msps)	24				
Buffer Depth	Uses your PC RAM (Hundreds on millions of samples)				
Analog Channels	0	1	1	1	2
Analog Input Range	+/-10V				

Powerful Debugging - Small and Portable

www.USBee.com

able to use a debugger such as gdb to see the registers change. But I was able to build a system with the miniMIPS. This was the best selection with a working system.

DEVELOPMENT SYSTEM

The development system I found was a Digilent S3BOARD Spartan-3 development board. It comes with a Spartan-3 XC3S200, 1 MB of RAM, slide switches, push buttons, LEDs, a four-digit, seven-segment display, and serial, VGA, and PS/2 ports. I had the board shipped with a Spartan-3 XC3S1000 because I needed as much room as I could get for the project.

I also ordered an expansion board with access to various pins in case I needed to probe the I/O pins of the FPGA. For the hardware implementation, I included several items, although I did not use them all. The modified miniMIPS processor, the interface to the SRAM (primarily to have visibility to the address and data lines of the miniMIPS), a program ROM device, the four-digit, seven-segment display, and the LEDs. With the hardware selected, I built a system with the miniMIPS, which provided the basis for the updated core.

Listing 3—I added an output port to the system coprocessor module. The port is connected to the register bank module and selects which register bank is in use.

```
entity syscop is
port
(
clock : in std_logic;
reset : in std_logic;

<--SNIP-->

-- Reading request in register bank
read_adr1 : in bus5; -- Address of the first register
read_adr2 : in bus5; -- Address of the second register
read_data1 : out bus32; -- Value of register 1
read_data2 : out bus32; -- Value of register 2
-- Register bank selection
bank_sel : out bus8 -- Which register bank to use
);
end syscop;
```

MODIFICATIONS

With the physical hardware selected and implemented, I started modifying the processor. The full source for the VHDL source code is available on the *Circuit Cellar* FTP site. Note that the VHDL source code shown in this article has had some of the comments removed in order to make it presentable.

Let's start by looking at the register bank implementation, as given in banc.vhd, which is posted on the *Circuit Cellar* FTP site. The two

Listing 4—I added a new register to the system coprocessor module to control the register bank. When updates to that register occur, an interrupt is triggered to flush the RISC pipeline. When other interrupts are triggered, the register bank will be saved and set to 0.

```
architecture rtl of syscop is

subtype adr_scp_reg is integer range 12 to 16;

type scp_reg_type is array (integer range adr_scp_reg'low to adr_scp_reg'high) of bus32;

-- Constants to define the coprocessor registers
constant COMMAND : integer := 0;
constant STATUS : adr_scp_reg := 12;
constant CAUSE : adr_scp_reg := 13;
constant ADRESSE : adr_scp_reg := 14;
constant VECTIT : adr_scp_reg := 15;
constant BANKSEL : adr_scp_reg := 16;

<-- SNIP -->
signal register_switch : std_logic;
signal save_banksel : bus8;
signal next_address : bus32;

begin
<-- SNIP -->

interrupt <= exception or interruption or register_switch;
next_address <= bus32(unsigned(MEM_adr) + 4);

vecteur_it <= scp_reg(ADRESSE) when (cmd_itret = '1' and register_switch = '0') else
              next_address when (cmd_itret = '0' and register_switch = '1') else
              scp_reg(VECTIT);
              bank_sel <= scp_reg(BANKSEL)(7 downto 0);

<-- SNIP -->

register_switch <= '1' when write_SCP='1' and
                   (adr_dest=BANKSEL or (adr_dest=COMMAND and write_data=SYS_ITRET))
                   else '0';

<-- SNIP -->

-- Command from the core
    if write_SCP='1' and adr_dest=COMMAND then
       case write_data is -- Different operations
             when SYS_UNMASK   => pre_reg(STATUS)(0) <= '1'; -- Unamsk command
             when SYS_MASK     => pre_reg(STATUS)(0) <= '0'; -- Mask command
             when SYS_ITRET    => -- Interruption return command
                                  pre_reg(STATUS)(0) <= save_msk;
                                  cmd_itret          <= '1';
                                  pre_reg(BANKSEL)(7 downto 0) <= save_banksel;
             when others       => null;
       end case;
    end if;

-- Modifications from the interruptions
if interruption='1' then
pre_reg(STATUS)(0) <= '0';
pre_reg(CAUSE) <= IT_ITMAT;
pre_reg(ADRESSE) <= MEM_adr;
pre_reg(BANKSEL) <= (others => '0'); -- Default to register bank 0
end if;

-- Modifications from the exceptions
if exception='1' then
pre_reg(STATUS)(0) <= '0';
pre_reg(CAUSE) <= MEM_exc_cause;
pre_reg(ADRESSE) <= MEM_adr;
pre_reg(BANKSEL) <= (others => '0'); -- Default to register bank 0
end if;

<-- SNIP -->
-- Memorization of the modifications in the register bank

process(clock)
begin
    if clock='1' and clock'event then
       -- Save the mask when an interruption appears
       if (exception='1') or (interruption='1') then
             save_msk <= scp_reg(STATUS)(0);
          save_banksel <= scp_reg(BANKSEL)(7 downto 0);
       end if;

       scp_reg <= pre_reg;

    end if;
end process;

end rtl;
```

versions are depicted in Figure 1. The original implementation is on the left and the modified implementation is on the right. The only addition is a new input, `bank_sel`, which decides which register bank is being used.

To create that addition, I changed the entity declaration of the register bank module (see Listing 1). This gave it a generic parameter to specify how many register banks it was to implement (default is one), and provided an 8-bit input, informing it which register bank it should be using.

I chose an 8-bit input for the `bank_sel` line, because it seemed reasonable to allow up to 256 register banks in an implementation. If you go beyond 256, besides requiring an extremely large part, you will run into issues. For instance, each thread will have little processor time to actually accomplish anything despite how short the context switch time may be.

Within the architecture definition of banc, I converted the one-dimensional declaration of the registers into a two-dimensional array, using `num_banks` as the limit on the array index. Each time a register is referenced, either to read or write, the `bank_sel` input indexes the group of registers (see Listing 2).

Listing 5—The changes in this file reflect the changes to the register bank and system coprocessor modules. This allows other modules in the CPU to see what's going on, if necessary.

```
package pack_mips is

-- Type signal on n bits
subtype bus64 is std_logic_vector(63 downto 0);
subtype bus33 is std_logic_vector(32 downto 0);
subtype bus32 is std_logic_vector(31 downto 0);
subtype bus31 is std_logic_vector(30 downto 0);
subtype bus26 is std_logic_vector(25 downto 0);
subtype bus24 is std_logic_vector(23 downto 0);
subtype bus16 is std_logic_vector(15 downto 0);
subtype bus8 is std_logic_vector(7 downto 0);
subtype bus6 is std_logic_vector(5 downto 0);
subtype bus5 is std_logic_vector(4 downto 0);
subtype bus4 is std_logic_vector(3 downto 0);
subtype bus2 is std_logic_vector(1 downto 0);
subtype bus1 is std_logic;
…
component banc
generic ( num_banks : natural := 1 );
port (
        clock    : in bus1;
        reset    : bus1;

        reg_src1 : in bus5;
        reg_src2 : in bus5;
        bank_sel : in bus8;

        reg_dest : in bus5;
        donnee   : in bus32;
        cmd_ecr  : in bus1;
        data_src1: out bus32;
        data_src2: out bus32
);
end component;

…
component syscop
port
(
        clock    : in bus1;
        reset    : in bus1;
        MEM_adr  : in bus32;
        MEM_exc_cause : in bus32;
        MEM_it_ok: in bus1;
        it_mat   : in bus1;
        interrupt: out bus1;
        vecteur_it : out bus32;
        write_data : in bus32;
        write_adr: in bus5;
        write_SCP: in bus1;
        read_adr1: in bus5;
        read_adr2: in bus5;
        read_data1 : out bus32;
        read_data2 : out bus32;
        bank_sel : out bus8
);
end component;
component minimips
generic ( num_reg_banks : natural := 1 );
port (
        clock    : in bus1;
        reset    : in bus1;
        ram_req  : out bus1;
        ram_adr  : out bus32;
        ram_r_w  : out bus1;
        ram_data : inout bus32;
        ram_ack  : in bus1;
        it_mat   : in bus1
);
end component;
end pack_mips;
```

Next, I wanted to create a new register that would act as the index to the register bank array, preferably in a place that would not require a completely new instruction to access and use it. Fortunately, the miniMIPS has a system coprocessor that handles interrupts and exceptions, and this module provides the necessary features. Figure 2 shows the changes to this module. In this case, I have added only an output, `bank_sel`.

For this change I added a new register, which was accessible using the `MTC0` and `MFC0` instructions. The 8-bit output bus from the syscop provides the control signals that can then be routed to the banc module (see Listing 3).

Some problems arise from switching the register banks in a pipelined architecture. One is what to do about the instructions that are still in the pipeline when the switch happens. If those instructions use values from or write results to registers, data corruption will occur. Those instructions cannot be relied on.

One solution to the problem is to insert three NOP instructions immediately after the register switch to have nothing in the pipeline. A second option is to use three instructions that don't use registers, such as absolute jumps. But both of these options rely on the software programmer to insert the solution and would be difficult to debug.

Instead, this design takes advantage of the fact that the pipeline is flushed under certain conditions.

Listing 6—This is my top-level declaration for the miniMIPS processor. In addition to the selection of the number of register banks, I also bring out the bank selection control signals to monitor with a logic analyzer. In a normal situation, this monitor would not be necessary and could be removed.

```
entity minimips is
generic (num_reg_banks : natural := 1);
port (
clock : in std_logic;
reset : in std_logic;

-- Ram connexion
ram_req : out std_logic;
ram_adr : out bus32;
ram_r_w : out std_logic;
ram_data : inout bus32;
ram_ack : in std_logic;

-- Hardware interruption
it_mat : in std_logic;
-- Bank selection monitor
bank_sel_mon : out bus8
);
end minimips;
```

In the original design, hardware interrupts, software exceptions, and a return from interrupt instruction result in the clearing of the pipeline stages, at which point the pipeline is reloaded. Now the syscop module detects when the register switch is taking place and triggers the same internal interrupt signal as the other triggers. The trick is that the pipeline expects to see the location to use to start loading the pipeline. Under hardware interrupts and software exceptions, the interrupt vector register is placed on the bus. When a return from interrupt instruction occurs, the address saved from the interrupt is placed on the pipeline. Fortunately, the syscop module is handed the current memory address in the `MEM_adr` input. By adding four to that value, the syscop puts out the instruction location just after the register switch instruction. This creates the desired result of flushing the pipeline and starting again at the next instruction.

What do you do with the bank selection register when an interrupt or exception occurs? While this could be made configurable, the design forces the selection register to 0 when an interrupt or exception is triggered. This provides the means to use bank 0 as a supervisory register

Listing 7—The miniMIPS processor is implemented here. The bank selection control signals are routed here and brought to the outside monitor. When the register bank module (U7_banc) is instantiated, it is given the number of register banks I want to create.

```
architecture rtl of minimips is
…
-- connexion to the register banks
-- Writing commands in the register banks
signal write_data : bus32;
signal write_adr : bus5;
signal write_GPR : std_logic;
signal write_SCP : std_logic;
-- Reading commands for Reading in the registers
signal read_adr1 : bus5;
signal read_adr2 : bus5;
signal bank_sel : bus8; -- Bank selection for which register set to use
signal read_data1_GPR : bus32;
signal read_data1_SCP : bus32;
signal read_data2_GPR : bus32;
signal read_data2_SCP : bus32;
…
begin
stop_pf <= DI_bra or bra_detect or alea;
genop <= bra_detect or EX_bra_confirm or DI_bra;
-- Monitor the bank_sel lines
bank_sel_mon <= bank_sel;
…
U7_banc : banc
generic map ( num_reg_banks )
port map(
        clock => clock,
        reset => reset,
        -- Register addresses to read
        reg_src1 => read_adr1,
        reg_src2 => read_adr2,
        bank_sel => bank_sel,
        -- Register address to write and its data
        reg_dest => write_adr,
        donnee => write_data,
        -- Write signal
        cmd_ecr => write_GPR,
        -- Bank outputs
        data_src1 => read_data1_GPR,
        data_src2 => read_data2_GPR
);
U8_syscop : syscop port map (
        clock => clock,
        reset => reset,
        -- Datas from the pipeline
        MEM_adr => MEM_adr,
        MEM_exc_cause => MEM_exc_cause,
        MEM_it_ok => MEM_it_ok,
        -- Hardware interruption
        it_mat => it_mat_clk,
        -- Interruption controls
        interrupt => interrupt,
        vecteur_it => vecteur_it,
        -- Writing request in register bank
        write_data => write_data,
        write_adr => write_adr,
        write_SCP => write_SCP,
        -- Reading request in register bank
        read_adr1 => read_adr1,
        read_adr2 => read_adr2,
        read_data1 => read_data1_SCP,
        read_data2 => read_data2_SCP,
        bank_sel => bank_sel
);
…
end rtl;
```

bank. All other tasks use the other banks (see Listing 4).

The pack_mips.vhd file, which is posted on the *Circuit Cellar* FTP site, provides a package of the interfaces to the various components. Therefore, it had to be changed to accommodate the new banc and syscop interfaces. Only those components are in Listing 5, along with the top-level miniMIPS component. The minimips.vhd file provides the top-level definition and behavior of the processor (see Listing 6 and Listing 7). The primary changes here involve creating the bank_sel bus for the banc and syscop modules and connecting the bus to those components. Most of the changes are purely internal to the CPU. In a system that has a processor that was already being used, no physical changes to the architecture would have been required.

Figure 3 shows the changes to the interface. In this case, I routed the bank_sel signals out of the processor so I could monitor them outside of the FPGA. I did this for debug purposes, but it takes up pins that might be needed for other purposes and makes this processor so it isn't a direct, drop-in replacement. The source code, which is posted on the *Circuit Cellar* FTP site, has this change, but you'll want to remove the change when debug is complete. Under normal circumstances, this interface would not be necessary. The basic system that was described earlier provided the basis for a system with the new processor.

HARDWARE

As I described earlier, the hardware design for the proof of concept consisted of the modified miniMIPS processor, together with the following components: an SRAM interface (unused by the software, it is used to trace signals), a UART (also not used by the software), an LED display to show which register bank is in use, and a four-digit, seven-segment LED display to show the current results of the thread's calculations.

The development board has a 50-MHz clock source, but I slowed the system

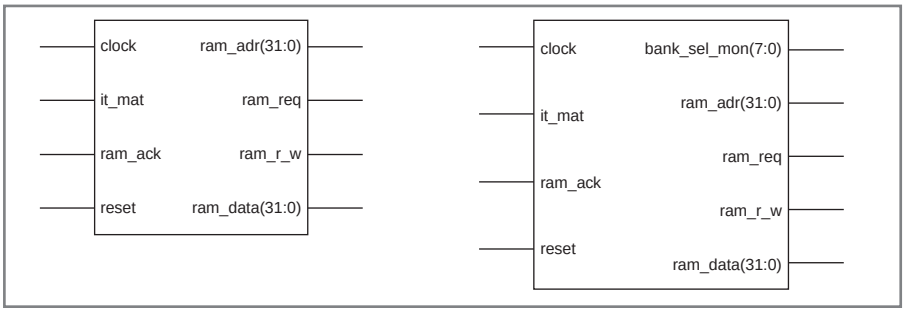


Figure 3—Under normal circumstances, the processor on the right would be the same as the one on the left. The monitor lines were added so I could see the action of the bank selection control lines. They can be removed in normal usage.

clock to almost 24 Hz so that I could see the behavior of the system.

For the Spartan-3 XC3S1000, I had a maximum of 7,680 slices available. Starting at one register bank, I built the system and examined the use of the part. Then, I incremented the register banks and regenerated. I was able to get eight register banks implemented on my device. Table 1 shows some of the results.

The other item of interest in the results is how the usage increased as the number of banks increased. Going from one bank to two increased the logic usage by 25%. Going to five banks took utilization up to almost 99%.

I generated bitmaps with up to eight register banks. However, the software did not perform reliably and there were many anomalies in the behavior of the system that I cannot explain without better debugging ability.

SOFTWARE

I had to write a program that would illustrate the concept as simply as possible because I did not have a debugger (either the original or the updated one) available for this processor. The maximum number of register banks I was able to use was five; therefore, I selected a program that would use all five banks to increment a counter by different amounts and display that counter on the seven-segment LED display.

The code is displayed in Listing 8. Lines 1 to 25 act as the initialization of the system. The `MTC0` instruction is used to switch the bank control register, thus selecting which register bank is in use. Let's review how each register bank is set up. `R1` is the main counter, which is initialized to 0 for all tasks. `R2` is the increment counter, which is added to `R1` to get a new total. For task 0, the count is 1. For task 1, the count is 2. Finally, I get to task 4, which has the increment count of 5. `R3` holds the number of the next task to be run, sort of like a linked list in registers. Each task, except task 4, holds the same number as its increment count. Task 4 needs to loop back to task 0, so 0 is placed in `R3`.

In line 28, I switch to task 0 (which was linked from task four), and the work begins. On line 29, the current counter is sent to the LED display, which is mapped to 0xFFFFFF80 in the address space. The `-128` is sign extended in the processor when the address is generated. On line 30, the task initializes a loop counter to 5, as each task will do its work five times before moving on to the next task. On line 31, the task's total counter, `R1`, is added to the increment count in `R2` and placed back in `R1`. The result is the output again to the LED display on line 32. On lines 33 and 34, the task's loop counter is decremented, and I compare it with `R0` to see if I have completed all of my work for this time period. If I haven't, I go back three instructions to line 31. Otherwise, I will fall through.

The task will have reached line 35 when all five times through its work are finished. So, it will jump to line 28, where the registers are switched and the work begins again. The register switch clears the pipeline, so the next three instructions still work; whereas if the processor hadn't cleared the pipeline, I would have ended up with a mixture of the prior set's inputs going into the current set's registers.

The assembler that came with the miniMIPS processor did not appear to have any documentation, so it was difficult to figure out how to write the assembly language to generate a program. Rather than having to work through that, and because I was going to need the resulting program in a ROM space in the design anyway, I decided to write a Perl program to "assemble" my source file into a VHDL source file. I won't go into the details of the Perl program here, but it is in the source code files posted on the *Circuit Cellar* FTP site. Note that not all of the miniMIPS instructions

Implementation	Original	1 register bank	5 register banks
Slice registers	1,814	1,854	5,832
Used as flip flops	1,780	1,820	5,798
Used as latches	34	34	34
Number of 4-input LUTs	4,659	4,344	8,468
Occupied slices	3,057	2,887	7,678
	~39%	~38%	~99%
Total four-input LUTs	4,731	4,422	8,790
Used as logic	4,659	4,344	8,712
Used as route-thru	78	78	78
Bonded IOBs	99	99	99

Table 1—This table gives you an idea of FPGA usage. The first column shows how the premodified miniMIPS processor used the FPGA. The second shows the modified miniMIPS with only one register bank. The last column shows a processor with five register banks.

Saelig

UNIQUE PRODUCTS & SUPPORT

www.saelig.com

Testgear

USB Bus Analyzers

Packet-Master™ - best value in USB1.1/2.0 analyzers and generators. Identify problems fast, see Host Commands, emulate host/device, etc.

USB12 (USB1.1) $699
USB480+ (USB1.1/2.0) $1199
USB500AG (USB1.1/2.0/Gen) $1399

2-ch 1GSa/s Scope

2-ch 1GSa/s (25GSa/s equiv.) 50/100 MHz scope. Competes head-to-head with high-end makes. auto-setup - easy to use! 1 Mpts memory depth - great for circuit debug.

DS1000E FreeCase! $595

Scope + Analyzer

25MHz 2-ch /16 logic scope and logic analyzer. 2000 wfm/s refresh rate. See waveforms and logic data simultaneously. Trigger on data or signal edge.

DS1022CD FreeCase! $699

60/100/120MHz AWG

60/100/120MHz USB 14-bit ARB w. USB RS-232, LAN/GPIB. 10 built-in wfms; 6 digit 200MHz frequency counter; opt. 16-ch gen. **DG3061A** $1895

DG3101A/DG3121A $2795/$3995

1/2GHz RF Generators

High accuracy/stability, wide range, low phase noise/leakage, serial control. Keypad/rotary setting. TGR1040: 10MHz-1GHz, FM int/ext TGR2050: 150kHz-2GHz AM/FM/PM

$2498/4558

EMC Spectrum Analyzer

Handheld Palm PC-based 2.7GHz Spectrum Analyzer. Multiple sweep settings – store wfms, set-ups, etc. Use WiFi PC for email, reports, calculations.

PSA2701T $1990

Color LCD Scope

Owon - 25MHz 2-ch + trigger standalone USB bench scope with 7.8" color LCD. Battery powered optional.

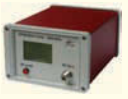
PDS5022S (25MHz) $325
PDS6062T (60MHz) $599

Low-Cost Scope

2-ch 40/100/200MS/s 8-bit scope range with 5/10/25MHz bandwidth and USB2.0 for fast screen updates. Inc. scope/FFT/logging software.

PS2203/4/5 from $297

RF Generator



High-res, extermely low-noise, portable RF generator. Outstanding signal purty (phase noise -125dBc/Hz at 1GHz, harmonics <-40dBc). Rapid 2ms power/frequency switching/settling. 9k-3.3GHz.

APSIN3000 $4999

16-Ch Logic Analyzer

Intuitive full-featured 16-ch 4MB 200MHz sampling memory. Sophistcated triggering. I²C, SPI, and UART monitoring. Includes clipleads, USB cable and software.

LAD-16128U $299

20MHz Scope

Winner!

PS5204 - 2-ch 250MHz Scope with 128 MS/s and real-time sampling of 1GS/s. Adv triggering. 250MHz spec. analyzer and 125MS/s waveform gen.

PS5203 32MS buffer $2151
PS5204 128MS buffer $3231

Waveform Generator

USB 2.0 high speed 16-bit digital pattern generator or arbitrary digital waveform generator. 100MB memory at up to 50 MSample/s.

WaveGen Xpress $699

6 in 1 Scope

6-in-1 USB scope adapter! 200kHz 2-ch 10-bit scope, 2-ch spectrum analyzer, 2-ch chart recorder, 16-ch 8MHz logic analyzer, 5-ch signal gen, 8-ch pattern gen. SPI/I2C/1-w/UART decoding.

PoScope with probes $199

Automotive Testing

Auto Diagnostics - Award-winning kits turn yours into vehicle-electrics diagnostic tool. **PS3423 kit** $2293

3-ch logger for 16-bit current/voltage/temp msmt. **KLARI-MOD MC** $1900

Mixed -Signal PC Oscilloscope

100 MHz Scope and Logic Analyzer lets you do complex triggering to find hard-to-get-at glitches, spikes, etc. Huge 4 or 8 MS buffer for deep data drilling and zooming. Optional built-in swept signal generator. 2 Analog channels x 10, 12, 14 bit with more than 60 dB dynamic range. 8 digital inputs for mixed signal display/trigger.

CS328A-4 (4MS Buffer) $1259
CS328A-8 (8MS Buffer) $1474
CS700A (signal generator) $299

Handheld Scope

20MHz or 60MHz rugged handheld USB scope with 3.8" color LCD. Built-in meter - great for your tool kit.

HDS1022MN (20MHz) $593
HDS2062M (60MHz) $699

Pen Scope

PS2105 - 25MHz USB powered scope-in-a-probe! Up to 100MS/s, 24kS buffer C/VB/Delphi/LabView/VEE drivers.

PS2105 (25MHz, 100MS/s) $369
PS2104 (10MHz, 50MS/s) $234

RF Testing

Economical, portable RF test enclosures, for initial compliance testing. **from $1295**

Shielding Tents with external frame, instant up/down for emissions-free room. **from $3500**

Multiparameter Loggers

Mini-logger with built-in temp/humidity/pressure/3-axis accel sensors. Customize msmt/record parameters; send data to PC after or during logging. Synchronize multiple loggers, merge data to single record

MSR145S from $468

SPI Bus Analyzer

SPI Bus Analyzer - SPI protocol exerciser and analyzer for standard SPI and non-standard 4-wire and 3-wire serial protocol interfaces up to 50 Mbps. PC control via free GUI 8PI Control Panel Software.

SPI Xpress $783

Serial-Ethernet Cable

Network your serial product easily without a PC using this 28" cable. Provides an instant IP address. RS-232 port compatible with most 10/100mpbs Ethernet hubs.

eCOV-110-P $89

TorqSense

Configurable, patented USB-output non-contact SAW digital rotary torque transducers with integral electronics. 1 - 10,000Nm versions. PC software indicates torque/speed/power/peak. Monitor/test/control drives. CALL

PSoC Starter

Get going quickly with PSoC visual design environment, inc. "Beginner's Guide to PSoC Express", CD media (devtools/compiler/projects/video-tutorial), PSoC demo board.

PSoC Starter $199

Misc

Keyboard Simulator

Easy-use USB board simulates standard keyboard/joystick, adds 55 I/O and 5 x 10-bit A/D inputs, 1 x 10-bit analog O/P. Settings/keyboard sequences stored on-board - no s/w needed.

PoKeys55T $81

Instant Ethernet

10/100 Ethernet controller on a chip! Fully hardwired TCP/IP core. No OS required. TCP offload improves system performance. PCB too!

W5100 (integrated PHY) $5.88
WIZ110SR $35

Ethernet-IO

Ether-IO - UDP/IP-controlled 24 digital I/O board 3 x 8-bit TTL ports each independently programmable. Connects to any TCP/IP Ethernet network.

Ether-IO 24 $119

.NET Board

Smallest (2.2" x2.2"!) lowest cost Microsoft's .NET Micro Framework development system. USB host & FAT file. Run C#; Visual Studio for programming. SD/USB memory, other I/F. **USBizi** $99

Wireless Analog

4-ch analog input wireless module 433/868/915MHz. Low power serial interface. Easy configure. 24 bit ID to avoids cross- talk. 2 digital output.

EmbedRF Module $59
EmbedRF DevKit $299

Temp/RH Sensors

UPSICAP™ - novel ambient sensors and modules accurately measure temp/RH. Amazing range/response: 0% to 100% RH, -90 to +85degC; 0.25s! Interchangeable without recalibration. **from $3 / $40**

USB to I2C

Devasys - Provides a simple "drop-in" solution for connecting your PC to 90kHz I²C + 20 I/O lines. Free software. Use multiple boards for more I²C/IO.

USB I2C/IO $89

Easy OLED Display

4D Systems - Compact, economical smart OLED with graphics for 'stand-alone' functionality to your project, drive from USB or RS232.

uOLED-96-G1 $89
uOLED-128-G1 $87

RF Modules

RF TX/RX Modules - simultaneously transmit composite video and stereo audio signals. 2.4 GHz ISM band. 4-ch RF, 5V operation

AWM630TX $16
AWM634RX $27

TinyOne Zigbee™

Complete RF solution for fast & reliable data transmission. Miniaturized OEM RF module ensures fast and easy integration into wireless applications.

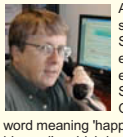
TinyOne Zigbee $58
B915FH-tiny/40-SMD-WA $63

1/2/4/8/16 x RS232

Add COMports via your PC's USB Port. **USB-COM-S** $20

USB-2COM $48
USB-4COM $105
USB-8COM $195
USB-16COM-RM (rackmount) $459

Saelig



Alan Lowne says: "I started Saelig as an engineer - for engineers! Saelig - an Olde English word meaning 'happy, prosperous, blessed' - which is what I want for my customers, my staff, and myself! See my blog for behind-the-scenes company details, business hints, kudos, etc. "

Saelig.com
unique electronics
888-75SAELIG info@saelig.com

Above are some of our best selling, unique, time saving products see our website for 100s more: volt/temp data loggers, wireless boards, LCD display kits, Ethernet/IO, USB/RS232/485, USB OTG, instant Ethernet serial, CAN/LINbus, USB cables/extenders, line testers, logic analyzers, color sensors, motion controllers, eng. software, wireless boards, SMD adapters, I2C adapters, GPS loggers, automotive testing, security dongles, video motion detectors, crystals/oscillators, custom switches, barcode scanners, robots, DSP filters, PLCs, Remote MP3 players, etc. FREE STARBUCKS card with a $250 order! Check www.saelig.com often for special offers, bargains, business hints, blog, etc.

are supported.

DESIGN LIMITATIONS

The design has a few limitations that need to be addressed before it can be used in a full system intended for production. Interrupts and exceptions currently trigger the register bank selection to bank 0. This was deliberate to ensure that a guaranteed register bank is used when these events occur. However, the previous register value is not preserved. Because the state of the system when the interrupt occurred is not truly preserved, the previous state has to be restored prior to returning from the interrupt. One possible solution to this problem would be to have a separate portion of the BANKSEL register devoted to what register is used for interrupts and exceptions. The processor knows when interrupts occur, so it would be easy for it to switch to that portion of the register and not modify the other register portion.

The miniMIPS does not appear to be designed to be in user versus supervisory modes and thus restrict certain instructions from executing in user mode. This means that a hostile piece of code would be able to switch to another thread's registers and change them, including the link return register R31. This is not a particularly desirable feature, but it may not be that much of an issue in certain embedded environments.

You may want to reduce logic usage by implementing the registers in either distributed RAM or block RAM as dual-port memory. This might increase the number of registers that can be implemented, but it may not be able to keep single-cycle accesses required for the pipeline implementation.

THE FUTURE

This project is a good starting point for expanding current soft-core processors and giving them more flexibility. Doing so enables real-time systems to meet their timing requirements. By removing or reducing the need to save and restore registers, you can reduce the context switch time. You will save more time for processing.

Listing 8—This is a simple program to exercise the various register banks. It's intended for a processor with five banks of registers. The R1 register in each one is incremented, in turn, by 1, 2, 3, 4, and 5.

```
Line program.asm
1   # Switch to task 0
2   MTC0 R0, R16
3   ADD R1, R0, R0
4   ADDI R2, R0, 1
5   ADDI R3, R0, 1
6   # Switch to task 1
7   MTC0 R3, R16
8   ADD R1, R0, R0
9   ADDI R2, R0, 2
10  ADDI R3, R0, 2
11  # Switch to task 2
12  MTC0 R3, R16
13  ADD R1, R0, R0
14  ADDI R2, R0, 3
15  ADDI R3, R0, 3
16  # Switch to task 3
17  MTC0 R3, R16
18  ADD R1, R0, R0
19  ADDI R2, R0, 4
20  ADDI R3, R0, 4
21  # Switch to task 4
22  MTC0 R3, R16
23  ADD R1, R0, R0
24  ADDI R2, R0, 5
25  ADD R3, R0, R0
26
27  # Switch to task in R3
28  MTC0 R3, R16
29  SW R1, R0, -128 # Stores contents to LED Display
30  ADDI R4, R0, 5
31  ADD R1, R1, R2
32  SW R1, R0, -128 # Stores contents to LED Display
33  ADDI R4, R4, -1
34  BNE R4, R0, -3
35  J 20
```

Larry Standage (lstandage@gmail.com) holds an M.S. in embedded engineering and a B.S.E. in computer systems engineering from Arizona State University. He has been working on and off in the embedded technology industry for the last 16 years. In his spare time between being a father to five children and teaching high school kids in a religious setting, Larry likes to play chess, read a book or two, and catch up on shows stored on his TiVo.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/223.

RESOURCES

Digilent, Inc., "S3 Starter Board Schematics," www.digilentinc.com/Data/Products/S3BOARD/S3BOARD-sch.pdf.

J. Frenzel, "MIPS Instruction Reference," University of Idaho, Department of Electrical and Computer Engineering.

miniMIPS Overview, www.opencores.org/projects.cgi/web/minimips/overview.

Xilinx, Inc., "Spartan-3 FPGA Family Data Sheet," DS099, 2008.

———, "Spartan-3 Starter Kit Board User Guide," UG130, 2005.

SOURCE

S3BOARD Spartan-3 Development board and Spartan-3 XC3S1000 starter board
Digilent, Inc. | www.digilentinc.com

ICbank
Gateway to Electronics

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

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

Sweet!

Introducing the MiniCore™ Series of Networking Modules

Smaller than a sugar packet, the Rabbit® MiniCore series of easy-to-use, ultra-compact, and low-cost networking modules come in several pin-compatible flavors. Optimized for real-time control, communications and networking applications such as energy management and intelligent building automation, MiniCore will surely add sweetness to your design.

- **Wireless and wired interfaces**
- **Ultra-compact form factor**
- **Low-profile for design flexibility**
- **Priced for volume applications**

MiniCore Module Development Kits

From **$49** Limited time offer.

Buy now at: **1.888.411.7228** • **rabbitwirelesskits.com**

1.888.411.7228
rabbitwirelesskits.com
2900 Spafford Street, Davis, CA 95618

RABBIT™

09009

by Steve Hendrix

Power Switching in the Real World (Part 2)

Practical Switching Arrangements

Steve continues his explanation of solid-state power switching by describing how to use a logic design to control real-world power. You can use his practical switching arrangements in your upcoming designs.

Last month, I introduced the topic of solid-state power switching. I covered safety precautions, presented some rules of thumb, provided tips and tricks for safely dealing with high-power devices, and described some of the theory behind the actual power-switching devices (*Circuit Cellar* 222, 2009). This month, I will build on that foundation by discussing how to safely use your brilliant logic design to control real-world power.

If you did not read the first part of this series, please go back and at least read the safety warnings. I get rather jaded by all of those lawyer-induced safety warnings, but here you could be dealing with enough power to do significant damage, so a bit more caution is necessary.

DRIVER CIRCUITS

There are almost as many ways to control a FET as there are engineers. Each method works well, as long as certain conditions are met, but none can handle every possible case. In this article, I'll cover several that I've found to be particularly useful in my own work.

An ideal FET conducts no current with no gate voltage and has no resistance when the gate voltage (always specified relative to the source) is above a certain threshold. Readily available FETs approach this ideal, and have improved greatly in the past 10 or 20 years. There are FETs available that require only a couple of volts to fully turn on. These newer FETs allow the "obvious" drive method, which was a total folly a few years back, to actually work quite handily in some cases. Figure 1 shows a direct and dirt-simple drive arrangement that enables a microcontroller to turn a FET on or off. Assuming the microcontroller ground is the same as the FET's power ground, a modern microcontroller running at 5 V can readily drive the FET gate to almost 5 V. There are many FETs available that turn on fully with a gate voltage of 3 to 4 V. But don't be fooled by the "gate threshold voltage." That number is just a rough approximation of the gate voltage at which the FET begins to turn on. It says little or nothing about the gate voltage required to turn the FET fully on. With some of the newest FETs, even a microcontroller running at 3 V can fully turn on the FET, but you must read the datasheets carefully.

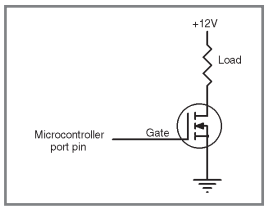


Figure 1—This method is simple and inexpensive, but it requires that the FET can turn fully on at the voltage supplied by the microcontroller. It also requires the microcontroller port pin to provide sufficient current drive to switch cleanly.

The robust drive capability of modern microcontroller output ports can easily lull you into complacency. Commonly available microcontrollers can source or sink tens of milliamps when they switch. But why do you care about their current capability, anyway? Isn't that FET gate high-impedance, with leakage currents measured in nanoamps or picoamps? Well, yes and no. In the static case, there's little or no current in or out of the gate. But the situation while you are switching looks a bit different. A FET gate has a significant amount of capacitance—and the bigger the FET (especially in terms of current handling capacity), the more capacitance. With limited current available to charge or discharge the capacitance, the time to get from "fully off" to "fully on" or vice versa can be significant. A typical high-power FET has a gate capacitance that is characterized by a graph similar to Figure 2. To change the gate voltage, you must add or remove charge by moving along the horizontal axis. If you have 1 mA of drive current available, you can move along this axis by 1 nC per microsecond. Thus, you would require 200 µs to move fully across the graph. But what's the hurry?

The lower-left region of the graph shows a single steady slope for gate voltage versus charge. It corresponds to the "off" state of the FET—no current, therefore, no power dissipation. The upper-right part of the graph shows three parallel lines, which correspond to the "on" state of the FET—almost no resistance, therefore, almost no voltage, and again almost no power dissipation. It's

the middle section, with the horizontal line, that gives you trouble. The horizontal line corresponds to the "partially on" state of the FET. Somewhere in the middle of that line, the FET dissipates maximum power. To minimize the amount of heat generated in the FET, you must transit this region as quickly as possible. If you were switching 5 V at 1 mA, the amount of heat would be negligible even for a small FET, unless you were trying to make millions or billions of switching cycles per second. Instead, if you are switching 100 V at 100 A, the peak power is at the halfway point, as I showed above, with 50 V and 50 A, and you dissipate 2,500 W in the FET at the middle of the transition between off and on (going either direction). When I failed to fully appreciate the significance of this middle region and built a circuit with a wimpy gate drive for a big FET, the first couple of switching cycles turned those fancy FETs into warm little puddles of glass. A power FET datasheet typically states a maximum number of joules (watt/seconds) that the FET can absorb within a time that is too short to reach a steady-state temperature or even to conduct the heat out of the FET. This is a rough but conservative rule of thumb to compute the energy in joules for one switching half-cycle (switching the FET on or off):

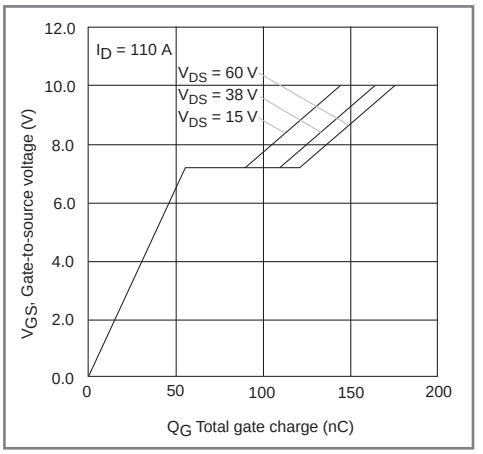


Figure 2—In the lower-left portion of the graph, the FET is completely off and the gate voltage changes linearly as you add or remove electrons. In the upper right, the FET is on and gate voltage also changes linearly. But in the center flat section, the FET is partially on and dissipating significant power while you add or remove electrons from the gate.

$$\frac{V \times I}{4 \times T} \quad [1]$$

V is the voltage being switched. I is the current being switched. T is the amount of time spent in the "linear region." When switching tens or hundreds of amps and volts, you need to keep the time under about 1 µs. To do so, you must be able to add or remove charge more quickly. You must have higher current available to switch the gate. To move 200 nC of charge in 1 µs, you need a gate drive of 200 mA. That's a bit beyond the capability of a typical microcontroller port pin.

Note that paralleling multiple devices, as described in the first part of this article series, compounds this problem. With multiple gates in parallel, their capacitance is cumulative. With several big FETs in parallel, it's not hard to come up with 0.1 µF or more of total capacitance. Because you would use parallel FETs precisely for a situation where high power is switched, you must be extra careful not to allow the gates to linger in that "partially on" region for a significant amount of time. So, you must provide a robust gate driver.

The other challenge you can see in Figure 1 is that most microcontrollers will not be happy running from a 10-V supply, which would be required to cleanly pull the gate up to the full-on region with direct drive. But if the system has only a 5-V supply rail, what are you to do?

CHARGE PUMPS

There are a number of commercially available integrated circuits that provide a level shift and buffering to yield the correct gate drive voltages for a FET. Unfortunately, I learned, to my chagrin, that some have additional functionality, such as a low-voltage cutout function that cuts off the entire gate drive if the supply voltage drops below about 9 V. I had no control over that, even though the particular FET I was using would have been perfectly happy at 6 V. There can also be issues of availability for such single-source parts. These are probably good parts, but I haven't had much reason to use them. In any case, the manufacturers' datasheets show most of the application details needed. The International Rectifier IR2122 driver is a good example of such a chip. It provides the ability to even use an N-channel FET for high-side switching.

You also might want to consider using one of the many RS-232 converter chips as a gate driver. They provide charge pumps, level shifting, and buffering that will neatly drive many FETs. When powered from 5 V, many of these chips provide an output that will swing from 10 V (solidly guaranteed to be fully on for most FETs) to –10 V (guaranteed to be off for any N-channel FET I've ever seen). I recently used a Maxim Integrate Products MAX3238, which has five transmit channels, to provide two outgoing RS-232 signals plus the gate drive for three FETs. Such chips are readily available from multiple sources and in several different numbers of channels. Many of them have been around for a long time and are likely to remain readily available for the foreseeable future. The only caveat would be that the amount of current available for gate drive may be marginal to guarantee the snappy switching of large FETs.

OPTOCOUPLING

System constraints may dictate high-side switching, while the control logic is referenced to the low side (ground). Particularly where the high side may have a variable voltage, it can be difficult to maintain tight control of the gate voltage when that gate voltage must be referenced to a moving target. Optocoupling is a time-honored method of transferring a logic signal from one reference to another.

Figure 3 shows the basic connections to use optocoupling to drive a P-channel FET on the high side of the load, where the control logic is referenced to the low side of the load, or not referenced to the load at all! Because a typical optocoupler provides anywhere from 250 to 1,500 V of isolation, you can use this method to drive loads that may be hundreds of volts away from your control logic, as long as you're careful about conductor clearances, and so on, throughout the system. The Enable logic signal simply turns on the photodiode in the optocoupler when high. The light then turns on the output phototransistor, pulling the FET gate 15 V below its source, which is solidly on for almost any FET. When the Enable signal is low (or not driven), the optocoupler turns off and the 1-kΩ resistor pulls the FET gate back up to equal the source, turning the FET and then the load off.

The Optional Buffer may be required for this method when using a large FET to switch a heavy load. As I discussed in the

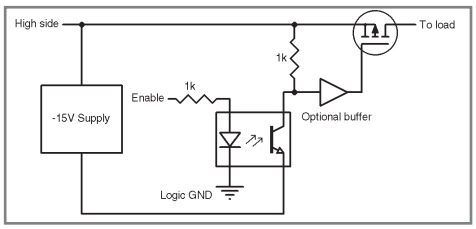


Figure 3—The optocoupler allows for control of the FET from a microcontroller or other control whose power supply rails are at a large voltage offset from the FET source terminal.

Driver Circuits section, you may need to provide more than a couple of milliamps of current to drive the gate through the region of partial conduction and its attendant high-power dissipation. This buffer would of course use the –15-V supply as its reference.

CAPACITIVE COUPLING

In a recent project, I needed to rapidly switch an output terminal between ground and a steady but adjustable positive voltage. The high-side voltage could vary from 100 mV to 4 V, and the power budget made a charge pump impractical. Instead, I used a P-channel FET to switch the high side. The FET I chose to meet the current, voltage, and size requirements would guarantee full turn-on with only about 2 V of gate voltage, so the 5-V logic output of the microcontroller could directly drive the gate when the high-side voltage was 4 V. With the logic signal at 5 V, the gate would see negative 1 V (relative to its source terminal, which is the relevant reference here) and the FET would be solidly off. With the logic signal at 0 V, the gate would see 4 V and would thus be solidly on.

But what about the case where the high-side voltage is only 100 mV? With the logic signal at 5 V, the FET is most assuredly off, but 0 V will provide 100 mV of gate drive, not even enough to think about turning the FET on. Because this application never required static switching, but included a minimum 0.1-Hz switching frequency specification, capacitive coupling was perfect to achieve clean switching with a minimal parts cost.

Figure 4 shows the deceptively simple solution I used. The low-side switching used a simple N-channel FET Q2 with the source at ground and the gate switched between 0 and 5 V, which was easy enough. The high-side drive was provided with a P-channel FET and controlled by a separate logic output of the microcontroller because the system also required that the output be floating at times. So, let's see what happens to that P-channel FET when we switch the *Pos signal.

Starting with *Pos in any steady state, the Q1 gate will be pulled to the source by the leakage in D1. (My actual circuit included pads for a high-value resistor as insurance, but the resistor proved unnecessary.) If *Pos happens to start in the low state, capacitor C1 charges up to V_{HIGH}. When *Pos switches high, the capacitor voltage is added to the 5-V logic-high level, and the FET gate tries to go to 5 V + V_{HIGH}. D1, however, turns on and limits the gate voltage to V_{HIGH} plus a diode drop, causing C1 to charge the opposite direction to 5 V – V_{HIGH}. On the next cycle, when *Pos goes low, the charge on the capacitor takes the gate to V_{HIGH} – 5 V, which is (just coincidentally, of course) exactly where you want the gate to be for a full turn-on. If for any reason the capacitor is charged up to a higher voltage, any excess beyond 5.1 V below V_{HIGH} will bleed off through D1. The gate sees a square wave with 5 V of swing, neatly constrained so that the upper voltage is just a diode drop above the source, and the lower voltage is 5 V minus a diode drop below the source. Thus, you achieve perfect on/off switching, with all of the dynamic current drive the microcontroller pin can provide, and nearly no static leakage.

CURRENT SENSING

When you switch high-power levels, you often need to sense how much power is actually being delivered. Voltage measurements are relatively easy. I have frequently encountered a need to measure the actual current in one or more of the power leads, and that's not always so easy. In some cases, this measurement supported a hardware overcurrent shut-off mechanism. In other cases, there was a system requirement to know the current, or the measurement was used for diagnostics. When I'm dealing in currents at tens or hundreds of amps, I usually want some way of knowing just how *much* current I'm dealing with!

In theory, current measurement is simple. Just insert a small resistor in the line and measure the voltage across it, right? If the current is small and the resistor is in the "ground" lead, it's easy enough. But if the current is high, it gets harder. The resistor value must be small enough to avoid dissipating an unacceptable amount of power, but large enough to provide a usable signal. If the resistor can be on the low side of the load, a simple op-amp circuit easily provides any needed gain to scale the voltage to a usable range, especially with modern op-amps where the inputs are allowed to go all the way to the negative rail and beyond.

Now look at the case where you must measure in the high-side lead, and both the voltage and current are large. I recently had a requirement to measure current in the high side, where the current could be as high as 150 A and the voltage could go as high as 150 V. Size and thermal constraints dictated that the sense resistor dissipate no more than about 1 W. To guarantee that the power dissipation in the resistor stays under 1 W, you must keep the voltage across the resistor under about 7 mV.

Accurately measuring a signal with a full-scale range of 7 mV, riding on a common-mode voltage of up to 150 V, to 10-bit accuracy, calls for a common-mode rejection ratio around 120 dB! Failing to do the calculations in advance nor even fully appreciating the problem (except in hindsight), I chose to use the tried-and-true Texas Instruments INA117 in-amp, which can directly handle up to 200 V of common-mode voltage. The INA117 takes a differential voltage, such as that across a current-sensing resistor in the high side, and references that voltage down to a logic ground rail or other selected low-voltage reference. An ordinary op-amp circuit then provides any needed gain.

Depending on the exact conditions, the INA117 provides around 70 to 80 dB of common-mode rejection, versus the needed 120 dB. With inadequate common-mode rejection, the high- side voltage comes through as a significant contribution to the measured current value. This method that worked so well in past applications with less severe constraints, worked quite poorly when stretched to high current and voltage. How can you achieve a better current measurement and still reference the

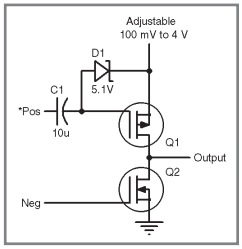


Figure 4—C1 isolates the gate from the microcontroller at DC, while allowing the full 5-V swing of the control signal through to the gate. D1 clamps the gate drive at both ends, ensuring that the gate always swings between the source terminal and 5 V below the source terminal.

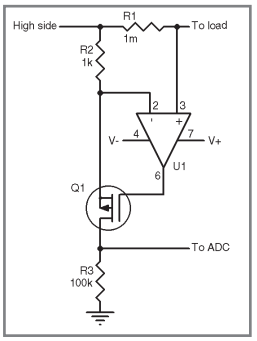


Figure 5—Op-amp U1 drives FET Q1 so its closed/loop action forces the voltage across R2 to match the voltage across R1. The current through the FET is then equal to the load current times the ratio of R1 to R2. R3 converts this current to a ground-referenced voltage for measurement with an ADC or for other use in the control circuit.

measurement to your system ground so you can readily measure it with an ADC or apply it to a comparator for a rapid overcurrent shutdown mechanism?

Linear Technology's LTC6102 provides one clever way to measure high-side current. The LTC6102HV version can measure small sense voltages riding as high as 100 V above the negative rail ("ground"). Figure 5 shows a simplified version of the measurement method. This measurement method can be implemented with discrete components for higher voltages, but the LTC6102 has some nice features. Start with a small-value sense resistor R1. Using 1 mΩ here provides a differential voltage of 1 mV per amp of current. The closed-loop action of op-amp U1 drives the gate of Q1 to force the voltage across R2 to exactly match that across R1. Because R2 is 1 million times larger than R1, the required current through R2 is exactly one one-millionth that through R1, giving you a nearly perfect current divider. That divided current must then flow through R3 as well, so the voltage across R3 is directly proportional to the current. The end result is a voltage at the ADC:

$$\frac{I_{LOAD} \times R1 \times R3}{R2} \quad [2]$$

It is neatly referenced to the ground rail.

Because the comparisons at the input of the op-amp are done directly at the high-side voltage, common-mode rejection is almost a nonissue. You have only the tolerances of the three resistors, plus the input offset of the op-amp, to set the accuracy of the final measurement. In the real world, of course, you may need to add a capacitor across R3 to provide low-pass filtering, and you may need some resistance in the line to the ADC for protection against out-of-range voltages or other fault conditions. All things considered, this circuit is about as simple and foolproof as they get for measuring current in the high-side lead. Several other such chips have recently become available using a similar measurement method, but the LTC6102 has extremely low input offset voltage and a high common mode voltage range.

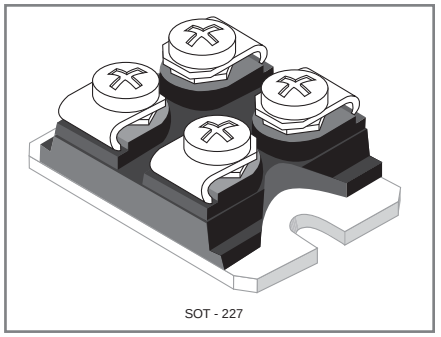


Figure 6—This SOT-227 package can provide mechanical support for a PCB mounted to its screw terminals, while its base is mounted to a thermally conductive enclosure for heat removal.

CONSTRUCTION HINTS

I sometimes cobble up a quick test for ordinary electronic circuits by skywiring resistors or capacitors with a quick solder joint or just twisted leads. For surface-mount, I'll build a tipi out of two or three components, or other rather ugly tricks to get a quick test going. When you deal with large currents, however, such shortcuts aren't always a good idea.

Large power FETs and IGBTs come in big beefy packages with terminals sized to carry the heavy currents they can handle. The SOT-227 package in Figure 6 is a good example. So how do you neatly make connections to such a beast? One technique I like is to build the PCB to directly fit on the screw terminals, with the SOT-227 providing the mechanical mounting of the (small) PCB to the case. The external wires carrying the heavy currents then connect directly onto those same screws using ring terminals crimped to wires of an appropriate size. Thus, the circuit board does not have to carry the really heavy currents, yet I've got easy access in my circuit to all three terminals of the FET or IGBT.

At high currents, just wrapping a wire under a screw head isn't going to provide an adequate connection. Most major electronics distributors can supply a variety of crimp terminals and a tool to attach them to wires. For quick turnaround without paying shipping charges (and sometimes lower prices as well), I find that my local home center carries hardware for AWG#6 and smaller. The local automotive parts store has similar and sometimes even heavier parts. Indeed, an automotive store can even supply you with ready-made cables of almost any length and gauge if you aren't comfortable with crimping them yourself or don't have the tools to do so. Either screw terminals or quick-disconnect blade-type terminals are suitable for a wide range of currents and voltages, and make field changes relatively easy. If you want to solder these high-power connections, carefully consider the heatsinking properties of all that heavy copper—a great thing during operation, but quite a challenge to that little 15-W SMT soldering iron!

Finally, before you actually apply power, take a good look around for ways that Murphy's Law can bite you. Consider with special care all of the possible paths between the terminals of a muscular current source like a big battery. You must also consider how you are going to quickly interrupt the current and get your system into a guaranteed safe state if (when?) something goes wrong and a component begins to release the magic white smoke in copious amounts. For the worst case, consider the locations of the fire extinguishers and the exits. Could your creation block access to either? All my careful design work should prevent the need for such precautions—but my pilot side always insists on looking for the worst case!

Steve Hendrix (SteveHx@HxEngineering.com), P.E., CFIAI, lives with his wife Kathy and five children, ages four through 22, next to Cuyahoga Valley National Park in Ohio. When he isn't busy hiking with them in the Park, leading Boy Scouts, foster parenting, or ringing handbells, he runs Hx Engineering, LLC. Steve has designed software and hardware for various projects: Space Station Freedom, vibration analysis, IEEE-488 bus hardware, biomedical sensing (including wireless EEG, EKG, and EMG recorders), industrial instrumentation, and automatic phototherapy equipment.

SOURCES

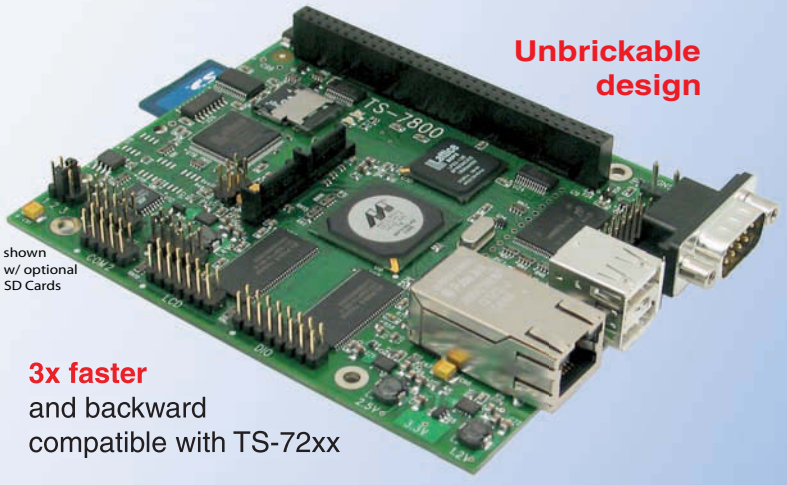
IR2122 Driver
International Rectifier | www.irf.com

LTC6102 Current sense amplifier
Linear Technology Corp. | www.linear.com

INA117 In-amp
Texas Instruments, Inc. | www.ti.com

Embedded Single Board Computers

High-End Performance with Embedded Ruggedness



shown w/ optional SD Cards

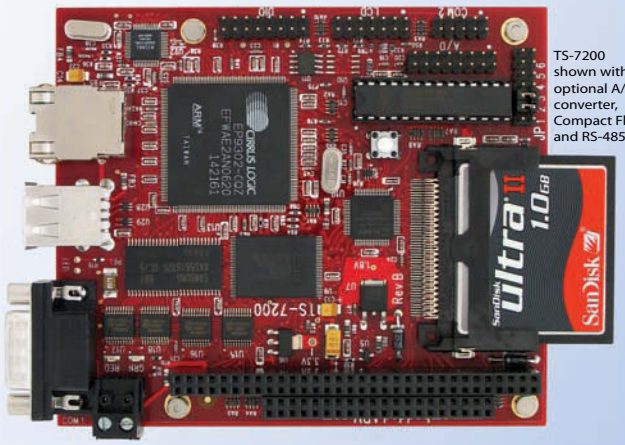
Unbrickable design

3x faster and backward compatible with TS-72xx

TS-7800
500 MHz ARM9

- Low power - 4W@5V — **$229** qty 100
- 128MB DDR RAM
- 512MB high-speed (17MB/sec) onboard Flash — **$269** qty 1
- 12K LUT programmable FPGA
- Internal PCI Bus, PC/104 connector
- 2 host USB 2.0 480 Mbps
- Gigabit ethernet
- 2 SD sockets
- 10 serial ports
- 110 GPIO
- 5 ADC (10-bit)
- 2 SATA ports
- Sleep mode uses 200 microamps
- Boots Linux in < 2 seconds
- Linux 2.6 and Debian by default

Low Price, Low Power, High Reliability using Linux development tools



TS-7200 shown with optional A/D converter, Compact Flash and RS-485

- options include:
  onboard temperature sensor, A/D Converter 8 channel 12 bit, Extended Temperature, Battery Backed Real Time Clock, USB Flash, USB WiFi

200 MHz ARM9
Power as low as 1/4 Watt

- 8 boards, over 2000 configurations — as low as **$99** qty 100
- Fanless, no heat sink
- SDRAM - up to 128MB — **$129** qty 1
- Flash - up to 128MB onboard
- 10/100 Ethernet - up to 2
- DIO lines - up to 55
- 2 USB ports
- COM ports- up to 10
- Programmable FPGAs
- SD card option
- VGA video
- LCD ready
- Linux, Real Time extension, Debian

- Over 20 years in business
- Open Source Vision
- Never discontinued a product
- Engineers on Tech Support
- Custom configurations and designs w/ excellent pricing and turn-around time
- Most products stocked and available for next day shipping

Design your solution with one of our engineers (480) 837-5200

Featured Products and PC/104 Peripherals

NEW!

$449 qty 1

TS-TPC-7390

200MHz ARM9 Touch Panel Computer

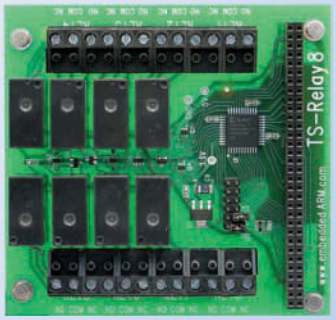
- Low Power, Industrial Quality Design
- Mountable aluminum frame
- 64MB SDRAM (128MB opt)
- 512MB Flash w/ Debian Linux
- Programmable FPGA- 5K LUT
- 7" Color TFT-LCD Touch-Screen
- 800x480 customizable video core
- Dedicated framebuffer- 8MB RAM
- Audio codec with speaker
- Boots Linux 2.6 in about 1 second
- Unbrickable, boots from SD or NAND
- Runs X Windows GUI applications
- Runs Eclipse IDE out-of-the-box

NEW!

TS-RELAY8

Eight Software Controlled Relays

- 8 SPDT relays
- Up to 277 VAC @ 5A
- Up to 30 VDC@ 5A
- Software controlled
- 40mA draw per coil
- I/O jumpers



$89 qty 1

NEW!

TS-ADC16

ADC, DAC and Digital I/O

- 16 16-bit ADCs
- Up to 100Ksps (10us)
- 1KB ADC RAM-FIFO
- 4 ADC voltage ranges
- Prog. pacing clock
- Externally triggered
- 4 inputs, 1 output
- 4 12-bit DACs
- 4 16 bit counters

$169 qty 1

see our website for x86 SBCs, peripherals and option details

Technologic Systems

We use our stuff.

Visit our TS-7800 powered website at

www.embeddedARM.com

by Tom Cantrell

Here Come the Wadgets

As the catalog of wireless hardware options grows, so does the range of potential applications. This month, Tom introduces you to three new wireless gadgets, or "wadgets." There's something for everyone, from a mini-me Java machine to a wireless wire and everything in between.

It's amazing to think that it's been a little more than 100 years since inventors like Marconi and Tesla pioneered wireless communication. What would the twentieth century have been like without it? No radio. No TV. No cell phone. No Wi-Fi. For better (e.g., moon shot) or worse (e.g., reality TV), wireless communication has had a profound impact.

So where's wireless going in the twenty-first century? Everywhere. You know the buzz words: *wireless sensor networks*, *smart dust*, *pervasive computing*, *Internet of things*. I'll add my own word to the list: *wadgets*, which is shorthand for wireless gadgets.

The wadget factories are running at full throttle judging by the ever-growing catalog of embedded wireless solutions. Designers have lots of options to choose from to find the best fit for a particular application. There is no better way to highlight the range of answers to the twenty-first-century wireless question than checking out three new wadgets with quite different takes on the subject.

Photo 1—The Sentilla "JCreate" motes pack a Java jolt into a small package. Their Perk kit comes with a USB plug-in radio for the PC and two motes. That's a step up from kits that come with just one mote and thus can't exercise features used in larger scale (i.e., n greater than 1) networks.

PERK UP

Imagine a day in the future when Moore's Law could make it possible

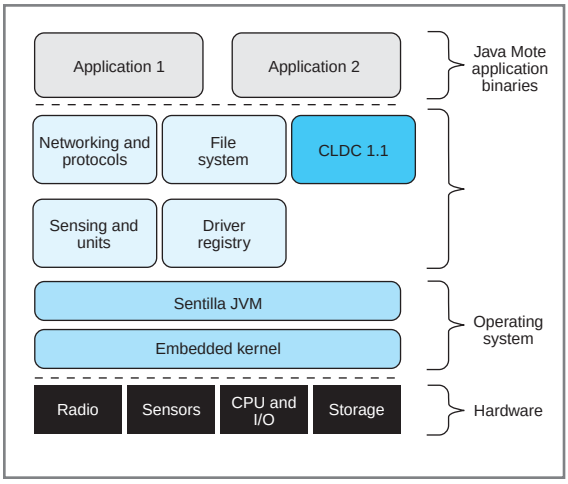


Figure 1—Running big-iron Java software on a little-iron processor (i.e., the Texas Instruments MSP430 MCU) is key to Sentilla's vision of "Pervasive Computing."

to use PCs-on-a-chip as nodes in a wireless sensor network. The beauty of such a scheme would be a unified programming environment for "computing" and "embedded" applications.

Sentilla is one of the many wireless sensor network outfits with roots in the "Smart Dust" initiative at the University of California, Berkeley. Formerly known as module supplier Moteiv, their name change reflects a new direction aspiring to a one-world unified programming nirvana. However, instead of trying to cram a PC into a wireless sensor module, Sentilla uses Java to bridge the gap. "Perk" is the apt name for Sentilla's "Pervasive Computing Kit" (see Photo 1).

From a hardware perspective, the Sentilla motes (called JCreates) pair Texas Instruments's MSP430 flash memory MCU and CC2420 802.15.4 radio in a way that's quite typical of the breed. But while the hardware may be standard-issue, it's the fact that Sentilla manages to cram a Java virtual machine (JVM) inside that makes the difference (see Figure 1).

As the force behind Java, it's no surprise that Sun Microsystems advocates the same "one world" approach of using Java across the board. Their "SunSpot" kit, which also consists of two nodes and a gateway, is quite similar to the Perk kit in concept. But the SunSpot nodes need a lot more silicon (i.e., 32-bit ARM processor, over 4 MB of memory), a difference further reflected in the price of the evaluation kits ($750 for the SunSpot kit versus just $199 for the Perk).

Right on the first page of Sentilla's release notes it says Perk is designed for "experienced Java programmers." But that doesn't mean you have to be a Java expert to kick the tires. Indeed, Sentilla has gone out of their way to deliver a quick and easy out-of-box experience. After installing the Sentilla software on your PC (see Photo 2), it takes only a few mouse clicks to get the network on the air with the motes running the Java equivalent of "Hello World" (blinking an LED).

The demo programs head onward and upward by showing how to program a mote to perform hardware I/O functions. For example, the motes each have a built-in accelerometer, so there's a Java program that turns a mote into a level, with the LEDs "falling" in the direction it is tilted. Other tutorials show how to send and receive messages across the network between the motes and PC gateway, which is where the "one world" concept (i.e., both the mote and the PC are programmed using the same Java tool chain) really starts to come into its own.

Another good source of examples is the Perk application showcase Sentilla set up at http://labs.sentilla.com. The programs you'll find there further illustrate the benefits of the unified mote and PC programming model. For instance, one program displays the number of new e-mails in your PC inbox on the mote's LEDs. Another has the mote and its accelerometers acting as the joystick for a pong game running on the PC. A third implements a data logger that samples mote data and displays the results as line charts on the PC display. These applications all take advantage of code created by others in the Java community and are freely available for you to reuse in your own application. I always say, "The smartest way to write a program is to use the one someone else has already written!"

YEAH, BUT...

Sentilla does a good job of getting inexperienced Java programmers off to a good start. And it's clear they've got a compelling vision of pervasive computing with everything unified under the umbrella of Java.

A visionary concept is one thing, but there's also the matter of turning it into a practical implementation, and it's here that I have some concerns. There is no doubt that some of them are just typical rev.0 issues that

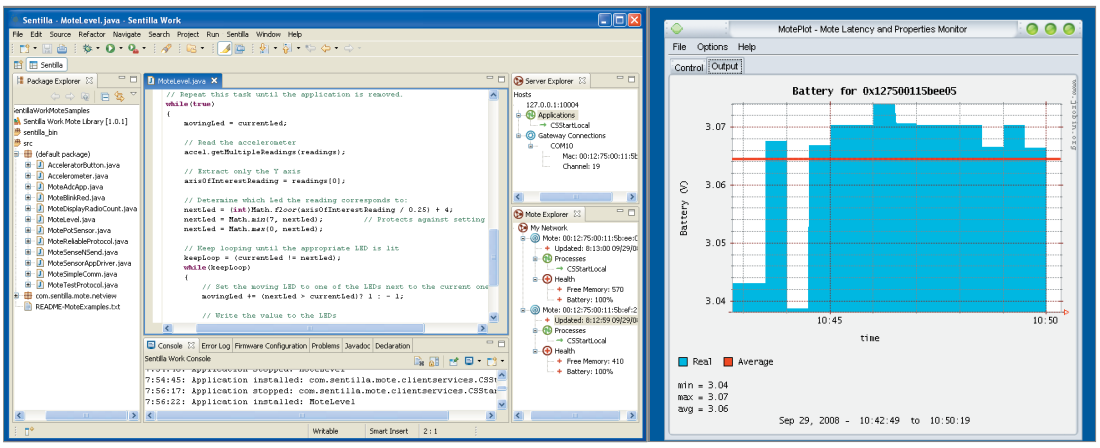


Photo 2—Other than the fact that it's Java rather than "C," this screenshot (a) of the Sentilla development environment should look familiar. It's the usual (i.e., Eclipse-based) IDE like you're used to seeing. What is special is that you use the same tool chain to write programs for the PC, such as this battery-monitoring application (b), and the motes.

can be improved in future releases. But others may fundamentally be part and parcel of the one-world Java vision that has big-iron software shrink-wrapped onto a small machine. At this point, I'm not exactly sure to what degree a particular issue is just a rev.0 concern or is indeed intrinsic to the concept, so let me just lay them out for you.

The documentation states that for now motes can host only a single program and it must be the same for every mote in the network. This sounds worse than it is because obviously the single program can incorporate a variety of mote application functions. Another possible workaround would have the PC hosting multiple networks (i.e., each with its own application) using different radio channels. Of course, there's always the option of swapping in a new application, although reloading the entire network may take minutes or so depending on the number of motes involved. As it stands, the single-application limitation is arguably a feature that simplifies network configuration because there's no need to uniquely map a particular application to a specific mote.

Using an included network monitoring utility, it didn't take me long to discover that the range for the Sentilla motes is quite limited. Taking one mote with me, I didn't get more than a few steps out of the room before contact was lost. I'm guessing the combination of a low-cost PCB trace antenna and packaging that finds the radio semi-shielded by the two AAA batteries isn't helpful. Anyway, at this point, it seems that Sentilla is more into "services" and "software" than being a "module supplier," so the point may be moot.

Actually, the short range made it easier to experiment with the Sentilla network protocol. I confirmed that there are multihop and dynamic discovery aspects by locating one mote within range of the gateway and the other out of range of the gateway, but within the range of the first mote.

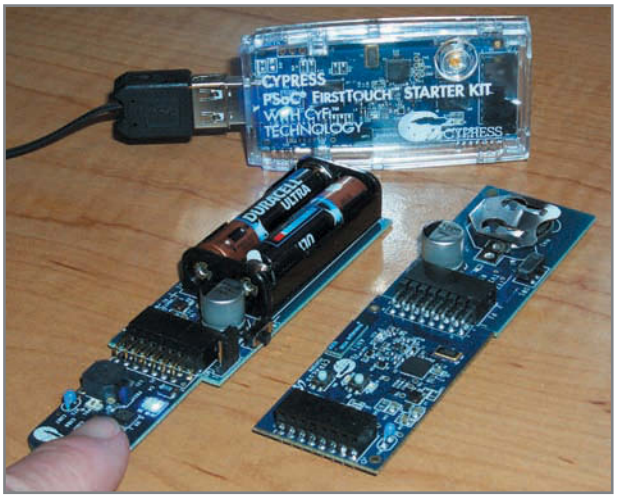


Photo 3—Covering applications from touch, light, and temperature sensing to wireless, the $69.95 Cypress FirstTouch starter kit with CyFi delivers a lot of bang per buck.

After leaving the furthest mote on and turning the closer one off, I tried to install an application. Not surprisingly, nothing happened (i.e., the gateway couldn't reach the furthest mote). Then, I turned on the closer mote, and sure enough, it was able to act as a relay and the application came up on both motes.

Make sure to keep some spare batteries on hand during development. According to the release notes, all of the radio traffic generated by repeated downloads consumes a lot of power, which can drain the batteries quickly. Fair enough, but it does make me wonder about the degree to which the luxury of Java programming comes with a power penalty that carries over to deployment.

"Garbage collection" is when the JVM calls a halt to things in order to clean up the memory map. According to the documentation, garbage collection may take "hundreds of milliseconds," which will definitely limit an application's real-time aspirations.

For PC-side Java applications, debug works as usual with IDE commands to single-step, set a breakpoint or watch variable, and so on. But the usual debug commands don't work for an application running on a mote. Instead, debugging the mote application requires a kind of post-mortem autopsy of "stacktraces" and "thrown exceptions." Not that it's a showstopper, but it is just a bit disappointing that a one-stop programming paradigm doesn't carry all the way through to debugging. Write once, debug everywhere.

Don't get me wrong. I think Sentilla deserves a lot of credit for getting their "Java for all" vision off the launching pad. Implementation issues can always be addressed, but it's only worth doing if the concept is worthy in the first place.

SOFT TOUCH

As the deadline for this column approached, another wadget showed up on my doorstep. This time it was courtesy of Cypress Semiconductor. The good news is that I was able to slipstream what I know about it into this column. The bad news is that I don't know much and don't have a lot of time to get under the hood, so consider this a preview of coming attractions.

The PSoC FirstTouch starter kit I received (see Photo 3) harkens back to the Cypress CapSense technology I wrote about last June ("Touch Me," *Circuit Cellar* 215, 2008). But this time the kit is embellished with the tagline "With CyFi Low-Power RF." Despite the cute name (say "Sci-Fi"), there's nothing too vaporish about what

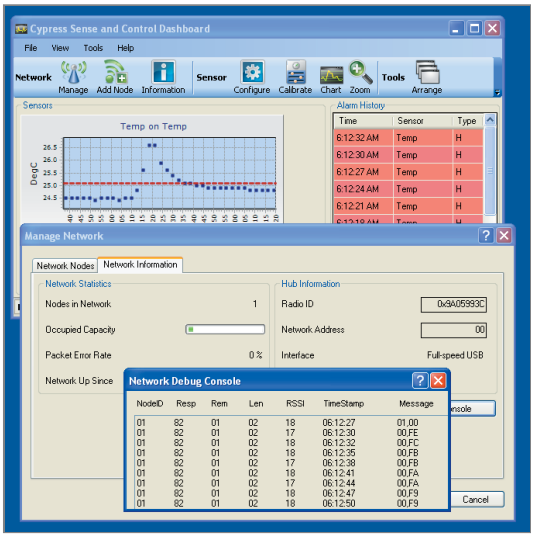


Photo 4—The Cypress kit includes a "sense and control dashboard" utility that handles the basics of CyFi network management.

Cypress is offering because it builds on their earlier RF experience with wireless keyboards, mice, and such.

The CyFi network accommodates up to 250 nodes using a simple star topology with a single hub in the center orchestrating the flow. Compared to fancier mesh or multihop schemes, a star topology has the potential to offer benefits in terms of ease of set up, predictable latency, and throughput. Of course, the only downside is that the hub is a critical link, so if it goes down, so does the network.

CyFi uses a proprietary 2.4-GHz radio, although it shares some characteristics with the IEEE 802.15.4 unit you typically find. For instance, both use Direct Sequence Spread Spectrum (DSSS) technology to reduce interference (both susceptibility and emission) and boost range. CyFi splits the spectrum into 80 1-MHz channels, increasing the odds the radio can find a quiet place to hide amidst all of the 2.4-GHz hubbub.

Along with a PSoC and the CyFi radio transceiver, the final piece of the puzzle is the CyFi network stack software. At just 8 KB, the CyFi footprint is tiny compared to something like ZigBee or Wi-Fi, which means CyFi can work with smaller PSoCs.

Photo 5—Embedded wireless has advanced to the point where it can make sense to use a radio to replace a single wire. The RideMaxx EFI controller uses a Bluetooth radio to connect the handlebar-mounted switch and under-the-seat controller.

Despite the tiny size of the stack, CyFi includes a variety of active link management features that seek to optimize throughput and power in a dynamically changing (interference and range mobility) environment. In addition to the aforementioned channel switching, other features include received signal strength (RSSI), programmable output power and data rate fallback (1 Mbps, 250 kbps, and 125 kbps).

The kit comprises a collection of five boards: two battery adapters (one AAA, one coin cell), a USB plug-in CyFi hub, an RF expansion board (i.e., a CyFi node), and a CapSense demo board. That's quite a bit of hardware value for just the $69.95 price of the kit. You can stack the boards like Legos to try the various demo programs (touch, temperature, light sensing) that are on the CD that comes with the kit.

Pulling it all together is the latest version of the PSoC tool suite, which now integrates the traditional C IDE (PSoC Designer) with Cypress's unique Express visual programming language into a one-tool-does-it-all

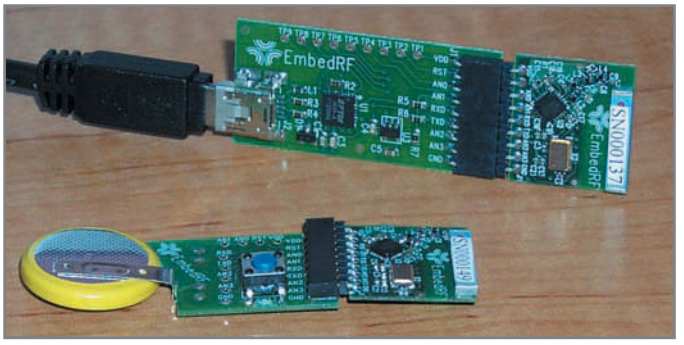


Photo 6—The EmbedRF solution is notable by virtue of its small size and simplicity. The radio modules are the small boards plugged into the USB gateway and a coin-cell battery board, respectively. With analog inputs and digital outputs, the module is capable of stand-alone operation as well as connecting to a host MCU via UART.

package. On the wireless front, the kit also includes a "sense and control dashboard" CyFi network management and monitoring utility that makes it quick and easy to get on the air (see Photo 4).

SIMPLE IS AS SIMPLE DOES

Both the Sentilla Perk and Cypress CyFi solutions have the intelligence to handle a measure of application processing in addition to their wireless role. This is especially useful in timing and safety-critical applications that need immediate local response and can't afford to be held up waiting for a wireless transaction.

On the other hand, there are a lot of simple "sense and send" applications that don't really need to do more than move a little bit of data now and then. For example, thumbing through a biker magazine I came across the RideMaxx from American Micro Fuel Device (www.americanmicrofuel.com). As you can see in Photo 5, it's an add-on that enables handlebar switching of a bike's EFI module between "performance" and "economy" modes.

A challenge for any after-market gear supplier is user installation. Running a wire from the handlebar switch to an under-the-seat EFI module sounds easier than it is. Doing it right means pulling the gas tank, which means draining it, which means at some point it's inevitable that someone will burn their house down. To counter that unfortunate prospect and a myriad of other user-installation support issues, the RideMaxx uses a radio instead of a wire to make the connection.

For basic "wireless wire" applications, a simple solution like the one offered by EmbedRF may be just the ticket (see Photo 6). There's no programming required with EmbedRF. Instead, modules simply deliver fixed-length packets at periodic intervals configurable between 0.25 and 12.75 s in 0.05-s steps. The packets comprise 10 bytes of user data along with a network ID used for addressing, a receive strength (RSSI) indicator, and a packet counter, which can be used by the receiver to detect lost packets.

Table 1 shows the layout for the 10 bytes of user data. The module includes four 10-bit ADC channels, any or all (or none) of which can be assigned to fill user data bytes. Two of the A/D channels (2 and 3) can alternatively be used as digital outputs. That's a key feature that gives the unit a measure, small as it may be, of stand-alone (i.e., no MCU required) closed-loop capability. Note that the digital output option and level is set by command, so it would seem that the user data bytes associated with A/D channels 2 and 3 could still be used. While I'm all for keeping things simple, it would be nice (i.e., use less time, power, and airwaves) if EmbedRF could support shorter packet lengths for applications that can get by with fewer bytes.

Byte number	Description
1	A/D ch. 0 MSB or user data byte 1
2	A/D ch. 0 LSB or user data byte 2
3	A/D ch. 1 MSB or user data byte 3
4	A/D ch. 1 LSB or user data byte 4
5	A/D ch. 2 MSB or user data byte 5
6	A/D ch. 2 LSB or user data byte 6
7	A/D ch. 3 MSB or user data byte 7
8	A/D ch. 3 LSB or user data byte 8
9	User data byte 9
10	User data byte 10

Table 1—The EmbedRF packet format is really simple, giving you exactly 10 bytes to play with. The four 10-bit A/D channels can be individually enabled, each consuming a pair of bytes.

There's a UART at each end for connection to a host MCU and an on-chip packet buffer minimizes timing concerns and interrupt overhead. On the other hand, should the need arise, there's a "zero latency" (i.e., less than 6 ms) option that forces the data to be sent immediately rather than wait for the preprogrammed interval to expire.

Because the EmbedRF nodes are fixed-function, there's no programming or fancy IDE involved. Instead, the development kit ($299) comes with a simple utility program that configures the node (i.e., programs its flash memory configuration settings). It has rudimentary network monitoring and logging capability (see Photo 7).

The EmbedRF radio uses the sub-gigahertz portion of the ISM spectrum (e.g., 915 MHz in the U.S. and 868 MHz in Europe) rather than the more global, and more crowded, 2.4-GHz band. Indeed, according to the datasheet, it can even work with lower frequencies, assuming the duty cycle is limited to meet FCC regulations. The sub-gigahertz frequencies are less crowded and also work better in indoor applications. Although it was by no means a rigorous apples-to-apples comparison, the EmbedRF radio did deliver notably better range than the other 2.4-GHz units during my walk-around range testing.

LESS WIRE, MORE OPTIONS

The message is that when it comes to wireless, there are plenty of options that, pardon the pun, cover the spectrum. Instead of just "smart dust," there's everything from "smart pebbles" to "smart rocks"—and they come in all shapes, sizes, and IQ levels. There are a variety of wireless standards to choose from, such as ZigBee, Bluetooth, Wi-Fi, and the newly emerging 6lowPAN. But as you've learned, there are, and will always be, plenty of proprietary solutions as well. The embedded wireless future covers everything from giant mesh networks with hundreds or thousands of nodes to the simplest two-node "wireless wire" applications.

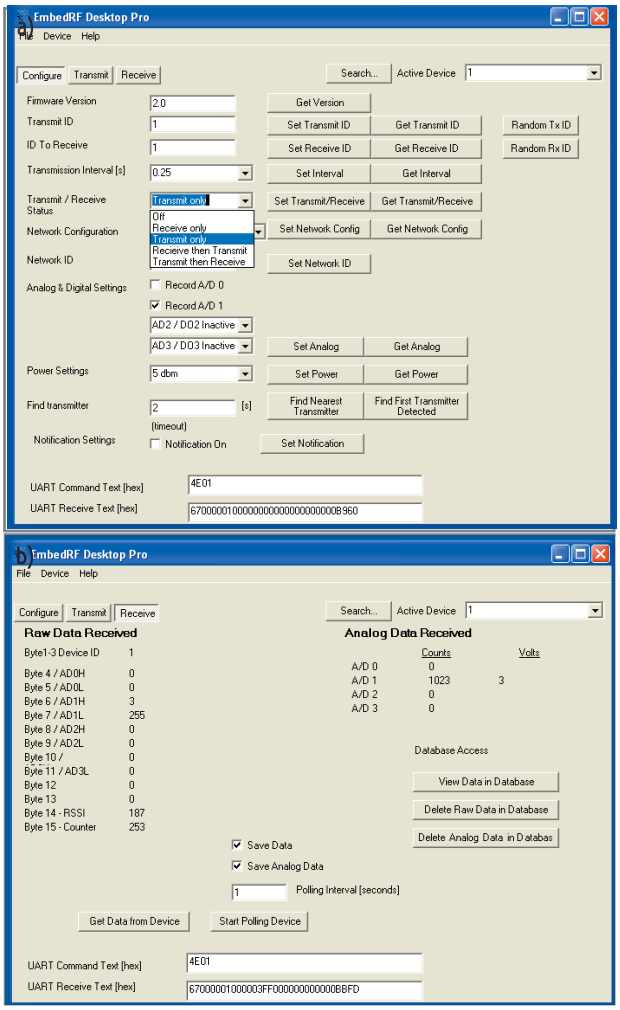


Photo 7—Because there's no programming, the EmbedRF software is simpler than most. It just handles the basics of mote configuration **(a)** and traffic monitoring **(b)**.

Before you launch into a project, make sure you verify the performance of a particular radio against the specifics of your application (e.g., site interference) and implementation (packaging and antennae). The fact that a setup that works great in the lab may not when deployed is a lesson you don't want to learn the hard way. Also keep in mind that while features like spectrum spreading and unique network IDs provide a degree of stealthiness, none of these solutions offers true security, such as encryption. That's an exercise that's left to the reader (i.e., your application software), so make sure you consider it. Finally, never forget that a "wireless" solution is "less" than a "wire" when it comes to high-reliability applications. Your design must be robust enough to deal with the fact there may (make that will) come a time when messages are delayed or even lost.

With the caveats out of the way, I say have at it. There are a zillion applications that can and should take advantage of embedded wireless technology. Ready or not, the wadgets are coming—and "not" isn't an option, so you'd better get ready.

Tom Cantrell has been working on chip, board, and systems design and marketing for several years. You may reach him by e-mail at tom.cantrell@circuitcellar.com.

SOURCES

PSoC FirstTouch Starter kit with CyFi Low-Power RF
Cypress Semiconductor Corp. | www.cypress.com

EmbedRF Development kit
EmbedRF | www.embedrf.com

Perk Pervasive computing kit
Sentilla Corp. | www.sentilla.com

CC2420 802.15.4 Radio and MSP430 microcontroller
Texas Instruments, Inc. | www.ti.com

The Newest Wireless Solutions

ZigBee®
RFID
Bluetooth™
GPS
WLAN
Cellular
ISM

MICROCHIP

MRF24J40MA Radio-Frequency Transceiver Module
www.mouser.com/microchip/a

New Products from:

RABBIT

RCM4510W ZigBee®/802.15.4 RabbitCore® Module
www.mouser.com/rabbitsemi/a

TEXAS INSTRUMENTS
Authorized Distributor

RF Front End Range Extender: CC2590
www.mouser.com/tirfif/a

RFM

XDM2140 Mesh RF Transceiver Modules
www.mouser.com/rfm/a

The ONLY New Catalog Every 90 Days

Experience Mouser's time-to-market advantage with no minimums and same-day shipping of the newest products from more than 366 leading suppliers.

MOUSER® ELECTRONICS
a tti company

The Newest Products For Your Newest Designs

www.mouser.com
Over A Million Products Online

(800) 346-6873

by Jeff Bachiochi

Access SD Memory Cards (Part 2)

Use the FAT File Format to Move Data

Jeff concludes his introduction to SD technology with information about using the FAT file format to move data. He also describes how to create directories and file entries, and delete, save, and append data using ring buffers.

Whether it is audio, video, graphic, database, system, executable, or any number of other file types, they can all be stored, moved, and copied from one storage medium to another, thanks to the operating system's ability to understand how each medium handles the data. While we can't necessarily execute an OS-specific file across multi-OS platforms, many platforms support multiple file systems. So, we can at least exchange data via several supported types of media.

When floppies took over from tapes as the storage medium of choice, DOS introduced us to the FAT file system. At the time, clever designers used shortcuts (space-saving data packing) to cram the most data onto the available media. This inherently put maximum physical limits on what the system could handle. With file sizes measured in kilobytes, this limit was of no concern at the time. Today's needs show how this may have been a bit shortsighted. Improvements to the original FAT file system have met those needs for now. Can we expect this to remain adequate? It services its intended need well; and while we should never say never, I believe it will continue to be supported for many

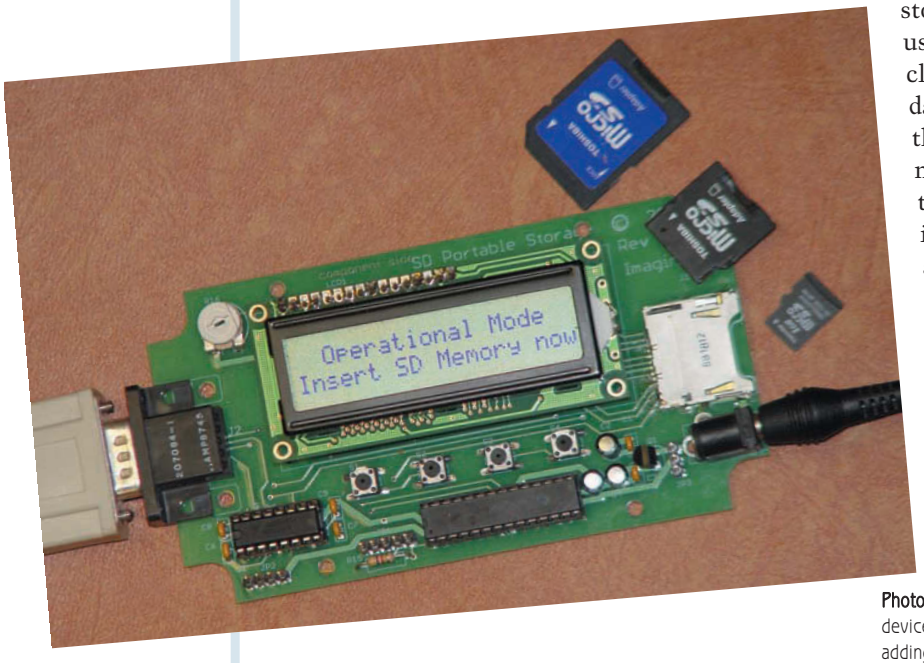


Photo 1—Like everything else, even removable solid-state memory devices have undergone a shrinking process. SD cards are great for adding file storage to your project.

Address	HEX								ASCII	
0910	1309	00FF	FFFF	FF00	0000	0000	210B	FE53	.....	....!..S
0920	4420	4D49	4352	4F20	2020	0800	0000	0000	D MICRO	.....
0930	0000	0000	00BC	5D0B	3900	0000	0000	0046	.....].	9.....F
0940	494C	4531	2020	2054	5854	2000	0000	0000	ILE1 T	XT .....
0950	001C	3900	0040	5819	3902	0004	0000	0046	..9..@X.	9.....F
0960	494C	4532	2020	2054	5854	2000	0000	0000	ILE2 T	XT .....
0970	0000	0000	0000	0000	0000	0000	0000	004F	.....	.....O
0980	4E45	2020	2020	2020	2020	1000	0000	0000	NE	.....
0990	0000	0000	0000	0000	0004	0000	0000	004A	.....	.....J
09A0	4546	4620	2020	204A	5047	2018	3D50	5819	EFF J	PG .=PX.
09B0	3919	3900	0025	4F59	3803	00EB	2700	0042	9.9..%OY	8...'..B
09C0	5300	7500	6E00	7300	6500	0F00	0274	002E	S.u.n.s.	e....t..
09D0	006A	0070	0067	0000	0000	00FF	FFFF	FF01	.j.p g..	.....
09E0	4800	6100	6C00	6600	2000	0F00	0244	006F	H.a.l.f.	....D.o
09F0	006D	0065	0020	0061	0000	0074	0020	0048	.m.e. .a	...t. .H
0A00	414C	4644	4F7E	314A	5047	2000	9371	5819	ALFDO~1J	PG ..qX.
0A10	3919	3900	00D9	5946	360C	0015	5928	00E5	9.9...YF	6...Y(..
0A20	3938	3136	3838	5F20	2020	1000	AF1B	181A	981688_	.....
0A30	391A	3900	001C	181A	39AE	0000	4000	0000	9.9.....	9.. @...
0A40	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A50	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A60	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A70	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A80	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A90	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AA0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AB0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AC0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AD0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AE0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AF0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0B00	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0B10	0000	0000	0000	0000	0000	0000	0000	008D	.....	.....

Figure 1—This 512-byte sector dump of the first sector of the root directory shows a few 32-byte directory entries. The first entry (beginning at address 0x091F) is reserved for the optional volume name, in this case "SD MICRO." The remaining entries consist of two text files, a subdirectory, and two digital image files. Note that these data dumps are depictions of the memory, as displayed by the ICD2 debugger, address data is in.

years because it is so ingrained into today's media.

Last month, I started discussing how you can use SD solid-state storage media in your embedded designs. Although the intended nibble interface is open to those who join the SD Association, a simplified version of the physical layer specification is available at www.sdcard.com. This interface is a standard SPI. While the throughput might be slower (1 bit versus 4 bits), it works well with a microcontroller's standard SPI hardware. Although I consider the USB thumb drive to be the most popular file transfer media, most cameras, phones, PDAs, MP3 players, and other portable electronics use a smaller form of solid-state storage. The SD card is widely used and available. This makes it a perfect match for embedded products (see Photo 1).

If you haven't read the first article in this series, please take a few minutes to review the points I covered last month ("Access SD Memory Cards (Part 1): Solid-State Storage Media in Embedded Apps," *Circuit Cellar* 222, 2009). The FAT file system is divided into four sections: the reserved region, the FAT region, the root directory region, and the file and directory data region. The reserved section contains the boot sector or BIOS parameter block with definitions of various media parameters including where to find the other regions. The FAT region is a list of tags indicating the status of each cluster or group of sectors. The root directory region contains a list of files or directory names and corresponding information. The file and (sub) directory region hold the actual file data or additional subdirectories.

With the interfacing and protocols introduced last month, I had room to describe only the first of four sections of the FAT file system. This month, I will begin by digging into the root directory region using the FAT16 format. From the reserved region, I previously determined a number of important parameters. For your reference, there are 512 entries in a directory. There are 237 sectors in each of two FATs. The first FAT begins in sector 0x86. There are 512 bytes per sector. There are 32 sectors per cluster, and there are 32 bytes per directory entry. Because I know that the FAT begins at sector 0x86 and each of the two FATs are 237 (0xED) sectors each, the root directory must be at 0x86 + 0xED + 0xED or sector 0x260. I'll start by looking at this sector or the logical block address (LBA).

ROOT DIRECTORY, FAT REGIONS

A dump of LBA 260 (the first root directory sector in Figure 1) shows nine entries: a volume ID, four files, two directories (one deleted), and two entries associated with a long file name. All of the directory entries, whether they are file or directory entries, take on the same format. When they are initially formatted, all directory entries contain 0x00s. A 0x00 as the first byte of directory entries indicates that the entry is empty and there are no more used entries in the rest of the directory. (This fact saves search time.) A file name (or directory name) must consist (for the most part) of one or more alphanumeric characters. Although long file names can exceed eight characters, I will not discuss them here. (You can read about how they are handled in most over views of the FAT file system.) The directory structure begins with an eight-character name and a three-character extension. A file name has an implied "." between its name and extension. This is determined by the twelfth byte in the directory entry. The attribute byte contains flag bits indicating, among other things, if the entry is a file or a directory. Other bytes contain various time and date information. Besides the name and attribute parameters, the last two are significant. The cluster value tells you where to look for the first sector of the file or subdirectory and the FAT location associated with the entry. The file length indicates the file size in bytes. (Directories always have a zero file length as can empty files.)

The first sector (of either copy) of a FAT region holds 256

16-bit pointers for the cluster numbers 0x00 to 0xFF (with the following sectors holding additional pointers for the remaining clusters). The 16-bit value stored at each cluster position begins life as zero. (Except for cluster 0 and cluster 1, these are reserved and cannot be used.) A value of zero means the cluster is not in use. When a directory entry is created, its cluster low word value (at offset 0x1a in the directory entry) points to the first cluster used by the entry. In Figure 1, cluster 2 was assigned to directory entry FILE.TXT, a text file of 4 bytes. This refers to the cluster where the data of the file begins and the FAT location where more information is held. When the entry is a directory or a file that is less than 16,384 bytes, it does not require more than a single cluster (32 sectors) (see Table 1). Therefore, the 16-bit FAT entry for cluster 0x0002 contains a value that indicates that this is the last cluster in the file with a value of (0xFFF8-0xFFFF). Otherwise, the 16-bit FAT entry for cluster 0x0002 would contain the value of the next cluster used by the file. Unless a cluster is the last, each FAT entry would then point to an additional cluster, and so on.

In the dump of the first sector of the FAT, you can see how this works (see Figure 2). The sector data begins after the block byte at address 0x90E. The first few words are 0xFFF8s and 0xFFFFs. The third word is for cluster 0x0002. The 0xFFFF indicates that this is the last cluster. I added a large file (HALFDO~1.JPG) so you can see how the chaining works. In the root directory, this file was assigned cluster 0x000C (12th). The twelfth word in the FAT entry is not 0xFFFF, but 0x000D. This file's data does not end in cluster 0x000C but continues on into cluster 0x000D. Now look at the thirteenth FAT word and you will see it doesn't end there but each one points to another cluster up to cluster 0x00AD. This FAT entry is 0xFFFF, indicating that this is the last cluster of the file. Back in the last 4 bytes of this file's entry in the root directory sector, you can see that the file's size for this file is 0x00285915 (2,644,245 bytes) and requires many clusters.

Additional clusters were allocated sequentially for this file, but they don't have to be. The chaining process can jump to any unused cluster. This is a good time to mention what happens when something is deleted. If a single byte—the first character of a directory entry—is changed to 0xE5, the file (or subdirectory) is considered unavailable. Note that all of the information including the actual data has not been altered in any way. This makes every deleted file potentially recoverable, unless it is damaged by data that has been subsequently written to the disk. Thus, remember that deleting a file does not delete the information. Apply a full format to totally wipe the media, not just a quick erase.

Address	HEX								ASCII	
0910	1309	00FF	FFFF	FF00	0000	0000	210B	FEF8	.....	....!...
0920	FFFF	FFFF	FFFF	FFF8	FFF8	FFF8	FFF8	FFF8	.....	.....
0930	FFF8	FFF8	FFF8	FF0D	000E	000F	0010	0011	.....	.....
0940	0012	0013	0014	0015	0016	0017	0018	0019	.....	.....
0950	001A	001B	001C	001D	001E	001F	0020	0021	.....	.....
0960	0022	0023	0024	0025	0026	0027	0028	0029	." #.$.%	.&.'.(.)
0970	002A	002B	002C	002D	002E	002F	0030	0031	.*.+.,.-	.../.0.1
0980	0032	0033	0034	0035	0036	0037	0038	0039	.2.3.4.5	.6.7.8.9
0990	003A	003B	003C	003D	003E	003F	0040	0041	.:.;.<.=	.>.? @.A
09A0	0042	0043	0044	0045	0046	0047	0048	0049	.B.C.D.E	.F.G.H.I
09B0	004A	004B	004C	004D	004E	004F	0050	0051	.J.K.L.M	.N.O.P.Q
09C0	0052	0053	0054	0055	0056	0057	0058	0059	.R.S.T.U	.V.W.X.Y
09D0	005A	005B	005C	005D	005E	005F	0060	0061	.Z.[.\.]	.^._.`.a
09E0	0062	0063	0064	0065	0066	0067	0068	0069	.b.c.d.e	.f.g h.i
09F0	006A	006B	006C	006D	006E	006F	0070	0071	.j.k.l.m	.n.o p.q
0A00	0072	0073	0074	0075	0076	0077	0078	0079	.r.s.t.u	.v.w.x.y
0A10	007A	007B	007C	007D	007E	007F	0080	0081	.z.{.|.}	.~.....
0A20	0082	0083	0084	0085	0086	0087	0088	0089	.....	.....
0A30	008A	008B	008C	008D	008E	008F	0090	0091	.....	.....
0A40	0092	0093	0094	0095	0096	0097	0098	0099	.....	.....
0A50	009A	009B	009C	009D	009E	009F	00A0	00A1	.....	.....
0A60	00A2	00A3	00A4	00A5	00A6	00A7	00A8	00A9	.....	.....
0A70	00AA	00AB	00AC	00AD	00FF	FF00	0000	0000	.....	.....
0A80	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A90	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AA0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AB0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AC0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AD0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AE0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AF0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0B00	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0B10	0000	0000	0000	0000	0000	0000	0000	000E	.....	.....

Figure 2—This is the first sector of the first FAT. This sector holds all of the 16-bit (for FAT16) FAT entries for cluster 0x00–0xFF (successive sectors hold successive cluster entries). Note that the cluster entry in a directory entry (for a subdirectory or file) points to the cluster where the entries actual data resides and also to an entry in the FAT. Any FAT entry with a value greater than 0xFFEF signifies this is the last cluster used by the subdirectory or file. Other values greater than 1 (0 and 1 are reserved) point to the next cluster used by the subdirectory or file.

PROJECT OBJECTIVES

I could have wimped out and just added the necessary functions to do logging to and dumping from a file already on the SD card. But, I wanted to make this project as helpful as possible. I added functions that help show how things are done but certainly aren't necessary for this project. The 2 × 20 LCD and three-button interface really made this a challenge. Assuming your formatted SD card has no files on it, you would need to be able to create an entry in the root directory region. This project gives you three choices: create a file, create a (sub) directory, or exit without doing anything. I use the top line of the LCD to give you a choice and the second line to indicate the function of the buttons. Usually, this is button 1, display the next item, button 2, display the previous item, and button 3, choose the item. Let's create

Directory entry offset	Parameter size	Value found	Description	Meaning
0x00	11 Bytes	"FILE1 TXT"	Name	"FILE1 .TXT"
0x0B	Byte	0x20	Attributes Bit 0 = Read only Bit1 = Hidden Bit2 = System Bit3 = Volume ID Bit4 = Directory Bit5 = Archive Bits0:3 = Long name	Archived file
0x0C	Byte	0x00	Reserved	
0x0D	Byte	0x00	Crea ion time (tenths)	
0x0E	Word	0x0000	Crea ion time Bits0:4 = Seconds/2 Bits5:10 = Minutes Bits11:15 = Hours	
0x10	Word	0x0000	Crea ion date Bits0:4 = Day Bits5:8 = Month Bits9:15 = Year + 1980	
0x12	Word	0x391C	Date of last access	8/28/2008
0x14	Word	0x0000	Reserved	
0x16	Word	0x5840	Time of last write	11 02:00 AM
0x18	Word	0x3919	Date of last write	8/25/2008
0x1A	Word	0x0002	Cluster low word	Cluster = 2
0x1C	Double word	0x00000004	File size (bytes)	File size = 4

Table 1—Each directory entry can hold useful information about a file or directory.

a (sub) directory in the root directory region (see Figure 3).

Entering a directory name is the first challenge. I can take advantage of a previously required routine to move 20 characters from the RAM into row 1 of the LCD. The blinking cursor never shows up on the LCD because the LCD print statement leaves it beyond the last visible character position. By moving the cursor to character position 1–20, it can be used as a prompt to you. Buttons 1 and 2 cycle forward and backward through legal characters (including a blank space and backspace arrow for correcting mistakes). Button 3 advances the cursor to the next position (or backspaces). The entry routine exits after eight characters are entered.

All of this entry is worthless if we don't have any open clusters, so we'd better check the FAT region. I use three basic routines with the FAT: locate a FAT entry (cluster), read the cluster entry, and write the cluster entry. I want to find an unused FAT entry (read 0x0000) and claim it for the new directory (write 0xFFFF). The offset into the FAT where the unused entry is found becomes the cluster number for the new directory. Note that directories require a single cluster and therefore will not require chaining in the FAT. A second copy of the FAT is kept for safety measures. While this isn't normally used, it's a good idea to keep it updated for compatibility.

Now the new cluster becomes the home for the new second-level directory. You must add two directory entries to this clean slate, the "dot" and "dot dot" director y entries. The "." directory uses the new cluster number as its cluster pointer. The ".." directory uses the parent cluster as its cluster pointer.

One last operation is necessary before this new directory can be used. You must go back to the parent director y and add the directory name (and pertinent data) to an empty directory entry. This completes the chain that can point us to the new director y's cluster.

NEW COMMANDS

Adding a file to a director y is actually easier than adding a directory. But there are a few differences. The file name includes a three-character extension. This is indicated on the LCD by a "." separating the eight-character name from the extension. The FAT is handled the same way. (At this point, the file size is zero.) Because the file size is zero, nothing needs to be placed anywhere in the new cluster. An empty directory entry needs to be filled in with the file name and pertinent data.

Assuming a new directory and file have been created (exist) in the root directory cluster, a number of new commands become available. The appropriate commands are offered when a directory or a file name is chosen. When a directory is selected, you can change to the new directory, delete the directory, add a new directory, add a new file, or return to the present directory. When a file is selected, you can delete the file, add a file, dump a file, log to a file, add a new directory, or return to the present directory.

Execution spends most of its time in the operational mode. This is where all the file and director y names in the present directory are displayed (one at a time). When an SD card is inser ted, this will always be the root directory (in this case, that's LBA 0x0260). If a directory name is selected (e.g., ONE) and you choose to switch directories, the entry's first cluster pointer (0x0004) is used as the cluster to be used as the present directory. (Refer to Figure 1 and you will see that directory ONE uses cluster 0x0004.) Figure 4 shows a dump of this

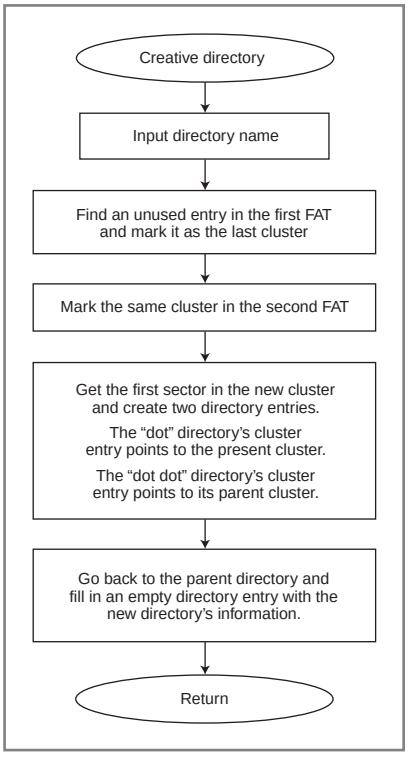


Figure 3—This flowchart greatly simplifies the process for creating a directory. When searching the FAT, for instance, you may need to search every word in every sector of every cluster (up to the maximum 237 sectors per FAT) for an entry just to find out that there is no room left. Obviously, the search loops are exercised more as the media fills up.

directory's first sector, LBA = 0x0003 (Cluster-1) × 0x20 (sectors per cluster) + 0x0260 (root directory) or 0x02C0. You can see the "dot," "dot dot," and (sub) directory TWO entries in the ONE (sub) directory. Note the first cluster word locations (at offset 0x1A of each directory entry pointing to the respective clusters of those entities). By changing the first character 0x54 of directory TWO to 0x0E5, this entry would be eliminated. When using this application to delete a directory, no check is made as to which files or subdirectories might be deleted as a result of this action. Many operating systems won't allow a directory to be deleted unless it is completely empty!

On the file side, choosing FILE1.TXT in the root directory gives us the opportunity to dump this file (see Figure 1). The first cluster position (offset 0x1A) in the FILE1.TXT directory entry (0x0002) points to where this file's data is stored. From the File_Size position (0x1C) in the FILE1.TXT directory entry, this file's length is 0x00000004. The first sector dump of cluster 0x0002 LBA 0x0280 shows the first four characters of this file to be "Test" (see Figure 5). This circuit's serial port is used as the output device for the dump operation. The file length determines how many characters will be sent. The characters are collected from every LBA (every sector of every cluster in the file's chain) until the appropriate number of characters have been transmitted. This is easy for FILE1.TXT because it has only four characters. However, the HALFDO~1.JPG file has multiple clusters. To dump this file, we begin with the first sector of cluster 0x000C, LBA 0x03C0 (0x000B × 0x20 + 0x0260), and send all 512 bytes. The LBA is incremented and all bytes of each sector are sent until the whole cluster (0x20 sectors) has been sent (0x100 bytes × 0x20 sectors = 0x2000 or 8,192 bytes). The FAT entry at word offset 0x000C is then interrogated to find out which cluster holds the next portion of data. An entire cluster's worth of data is sent (another 8,192 bytes.) The whole FAT thing is repeated until 2,644,245 bytes have been transmitted.

Address	HEX								ASCII	
0910	1309	00FF	FFFF	FF00	0000	0000	210B	FE2E	.....	....!...
0920	2020	2020	2020	2020	2020	1000	0000	0000		.....
0930	0000	0000	0000	0000	0004	0000	0000	002E	.....	.....
0940	2E20	2020	2020	2020	2020	1000	0000	0000	.	
0950	0000	0000	0000	0000	0000	0000	0000	0054	.....	.....T
0960	574F	2020	2020	2020	2020	1000	0000	0000	WO	.....
0970	0000	0000	0000	0000	0005	0000	0000	0000	.....	.....
0980	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0990	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09A0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09B0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09C0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09D0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09E0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09F0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A00	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A10	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A20	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A30	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A40	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A50	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A60	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A70	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A80	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A90	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AA0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AB0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AC0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AD0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AE0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AF0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0B00	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0B10	0000	0000	0000	0000	0000	0000	0000	00EE	.....	.....

Figure 4— The root directory's subdirectory entry (ONE) points to cluster 0x00004, LBA 0x2C0. This dump shows how it is similar to the root directory but has two required entries, the "dot" entry pointing to its own cluster, and the "dot dot" entry pointing to a directory (one level up), in this case the root directory. This subdirectory also holds a new subdirectory (TWO).

RING AROUND THE ROSIE

The last function, and the main reason for this project, is the logging of data from the serial port. If you have been following the processes up to this point, you should have a good understanding of how this is accomplished. The serial port has been implemented with output and input ring buffers. Each ring buffer has a head and a tail pointer. When the buffers are empty, the head pointer equals the tail pointer. Each pointer can point to any address of the buffer from its lowest address to its highest address (the buffer's length). The pointers are usually incremented and must be repositioned to the beginning of the buffer if they exceed the buffer's length—thus the term ring, or circular buffer. Once the ring buffers are implemented, you no longer have to deal with the serial port hardware directly, just the loading of the output buffer and unloading of the input buffer.

From the serial port side, any characters received cause an RX interrupt. The interrupt routine handles taking the received character and putting it into the input ring buffer. The routine may add only characters to the ring buffer via the buffer's head pointer. Received characters are placed at the head pointer (and the head pointer is incremented) only if there is room in the buffer. There is room until the head pointer + 1 equals the tail pointer. At this point, adding a character (and incrementing the head pointer) makes the head and tail pointers equal. This was previously defined as an empty buffer, so this would produce a buffer overrun condition and the buffer's data would be lost. One of two things must happen at this point, either the serial port must use flow control to stop the data from coming in or the data must be tossed out. This should not occur if the application can remove the data from the ring buffer and store it in the SD media faster than the data can be received.

On the serial output side, the output ring buffer will be

emptied via the output ring buffer's tail pointer. Unless the output ring buffer's tail point equals the head pointer, there is a character available for transmission. The TX interrupt routine is responsible for keeping the output ring buffer empty.

When the four characters of FILE1.TXT were dumped, the characters were placed into the output ring buffer using the buffer's head pointer. Because moving characters from the sector buffer to the output ring buffer will be fast, the application may stall waiting for the TX interrupt routine to empty the output ring buffer. So the dump time will be directly related to the data rate.

When logging data, assuming the data rate is sufficiently high compared to the data input rate, any bottleneck will come from the SD cards inability to write a block of data fast enough and get back for more without allowing the ring buffer to wrap. I thought I'd try logging a 0.5-MB file at a data rate of 19,200 bps for a test. I expected approximately 2,000 characters per second. I saw a sector write (lasting 35 ms) every 250 ms. That's four sectors, or 2,048 (i.e., 512 × 4) bytes per second. I pulled the SD card out and put it in my PC to check the file. It was the proper length at 567,408 bytes and viewed correctly. So, while I was in Windows Explorer, I used it to create an empty text file to try another test at a higher data rate.

I put the SD card back into my project board and repeated the test. I saw a sector write (lasting 35 ms) every 125 ms. Looking good! However, when I ended this logging session, the directory was trashed. (It viewed as if it had lots of garbage entries.) Hmm. It must have run into timing issues. But wait, that would have caused a loss of data and not affected the directory. Hmm. The short story is the directory entry created using Windows Explorer didn't assign a FAT entry (so the FAT entry was zero) because an empty file has 0 bytes. When I began logging, I looked at the directory entry's FAT and assumed it had been assigned. (After all, that's what I do in this application.) Because a FAT entry of zero is used by the root directory, logging to it causes the root directory's sector to be overwritten, causing catastrophic results. With this incorrect assumption corrected, logging at 38,400 bps worked as expected.

Address	HEX								ASCII	
0910	1309	00FF	FFFF	FF00	0000	0000	210B	FE54	.....	....!..T
0920	6573	7400	0000	0000	0000	0000	0000	0000	est.....	.....
0930	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0940	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0950	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0960	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0970	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0980	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0990	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09A0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09B0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09C0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09D0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09E0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
09F0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A00	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A10	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A20	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A30	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A40	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A50	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A60	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A70	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A80	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0A90	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AA0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AB0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AC0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AD0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AE0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0AF0	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0B00	0000	0000	0000	0000	0000	0000	0000	0000	.....	.....
0B10	0000	0000	0000	0000	0000	0000	0000	000E	.....	.....

Figure 5—The minimum chunk that can be allocated to a file (or subdirectory) is one cluster. A cluster has 32 sectors associated with it. This file (FILE1.TXT) of only 4 bytes and any file, will have at least 32 sectors reserved for it.

SO MUCH MORE

While this project succeeds in performing the tasks required to explain how to use SD memory in a project, there is plenty more that can be discussed. The Microchip Technology PIC24FJ64GA002 has other useful hardware that you can explore. I purposely left time and date stamping out of my directory entry routines (other than making sure the entries were legal). This microcontroller has a hardware real-time clock that you can use to implement accurate time and date entries. You will also find a programmable-length CRC generator, which would make using CRCs a lot easier. I'll leave these and other enhancements for to you to experiment with. If you would like me to devote additional space to any of this, drop me an e-mail.

You'll find C routines implementing the FAT file system offered by many manufacturers. However, you won't learn much about it by just calling someone else's code. I like to seize every opportunity to expand my knowledge base. Time constraints don't always allow this process, but I hope I've sparked your curiosity. Every so often, you should try to take this less traveled path.

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.*

SOURCE

PIC24FJ64GA002 Microcontroller
Microchip Technology, Inc.
www.microchip.com

ICbank
Gateway to Electronics

MP3P DIY KIT, Do it yourself

(Include Firmware Full source Code, Schematic)

• myPIC

Only
$160 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 dsPIC33FJ256GP710 / 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)

Only
$150 qty 100
$200 qty 1

Powerful feature

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

Item	Specification
MCU	Atmel ATmega128L
MP3 Decoder	VS1002 / VS1003(WMA)
IDE Interface	Standard IDE type HDD(2.5", 3.5")
Power	12V, 1.5A
LCD	128 x 64 Graphic LCD
Etc	Firmware download/update with AVR ISP connector

• myAudio (MP3 DIY KIT IDE)

Only
$180 qty 100
$220 qty 1

Powerful feature

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

ICbank
Gateway to Electronics

www.icbank.com

by Robert Lacoste

Microstrip Techniques

What are microstrips and how are they used in PCB designs? Robert sheds some light on the topic of microstrips and describes microstripping techniques that you can use at your workbench.

Welcome back to The Darker Side. This month, I am really a happy guy because I have achieved two of my goals. First, it should please my editors because I will present a subject in line with *Circuit Cellar*'s monthly theme, which is, if you haven't read the cover, wireless communications. Second, the title of this column has never been more appropriate, because I will talk about something that may really seem magical to the novice. Follow along as I introduce microstrip techniques and explain how to implement zero-cost components by simply drawing PCB tracks.

BACK TO IMPEDANCE MATCHING

In "Antenna Basics," I explained why impedance matching can make the difference between a working and a nonworking project, especially in RF designs where signal power is an expensive resource (*Circuit Cellar* 211, 2008). In a nutshell, the power transfer between a source and a receiver is maximized when impedances are matched. This means that the source impedance is the complex conjugate of the load impedance. Both resistances are equal and both reactances are equal in value, but with opposite signs. Roughly speaking, a capacitive load must be matched with an inductive source and vice versa. If the source impedance is a 50-Ω pure resistance, matching will be achieved when the load is also a pure 50-Ω resistance. And if a cable or wire is used between them, it must also have a 50-Ω characteristic impedance. If the matching is not perfect, you won't get as much power as you could: some power will be dissipated somewhere else than in your load. It is usually reflected back to the source and dissipated,

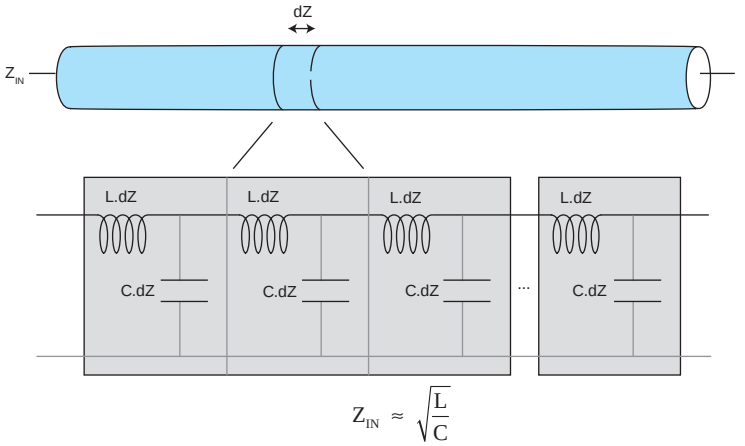
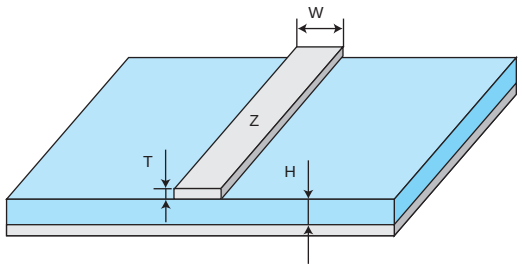


Figure 1—A transmission line can be modeled as a succession of small L/C networks. The mathematical relationship between the L and C parameters and the characteristic impedance of the line is simple to apply and slightly more complex to demonstrate.



$$Z \approx \frac{87}{\sqrt{\varepsilon_r + 1.41}} \bullet \ln\left[\frac{5.98\,H}{0.8\,W + T}\right]$$

$$\text{if } W > T: \varepsilon_{eff} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}\left(1 + 12\left(\frac{H}{W}\right)\right)^{-\frac{1}{2}}$$

Substrate	ε_r	Loss/inch per gigahertz	Z = 50 Ω, T = 35 µm			
			H = 1.6 mm		H = 0.8 mm	
			W	ε_{eff}	W	ε_{eff}
FR4	4.5 ± 10%	0.08 dB	2.95 mm	3.38	1.45 mm	3.38
RO4003	3.38 ± 1.5%	0.02 dB	3.30 mm	2.65	1.65 mm	2.65

Figure 2—A microstrip is simply a copper track running on a side of the PCB while the other side is a plain ground plane. The formula will give you the characteristic impedance of the track, as well as the effective dielectric constant based on the geometric parameters. The table provides usual values for 1.6- and 0.8-mm-thick PCBs, as well as for the standard FR4 substrate or the most advanced Rogers RO4003.

but it could also generate nasty effects like distortion or spurious signals.

You must be concerned with impedance matching, especially in any high-frequency project, and 90% of the time you should try to keep a 50-Ω characteristic impedance through your circuit, or may be 75 Ω if you are working on video. Do you know where these usual 50- and 75-Ω values come from? I must admit that I discovered the supposed explanation only recently in Thomas H. Lee's book, *Planar Microware Engineering: A Practical Guide to Theory, Measurement, and Circuits*. According to Lee, for a given coaxial cable diameter, there is a precise ratio of inner to outer conductor size, which gives the minimum intrinsic resistive loss. This ratio corresponds to a characteristic impedance of 77 Ω. That's why a close value, 75 Ω, is used for video where signals are small and where attenuation should be kept as small as possible, even on long cables. Okay, but why 50 Ω elsewhere? Because there is another optimization to deal with. A coaxial cable can transmit a given maximum peak power, corresponding to the dielectric breakdown of the cable due to high RF voltages. It happens that, for a fixed external size, this power-handling capacity is maximized with another ratio of inner to outer conductor size, corresponding to a characteristic impedance of 30 Ω, at least when air is used as a dielectric. Back in the 1930s, engineers hesitated between 77 and 30 Ω and chose an intermediate value for common coaxial cables, 50 Ω. To be honest, there are other explanations on the 'Net, but this one seems plausible.

TRANSMISSION TO MICROSTRIP

Before digging into the microstrip topic, let's spend a minute on transmission lines. Imagine you have a piece of coaxial cable divided into a large number of small sections, each of length dZ

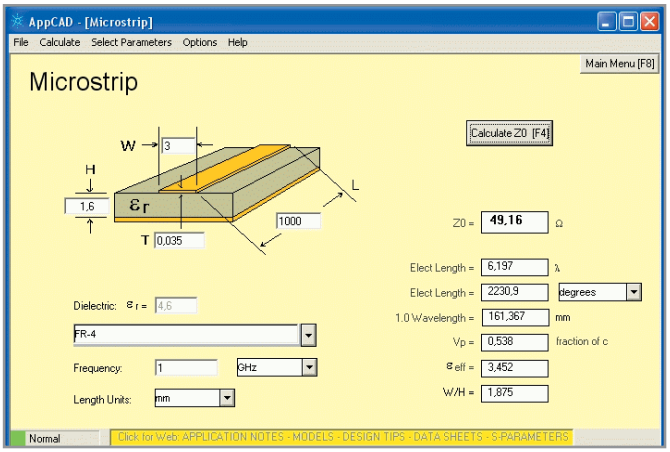


Photo 1—AppCad is a free PC utility, which will simplify your life. It includes a powerful Microstrip calculator.

(see Figure 1). If you ignore parasitic resistances, each section could be modeled as a small serial inductance, because any wire has a nonnull inductance, and as a small capacitor between the wire and the ground, because the central wire is not far from the grounded shield. These parasitic inductances and capacitances are roughly proportional to the length of the small section, so they can be noted L.dZ and C.dZ, with L and C in henries per meter and farads per meter. Therefore, the wire can be approximated as a serially connected set of identical L.dZ/C.dZ networks. If you apply a voltage on one end of the cable, some current will flow until all capacitors are charged. If you are, well, rich and have an infinite length of cable, this current will flow forever. And if you apply an AC input, you will get a given impedance. This impedance is what is called the cable's characteristic impedance. In fact, it can be easily demonstrated that this characteristic impedance is simply the square root of L/C. Just remember that the characteristic impedance of the cable increases with L and decreases with C.

What happens if you do not have a coaxial cable but simply a copper track on a side of a PCB with a complete ground plane on the other side? The situation is the same as it is with the coaxial example. The track can be split into small sections, and each section can be modeled the same way. So, this track will have a characteristic impedance. Such a PCB track on a full ground plane is called a microstrip, which is the most common way to connect RF components on a PCB (see Figure 2). You can use other settings like stripline—which is a track sandwiched between two ground planes on a multilayer PCB—but microstrip is the most frequently used option because it is affordable and well-suited to SMT components.

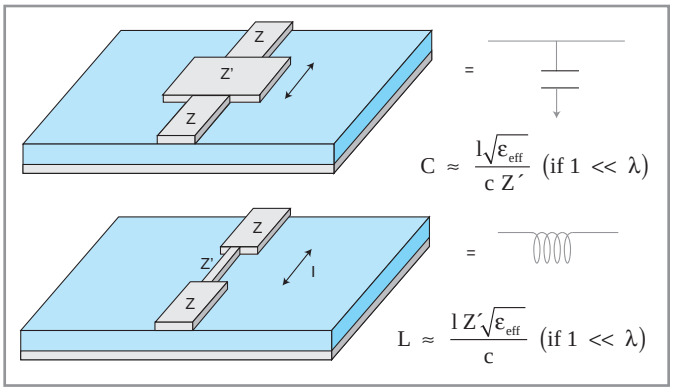


Figure 3—Parallel capacitors or serial inductors can be easily implemented on a microstrip line just by changing the track width, meaning the track impedance, on a precise length. Any reasonable impedance Z can be used.

How do you control the characteristic impedance of a microstrip track? Usually with the only parameter that you can easily manage on your PCB CAD tool: track width. Intuitively, if the track is wider, the capacitance between the track and the ground plane will increase and the characteristic impedance will decrease. If the track is thinner, its inductance and its characteristic impedance will increase too. So, there should be a given track width that corresponds exactly to 50 Ω, at least for a given PCB technology. This width is dependent on the PCB substrate (FR4 is the most common) through its dielectric constant and the PCB thickness (1.6 mm or 0.8 mm for double-sided designs) and slightly on the copper thickness, which is 35 µm most of the time. The formula and the usual values are given in Figure 2, even if there are good free calculation tools on the 'Net, such as Agilent Technologies AppCad (see Photo 1). Roughly a 50-Ω track corresponds to a 3-mm wide track on a standard 1.6-mm thick PCB and to 1.5 mm on a 0.8-mm PCB. That's why it is often more appropriate to use a 0.8-mm thick PCBs for RF projects, simply because the tracks have a more manageable width. In summary, whenever you design a high-frequency project, you must always use tracks with the width for proper impedance matching with a full ground plane

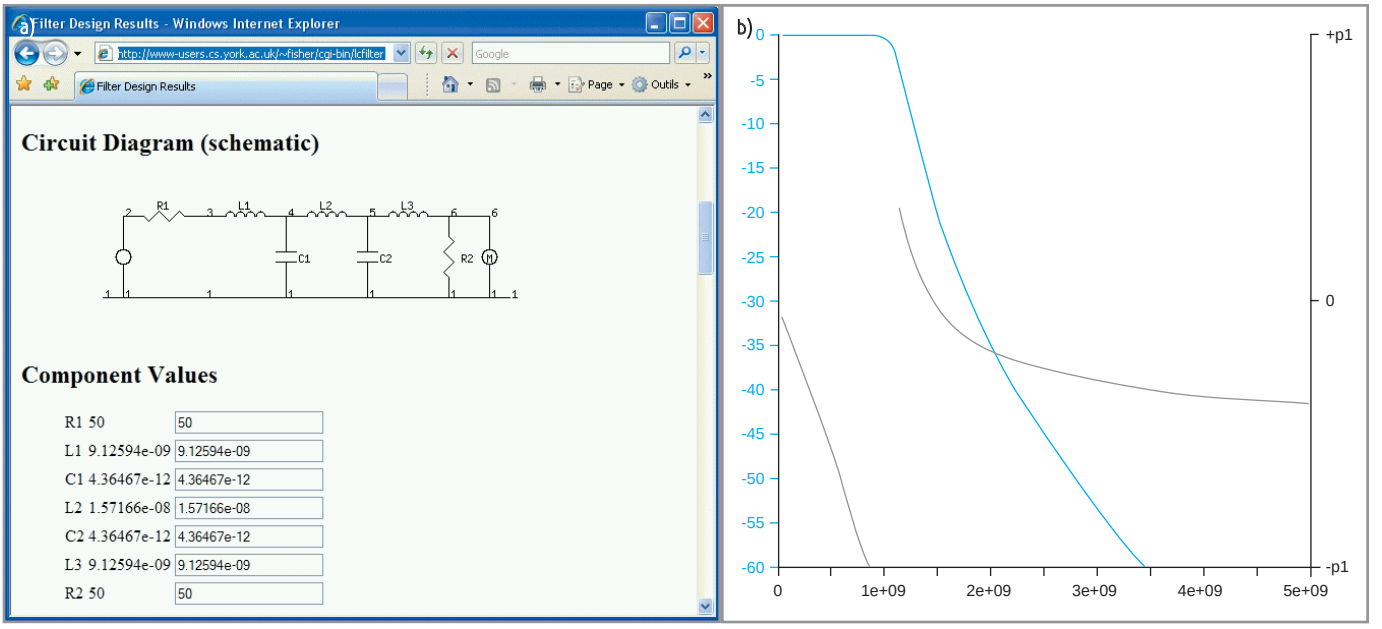


Figure 4—I designed the classic lumped component version of the 1-GHz low-pass filter with a 'Net-based calculator. It used three inductors and two capacitors.

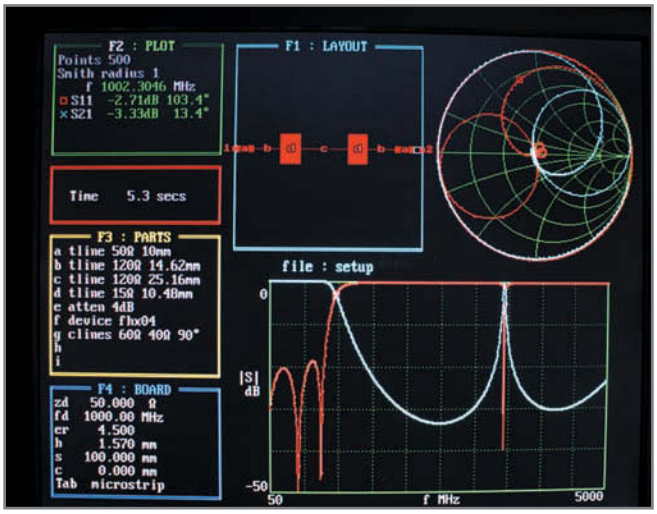


Photo 2—The PUFF application is a DOS-based microstrip drawing and circuit-model simulator. The bottom plot is the simulated transfer response (in blue) and return coefficient (red) of the filter. The behavior is well in line with the lumped version shown on Figure 4; however, a spurious response is expected at 3.5 GHz on the microstrip version.

on the opposite layer. The only exception is when the length of the track is shor t, compared to the signal's wave length (e.g., a couple of millimeters, as in case impedance matching may be neglected).

A word of caution. Be careful if you use a multilayer PCB. The track width will need to be calculated based on the distance between the microstrip track and the ground plane, which is usually on the first inner layer. Ask your PCB supplier for the actual distance because this could be process-dependent. Also remember that the standard FR4 PCB substrate has a fuzzy dielectric constant (specified as ±10%) and high losses, which make it difficult to use when the working frequency exceeds a couple of gigahertz. Specific high-frequency substrates (e.g., Rogers RO4003) are much more efficient and well characterized, but this has a cost (see Figure 2).

DISTRIBUTED COMPONENTS

Now you know how to calculate the width of a PCB track to get a precise 50-Ω impedance. But what happens if you have a 50-Ω microstrip track and you increase or decrease its width on a small length? Remember the transmission line L/C model? If you decrease the track width, you will create a small section with a higher impedance, which is roughly equivalent to a serial inductance. And if you increase it, you will create a section with a lower impedance corresponding to a parallel capacitor to ground! Figure 3 shows how zero-cost L or C can be integrated on a microstrip design; it also shows how to calculate their values. Usually, the impedance of the small section is arbitrarily fixed to the largest or smallest value corresponding to a reasonable track width, often around 10 or 20 Ω for capacitors and 100 to 200 Ω for inductors. The track length calculation is based on this hypothesis and the desired component value thanks to the supplied formulas.

You need to be careful for two reasons. One, the dielectric coefficient used in the formulas is not the raw dielectric constant of the substrate found in the supplier's datasheets, but the effective dielectric constant of the microstrip wire, which is between the substrate and the air with a slightly different behavior. The formula and usual values are provided in Figure 2. For example, the dielectric constant of FR4 is around 4.5, but the effective dielectric constant of a microstrip on FR4 is 3.38.

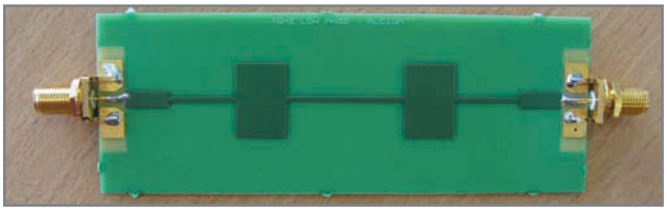


Photo 3—This is the assembled microstrip filter. From left to right you should recognize a small length of 50-Ω track (connected to the SMA connector), an inductor made with 14.62 mm of thin wire, a capacitor made of 10.48 mm of wide track, then a longer inductor, another capacitor, and a final inductor. Funny, isn't it?

Two, you must remember that I took the hypothesis of "small sections" of a track for this discussion. What does it mean in practical terms? Simply that the L/C model will be erroneous as soon as the track dimensions (width or length) are not significantly small compared to the wavelength of the signal you are working on. This wavelength is the speed of the signal on the PCB track divided by the working frequency. The signal speed is c (the speed of light, 3.10^8 m/s) divided by the square root of the effective dielectric constant. For example, this corresponds to a wavelength of 6.7 cm at 2.4 GHz on FR4. At that frequency, you can expect to have issues as soon as the component's dimension is larger than 1 cm or so. This fixes a limit to the value of the components you can design in microstrip form.

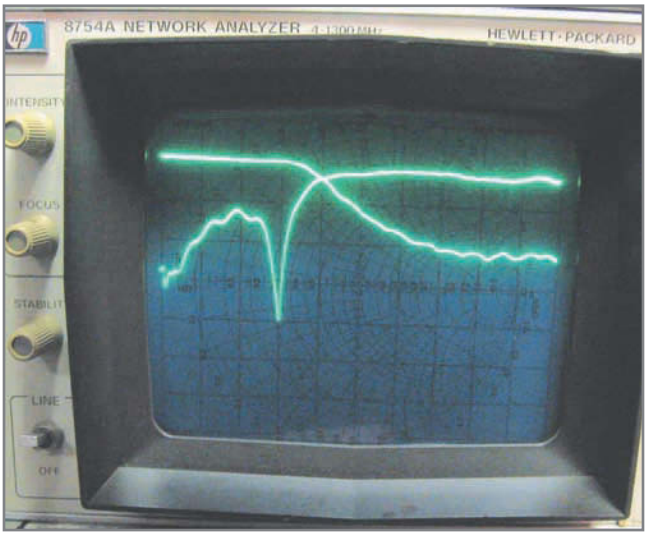


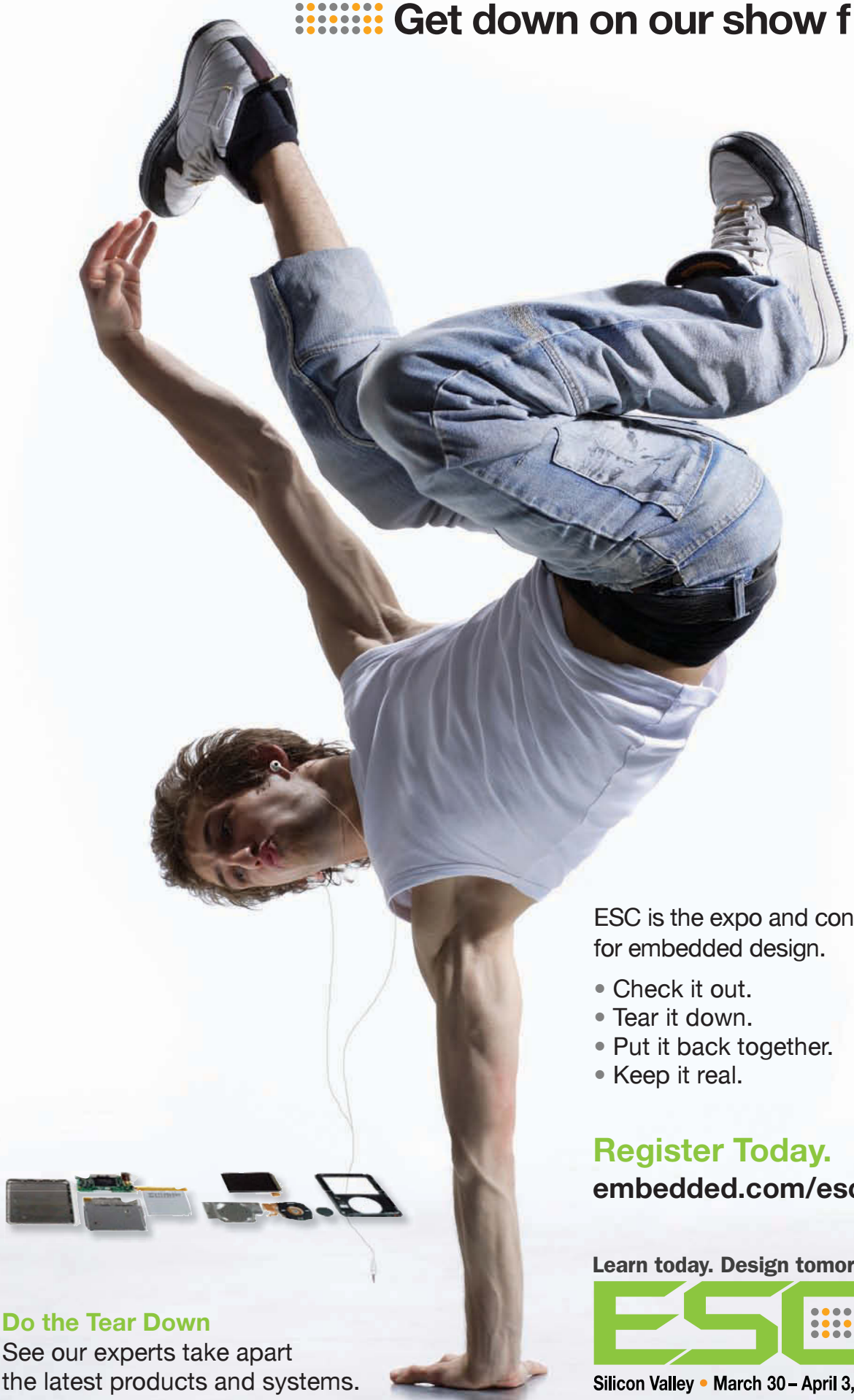
Photo 4—This is the measured response of the 1-GHz microstrip filter shown in Photo 1. Horizontal axis is from 8 MHz to 2.6 GHz. Curves are respectively transmitted and reflected coefficients with a 10-dB/division vertical scale. Impressively close to the simulation, with an attenuation of 20 dB from 1.5 to 2.6 GHz.

NEED A ZERO-COST FILTER?

Enough theory. Now it's time for some funny experiments. What could you do with L and C devices? You can try to design a 50-Ω 1-GHz low-pass filter with absolutely no discrete components. Such a filter would be made only with specific copper tracks on the PCB, so its cost would be virtually null, at least if you consider PCB surface as free. The first step is to design the filter as if you were using classical lumped components. Because I'm lazy from time to time, I used a free online filter calculator developed by Tony Fisher at The University of York. The result in Figure 4 includes three inductances (9.12, 15.7, and 9.12 nH) and two 4.36-pF capacitors. The 3-dB cut-off frequency is precisely 1 GHz. The calculated attenuation at 1.5 GHz is around 20 dB. You can build this filter using standard components. It will work if you are lucky enough to find a 9.12-nH inductance somewhere.

How can you transform these values into microstrip components? Let's assume that the PCB will be a standard 1.6-mm, double-sided FR4 substrate. Figure 2 indicates that the width of a 50-Ω track is a little less than 3 mm. To build the inductors, you need to use a thinner track with any arbitrary but convenient track width, say 100 Ω, which corresponds to a width of 0.678 mm. You then just have to calculate the corresponding lengths, thanks to the equations in Figure 3. With this 100-Ω track, the 9.12-nH value can be achieved with a track length of 14.62 mm, and 15.7 nH corresponds to 25.16 mm. Similarly, for the capacitors, you must select an arbitrary small impedance value, say 15 Ω, which corresponds to a track width of 15.2 mm, and the calculated track length for 4.36 pF is 10.48 mm. The final mandatory phase is to check that the largest dimension of the components is reasonably smaller than the wavelength of the highest frequency you are working with. Here, the largest length is the capacitor length, around 2.5 cm. You already calculated that 2.4 GHz was safe up to dimensions of 1 cm. So, you can expect the filter to work correctly at 1 GHz, but it may start to be a little

Get down on our show floor.



ESC is the expo and conference for embedded design.

- Check it out.
- Tear it down.
- Put it back together.
- Keep it real.

Register Today.
embedded.com/esc/sv

Learn today. Design tomorrow.

ESC

Silicon Valley • March 30 – April 3, 2009

Do the Tear Down
See our experts take apart the latest products and systems.

far from the predictions at frequencies of 2 to 3 GHz and higher.

The next step is to simulate the filter design. With RF designs, a simulator phase is always less expensive than some tens of PCBs thrown to the garbage bin. You have a couple of options for the simulator. The simplest solution would be to use a circuit-only simulator, which has provisions for microstrip models, such as the free, useful QUCS simulator. It is efficient and easy, but the disadvantage is that you won't get a drawing of the physical microstrip design. At the other extreme, you can use a full-featured 2-D or 2.5-D electromagnetic simulator, such as Sonnet Software, which has a free Sonnet Lite version that can be used for designs as simple as this one. Results will be accurate, but the tool's complexity and calculation time are significantly higher. I like an intermediate approach: PUFF. This old DOS-based simulator is based on circuit models rather than EM simulation, but it includes a pretty graphical input of the microstrip design. Moreover, it is nearly free because it comes with a couple of books. It runs well in a DOS box under Windows XP; but unfortunately, I wasn't able to make a screen copy, so you have only Photo 2. The layout windows are a graphical representation of the filter design, with a 50-Ω track on both ends and a succession of thin, inductive, and wide capacitive segments. The simulation took just 5 s on my standard PC and shows a good 1-GHz low-pass behavior but with a drastically reduced attenuation of around 3.5 GHz. A tool like PUFF can't explain why, but you already know the answer. At such a high frequency, the circuit dimensions are no longer "small" compared to the wavelength and it's likely a given track of the filter forms a tuned resonator at exactly 3.5 GHz. By the way, you can use a tool like Sonnet Lite to find it out. With the current density plot feature, a Sonnet design file is posted for your convenience on *Circuit Cellar*'s FTP site.

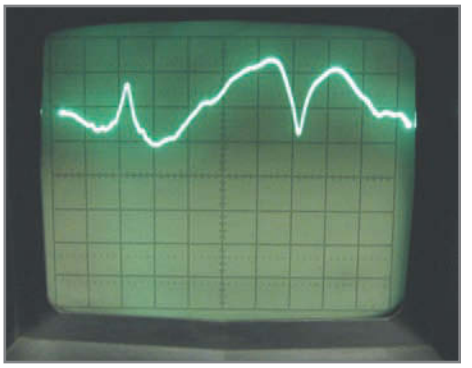


Photo 5—This is the transmitted coefficient of the 1-GHz filter on a higher frequency range. The horizontal scale is 2 to 8.4 GHz, vertical scale is still 10 dB/div with 0 dB at the top of the screen. The isolation starts at 20 dB at 2 GHz, but is significantly reduced at 3.47 GHz, in line with PUFF simulation. The filter is then unusable from 4 GHz upwards.

TO THE BENCH

I couldn't resist the pleasure of building and testing this filter. Look at the PCB in Photo 3. You should be able to recognize the successive L, C, L, C, and L sections. I soldered two SMA connectors for the test, and voila, the filter was ready. I hooked it on my old Hewlett-Packard HP8754A/H26 vectorial analyzer, which has a measurement range of 4 MHz to 2.6 GHz (see Photo 4). Quite nice, isn't it? I measured the 3-dB cut-off frequency at 1,030 MHz, close to the specification. The attenuation at 1.5 GHz is a reasonable 18 dB. I had to switch to a different test setup to evaluate the performances of the filter at frequencies above 2.6 GHz. I used an even older Hewlett-Packard HP8620C microwave sweeper with a 2- to 8.4-GHz plug-in connected to the input of the filter, and a Hewlett-Packard HP8755 scalar analyzer on its output. The result I got is in Photo 5. The first odd behavior was measured at 3.47 GHz with an unwanted peak response, exactly as expected through the PUFF simulation. At higher frequencies, the filter was no longer filtering anything. This was anticipated. Theoretical analysis showed that the filter dimensions were starting to be too large compared to the wavelengths.

I hope you are as pleased as I was when I saw that the actual performances were so close to the simulation, at least up to 4 GHz! I must admit that this was the first time I saw actual results so close to the simulation.

I was encouraged by this experiment, so I checked another aspect of microstrip circuits with the test PCB shown in Photo 6. Textbooks explain that a 90° turn on a microstrip track must have a precisely calculated chamfered corner to keep the impedance under control. This is understandable. If you design a nonchamfered 90° turn, there is some "extra" copper on the corner, which acts as a small capacitor, which impacts performances. My test PCB included two identical 50-Ω tracks on an FR4 substrate, one with chamfers and one without. I tested it with the same two test setups as the 1-GHz filter and was a little disappointed. There was no visible difference between both designs up to 4 GHz. However, I then tested them at a high frequency thanks to a 12.4- to 18-GHz plug-in I bought years ago for my HP8620C. I admit that trying to use an FR4 PCB at 18 GHz is a little risky, but it enabled me to see an impressive difference between the two versions. Refer to Photo 7 to see why you should always use chamfered corners on microstrips.

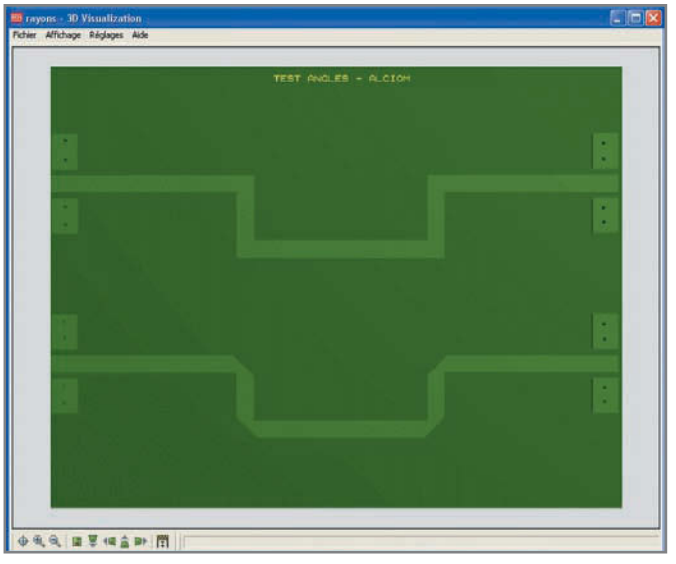


Photo 6—I designed this small PCB to check if chamfered corners on microstrips actually made a difference, as stated by the books.

WRAPPING UP

Obviously, microstrip designs are unnecessary for a standard low-speed microcontroller design. However, knowledge of PCB track impedance matching rules is mandatory when dealing

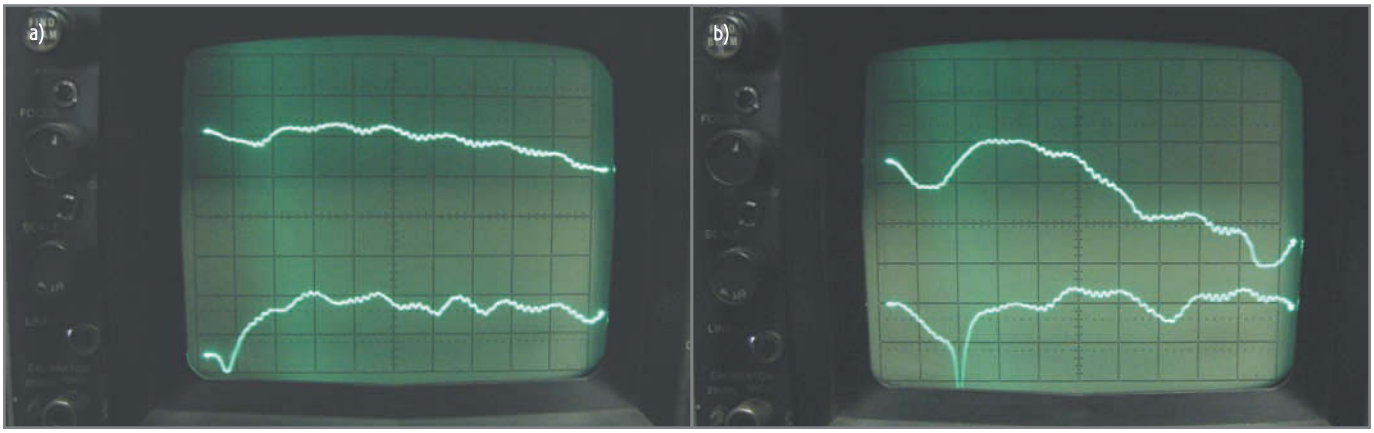


Photo 7—The top curves of each plot show the measured transfer coefficient of the respective chamfered (a) and unchamfered (b) microstrip lines at high frequencies, from 12.4 to 18 GHz. Vertical scale is 5 dB/div with a 0-dB one division below the top. The difference was not visible at lower frequencies but here it is impressive. With the chamfered version, the loss is around 5 dB, slowly increasing to 10 dB at 18 GHz. However, with the unchamfered one, the loss is far more erratic and going up to –20 dB. Ignore the bottom curves of each plot.

with a large number of high-speed digital boards. Your gigahertz-clocked PC probably won't work without impedance-matched tracks. And, of course, their use is mandatory for RF designs.

"At which minimal frequency should I start to be worried about track impedance matching?" That's a common question. The easiest way is to compare the wavelength of the signal and the length of the longest track on your PCB. If the wavelength is far longer than your PCB dimensions, then you can usually safely ignore track matching. However, if you have a track that starts to be close to the wavelength you are working on, then you'd better take care and use impedance matching. Take the example of a classic 10-cm track on FR4. If you consider "safe" as a factor of 10, then you should use matched tracks as soon as the wavelength on FR4 is shorter than 1 m (i.e., 10 × 10 cm), which corresponds to a wavelength in the air of 1 m times the square root of the 3.38 effective dielectric constant. The result is 1.84 m, which corresponds to a frequency of 300,000,000/1.84 = 163 MHz. Do you use signals faster than about 150 MHz? If so, I hope that microstrips are no longer on the darker side for you.

Robert Lacoste lives near Paris, France. He has 18 years of experience working on embedded systems, analog designs, and wireless telecommunications. He has won prizes in more than 15 international design contests. In 2003, Robert started a consulting company, ALCIOM, to share his passion for innovative mixed-signal designs. You can reach him at rlacoste@alciom.com. Don't forget to write "Darker Side" in the subject line to bypass his spam filters.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/223.

RESOURCES

D. Brooks, "Embedded Microstrip: Impedance Formula," *Printed Circuit Design*, 2000.

T. Lee, *Planar Microware Engineering: A Practical Guide to Theory, Measurement, and Circuits*, Cambridge University Press, Cambridge, U.K., 2004.

The University of York Department of Computer Science, "LC Filter Design calculator," www-users.cs.york.ac.uk/~fisher/lcfilter.

SOURCES

AppCad
Agilent Technologies, Inc. | www.hp.woodshot.com

PUFF Microstrip layout and simulator
California Institute of Technology RF and Microwave Group
www.its.caltech.edu/~mmic/puff.html

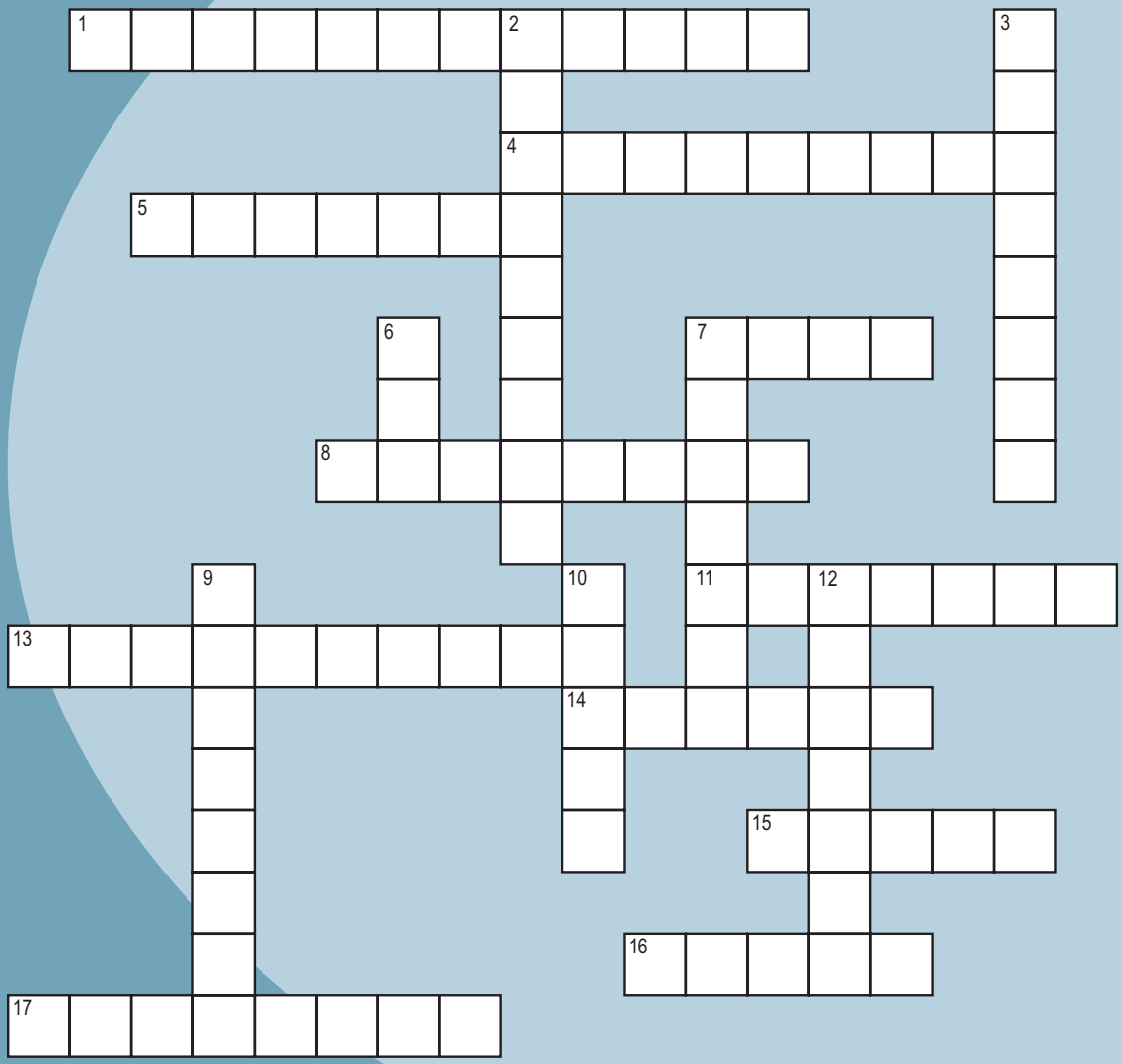
HP8620C Microwave sweeper, HP8754A/H26 vector network analyzer, and HP8755 scalar analyzer
Hewlett-Packard | www.hp.com

Qucs project
Qucs team | http://qucs.sourceforge.net

RO4003 Substrate
Rogers Corp. | www.rogerscorp.com

Sonnet Lite
Sonnet Software, Inc. | www.sonnetsoftware.com/products/lite

CROSSWORD



Across

1. 1 billion watt-hours
4. Gain of oxygen
5. XTAL on a schematic
7. A built-in assembler
8. A release with changes
11. .ENC
13. Unused radio frequencies
14. Base-2
15. American scientist (1797–1878) noted for his work with electromagnets and self-inductance. He invented the relay.
16. 10^{21}
17. The return of a device's output back to the input

Down

2. A set of linked programs
3. Horizontal line; 0.33333…
6. How many miles per hour is 1.609 km/h?
7. Operators: NOT, AND, OR
9. A pr ocessor with no I/O connected to one that does
10. Remove errors
12. Electric charge in amps

The answers are available at www.circuitcellar.com/crossword.

Accelerate your Design Time with EmbeddedDeveloper.com

Speed up your design time during the critical evaluation phase of your project by avoiding locating and comparing devices from different manufacturers. We make it easy: just one visit to **EmbeddedDeveloper.com** will shave hours or days off your schedule!

Embedded Developer's simple navigation and intuitive search engines help you quickly locate, compare, and evaluate thousands of different microcontrollers from different manufacturers--without knowing part numbers.

Compare all devices by their performance and features in seconds, download a data sheet, find and buy development tools, or link to a distributor for an RFQ or chip sample--without ever leaving the site!

Embedded Developer is the only site in the world where you're only a few clicks away from buying the right device or tool for your next job. So fasten your seat belts and log-in today--we guarantee this site will put you in pole position in the time-to-market race!

HEARST *business media*
ELECTRONICS GROUP

EMBEDDEDDEVELOPER.COM
FIND. COMPARE. BUY.

ConVergence PROMOTIONS

IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

AD FORMAT: Advertisers must furnish digital submission sheet and digital files that meet the specifications on the digital submission sh eet. **ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2** ″ x **3**″ **FORMAT**. Call for current rate and deadline information. E-mail adcopy@cir cuitcellar.com with your file and digital submission or send it to IDEA BOX, Cir cuit Cellar , 4 Park Street, V ernon, CT 06066. For more information call Shannon Barraclough at (860) 875-2199.

The Vendor Directory at www.circuitcellar.com/vendor/
is your guide to a variety of engineering pr oducts and services.

INDEX OF ADVERTISERS

The Index of Advertisers with links to their web sites is located at www.circuitcellar.com under the current issue.

Page		Page		Page		Page	
75	AAG Electronica, LLC	29	ExpressPCB	67	Lakeview Research	53	PROPOX Sp. z o.o.
12	AP Circuits	27	ezPCB	77	Lawicel AB	74	Pulsar, Inc.
22	APEX Expo	74	FDI-Future Designs, Inc.	13, 76	Lemos International	28, 42	Rabbit, A Digi International Brand
75	All Electronics Corp.	75	FlexiPanel Ltd.	74	LinkSprite Technologies, Inc.	76	Rabbit, A Digi International Brand
76	Apex Embedded Systems	12	Front Panel Express LLC	32	Linx Technologies	76	Reach Technology, Inc.
7	Atmel	28	Geist Technology	77	MCC (Micro Computer Control)	76	Renesas Technology
33	CWAV	19	General Circuits	23	MachinePIER	39	Saelig Co.
21	CadSoft Computer, Inc.	53	Grid Connect, Inc.	5	Microchip	11	SEGGER Microcontroller Systems LLC
34	Calao Systems	3	HI-TECH Software LLC	78	microEngineering Labs, Inc.	C3	Tech Tools
77	CapTron Corp.	55	HobbyLab LLC	56	Mouser Electronics	48, 49	Technologic Systems
16	Comfile Technology, Inc.	41, 63	ICbank Inc.	74	Mylydia, Inc.	75	Technological Arts
75	Crossware Products, Inc.	77	IMAGEcraft	C2	NetBurner	77	Tern, Inc.
23, 76	Custom Computer Services, Inc.	1	Imagineering, Inc.	67	Nurve Networks LLC	77	Tin Can Tools, LLC
74	DLP Design	78	Ironwood Electronics	75	Ontrak Control Systems	78	Trace Systems, Inc.
55	DesignNotes	32, 34	JKmicrosystems, Inc.	68	PCB-Pool	75	Triangle Research Int'l, Inc.
13	EMAC, Inc.	75	JKmicrosystems, Inc.	C4	Parallax, Inc.	65	Trinity College Robot Contest
76	Earth Computer Technologies	2	Jameco	77	Phytec America LLC	9	WIZnet
73	Embedded Developer	67	Jeffrey Kerr, LLC	75	Pioneer Hill Software		
69	Embedded Systems Conf.	17	Keil Software	10	Pololu Corp.		

PREVIEW of March Issue 224

Theme: **Robotics**

Vision-Guided Robotics: Build a Next-Generation Balancing Robot

Wireless Mobile Robotics

Robot Navigation and Control (Part 1): Construct a Navigation Control Subsystem

Digital Decoding: A Design for Decoding Periodic Signal Transmissions

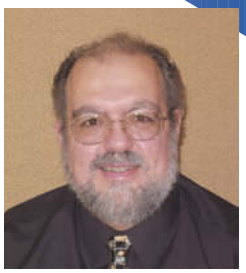
Networked Timing: A Timer with Advanced Planning Tools

LESSONS FROM THE TRENCHES **FAT File System Review (Part 1):** Open Files & Perform Operations

FROM THE BENCH **Programmable Robotics (Part 1):** Build on an Existing Robot Platform

SILICON UPDATE **A Really Simple Plan:** The "8-Bits" Saga Continues

PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

Are We Dumb Enough Yet?

OK, this is a rant, but I have to clarify that it isn't aimed at you. It's aimed at everybody else. ;-)

Seriously, if you are reading *Circuit Cellar*, then you have already passed the selection criteria: you're smart, you don't accept spoon-fed techno-crap, and you are not a slouch. Let's face it, *Circuit Cellar* is only casual reading for a very educated and select minority, and that's just fine with me.

Unfortunately, I wonder about the rest of society. Am I in the minority, or are any of you getting a lot more stupid questions these days too? You would think after 15 to 20 years of owning and using computers on a daily basis that people would understand them better. Lots of them, including some of my relatives and friends, still ask naive questions. Examples include: Does anybody besides Microsoft make a word processor? Why do I need antivirus software? How many pictures can you fit on an SD card? Certainly, such simple questions can be effortlessly answered with a two-minute web search.

OK, I'm an old guy these days, but the younger the person (in decades) that I speak to, the less basic knowledge they seem to have. They know every popular band and song artist, but don't ask them what continent Venezuela is on or how many meters are in a kilometer. It's like facts aren't important because knowledge is so readily accessible via the web that they don't feel the need to retain it "offline." Unfortunately, I think this "spoon feeding" started a long time ago and is endemic to our dumbed-down western culture. Rather than research subjects that expand individual knowledge, people too often simply view individual answers as data bits with short-term application and no need for retention. Asking me how to do something or how something works is like going to Wikipedia, only with less work on their part. I even have friends who'll e-mail me asking how to connect a new piece of electronic gear when it's obvious that they had to move the unread installation manual out of the way to type the e-mail.

A lot has been said about the demise of print media, particularly newspapers, because people are supposedly getting their information on the Internet. But I wonder if much of the demise is self-inflicted. I believe that popular media has become so dumbed down and sound-bite-centric that it has completely dropped the ball on in-depth coverage. The marketing departments have brainwashed the editorial people to focus on an early twenties-type readership who all have attention deficit disorder but, fortunately, a fist full of credit cards. In the meantime, they lose white-collar college graduates who are tired of the complete lack of unbiased editorial and technical frankness—euphemisms like "specialized counting system" instead of just saying "hexadecimal." Heaven forbid a reader might have to go figure out what a word meant and learn something.

Broadening the language and taking out the technical words is nothing new. I don't expect that newspapers will emulate *EE Times* or *Circuit Cellar* in their descriptions, but it's interesting that even when they publish articles in their technology sections about the perceived dumbing down of technology awareness, they still write everything as though we're morons. No wonder many of us look for news resources elsewhere.

Sometimes I feel like we are an isolated minority, but perhaps technical people have always had this burden. We're valued greatly when it comes to the intelligence that spawns technical revolution, but we're disparaged when we criticize the cultural degradation that comes with mass use of it. Language seems like an insignificant issue, but if correct computer terminology isn't part of the public language, then people will remain dummies. If average readers keep seeing "specialized counting system" they learn nothing. If they see acronyms such as DRAM and USB, or a term such as multithreaded programming when they read about computers, maybe they'll make some effort to figure out what it all means. Certainly, there are many newspaper editors who will vehemently disagree with me, but ask how many of them want their IBM Selectrics back before factoring in their criticism. I have my biases too, but some of these guys must surely hate computers.

I'd like to say that things are getting better, but continually lowering the "cognitive" bar to attract a wider readership only makes it worse. I can't speak for all of western society, but it's starting to look like the American public's brain is expected to run in neutral most of the time. The only bright light on the horizon has been the computers themselves. Like the cars that replaced horses and the bulldozers that replaced shovels, it looks like we continually strive to develop machines to replace manual tasks. This time, however, we are expecting them to expand the heavy lifting to include doing our thinking too.

Along with me, I predict that a large group of independent thinkers in this audience will view this future with disdain. Unfortunately, the majority of the public will be happy as a clam. Of course, it also means they have the IQs to match.

steve.ciarcia@circuitcellar.com

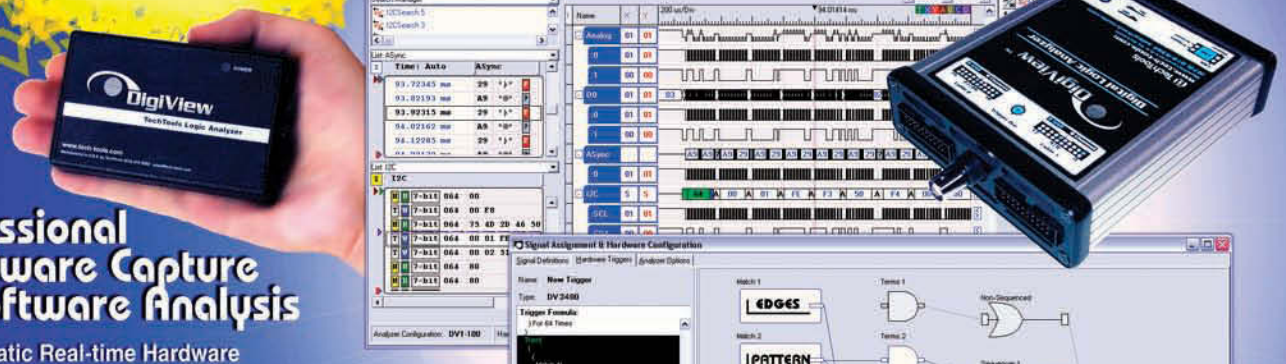
PC Based Logic Analyzers

NEW Model DV3400

- Faster Sampling (200/400 Msps)
- More Channels (36)
- More Memory (4x)
- Advanced Match circuits (8)
- ±6V Adjustable Threshold (x2)
- Cascadable Sequencers (4)
- Enhanced Compression

from $499.00

DigiView



Professional Hardware Capture + Software Analysis

- Automatic Real-time Hardware Compression eliminates the need to reduce resolution. Our newest version makes "dead time" insignificant.
- Edge and Pattern Triggers on all models. Advanced Model also includes Range (>, <, =, !=, >=, <=) and Stable Matches with Duration, and 4 flexible cascadable sequencers with pass-counters.
- Pattern Searches with Match & Duration.
- Specialized Sequential Searches for Serial and State Mode Signals.
- Display I²C, Synchronous (SPI), Asynchronous (RS-232), State, Boolean, Bus and Analog Data.
- Single or Dual Waveform Views.
- Resolution Zoom in Wave Views.
- Time based Link Groups for all views.
- Specialized Exports from Data Tables and List Views.
- Multi-Signal Data Tables.
- Drag & Snap Markers.
- "Click to Center" function.
- "Snap Previous/Next" function.
- Print or Save Images with comments.
- USB 2.0 (480 Mbps).
- USB 1.1 compatible (12 Mbps).

Very Flexible, "easy to configure" Advanced triggering.

PICmicro® MCU Programmer

Multi-Function, In-Circuit & Gang Operation

only $199.00

QuickWriter™

See our Memory Emulators from $179.00

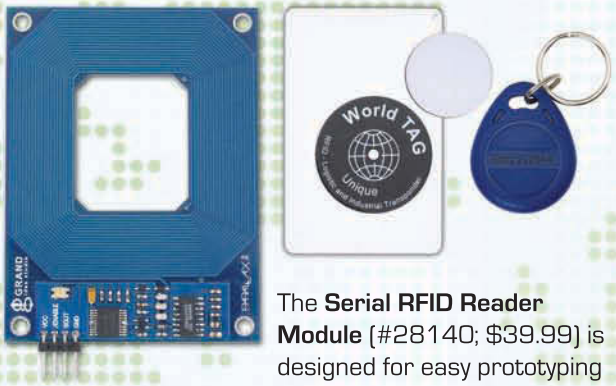
TechTools

www.tech-tools.com

(972) 272-9392 • sales@tech-tools.com

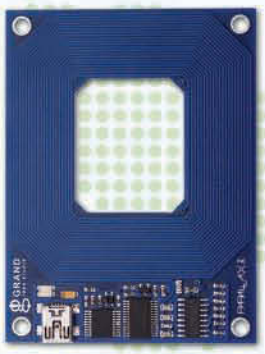
.....World of Wireless.....

Enhance your latest electronic endeavor with these wireless communication modules from Parallax. Full product details, downloads, and project ideas available at www.parallax.com.



The **Serial RFID Reader Module** (#28140; $39.99) is designed for easy prototyping and integration into stand-alone microcontroller-based applications. It can be used in a variety of hobbyist and commercial applications, including access control, automatic identification, robotics, navigation, inventory tracking, payment systems, and car immobilization.

We also offer a variety of transponder tag styles (sold separately); visit parallax.com for details.



The **USB RFID Reader Module** (#28340; $49.99) connects directly to a USB port both for power and to output transponder tag data to your custom PC application. 2400 baud serial output stream to PC is accessible from any programming language that can open a COM port.

Our RFID sampler kits include the reader module and an assortment of four RFID transponder tags.

- **Serial RFID Reader Sampler Kit** (#32390; $45.99)
- **USB RFID Reader Sampler Kit** (#32395; $55.99)

The **EmbeddedBlue Transceiver AppMod** (#30068; $79.99) provides standard Bluetooth connectivity for Parallax microcontroller applications without the need for detailed Bluetooth knowledge. 2x10 pin format plugs into the AppMod header on the Board of Education or Super Carrier Board (sold separately).

The **Parallax 433 MHz RF Transmitter and Receiver** (#27980; $29.99 and #27981; $39.99, respectively) provide a very easy and low-cost method of sending data between microcontrollers or to a PC from 500 ft+ line-of-sight (depending on conditions). Also available as a 4 module **Transceiver Package** (#28180; $119.99), a $20 savings.

The **(PINK) Parallax Internet Netburner Kit** (#30013; $129.99) is an embedded web server which enables your microcontroller to communicate over a network or the Internet. The PINK module allows your microcontroller to send e-mail or UDP messages to alert you to events or provide status messages. Simple TTL serial interface.

Order **Parallax Communication Modules** at www.parallax.com or call our Sales Department toll-free at 888-512-1024 (Mon-Fri, 7 a.m. - 5 p.m., PST).

PARALLAX INC

www.parallax.com