

#198 January 2007

www.circuitcellar.com

# CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

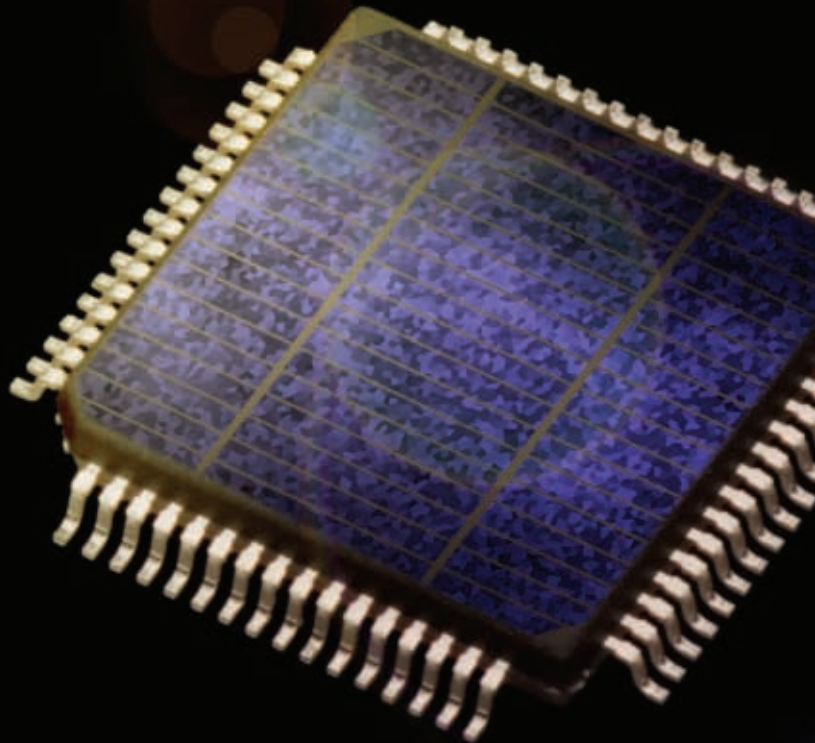
## EMBEDDED APPLICATIONS

Self-Powered Data Logger

Build an Arbitrary  
Waveform Generator

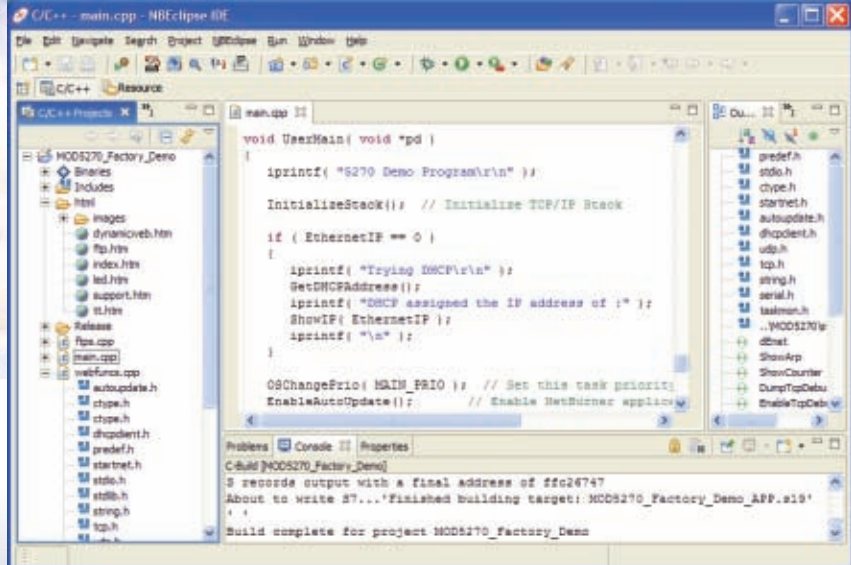
New Uses for Flash  
Memory

Voltage Solutions



# The NetBurner Eclipse Ethernet Development Kit

Only **\$99**



## Featuring NetBurner's Eclipse IDE!

One Click Compile and Load | Intelligent Code Completion | Integrated Debugger

## The Complete Hardware and Software Solution

Includes NetBurner MOD5270 Ethernet Core Module

**Performance and Memory** | 32-Bit CPU | Freescale ColdFire MCF5270 147Mhz | 512K Flash Memory | 2MB SDRAM

**Device Connectivity** | 10/100 Ethernet | 3 UARTs | I<sup>2</sup>C | SPI | 47 Digital I/O | SD/MMC Flash Card Support

### Communication Software

| TCP/IP Stack | HTTP Web Server | FTP | E-Mail | PPP | Flash File System

### Development Software

| NB Eclipse IDE | Graphical Debugger | Deployment Tools | Examples

### System Software

|  $\mu$ C/OS RTOS | ANSI C/C++ Compiler and Linker



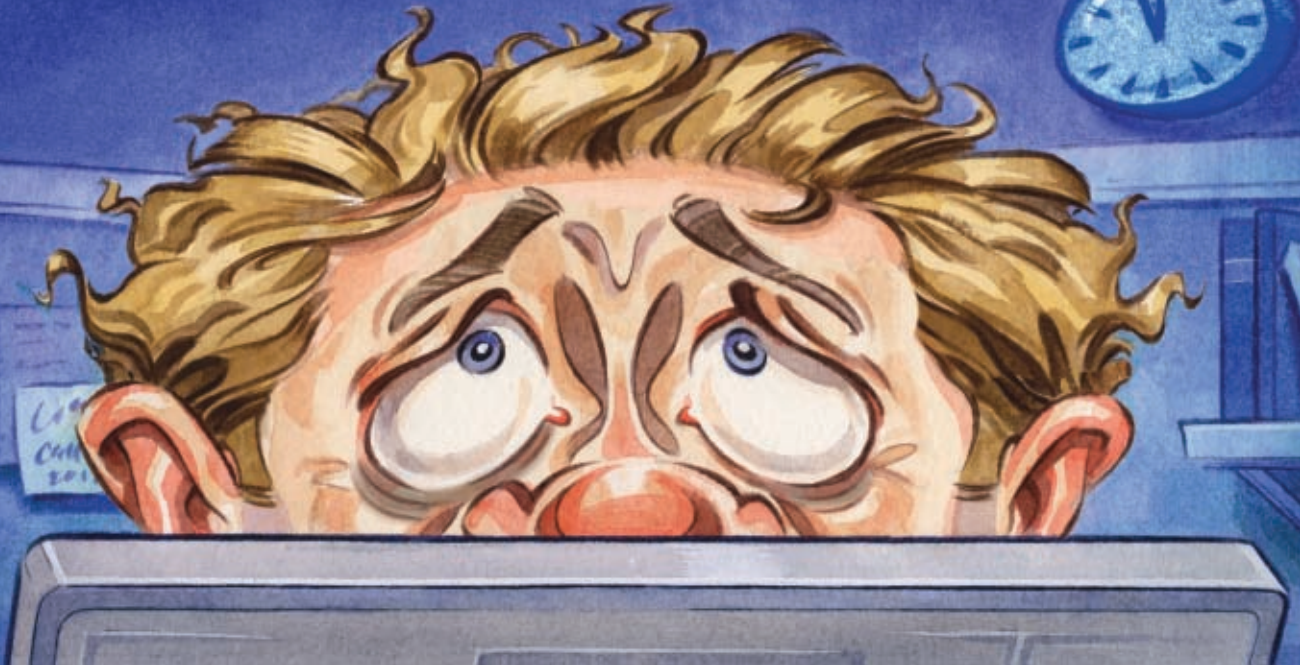
**netburner.com**



Product No. | NNDK-MOD5270LC-IT  
Information and Sales | sales@netburner.com  
Telephone | 1-800-695-6828

# No code? No way.

Focus on your design and leave the coding to our PSoC Express™ visual embedded design tool.



Without writing a single line of assembly or “C” code, generate a complete custom design with PSoC Express. Here’s what you get with Version 2.1 of our visual embedded software:

- Support for our family of Programmable System-on-Chip™ (PSoC®) mixed-signal arrays—powerful, programmable digital and analog blocks with integrated MCU and flash memory
- Rich visual environment with simulation enables you to see your design and evaluate its performance instantly
- Built-in support for interdevice communication; seamlessly divide design problems into smaller pieces using multiple PSoC devices
- Retarget to any PSoC mixed-signal array at any time; design first and select device later

## TRY IT OUT NOW.

Whether you have a minute or an hour to invest, we have a way to get you working with PSoC Express:

- View our online PSoC Express demo:  
[www.cypress.com/expressdemo](http://www.cypress.com/expressdemo)
- Request a FREE PSoC Express evaluation kit:  
[www.cypress.com/expresskit](http://www.cypress.com/expresskit)
- Test drive PSoC Express at a live seminar near you:  
[www.cypress.com/expressour](http://www.cypress.com/expressour)
- Download FREE PSoC Express software:  
[www.cypress.com/psocexpress](http://www.cypress.com/psocexpress)

Make all your embedded designs fast and easy.  
[www.cypress.com/getexpress](http://www.cypress.com/getexpress)





# Link Instruments

PC-Based Test Equipment

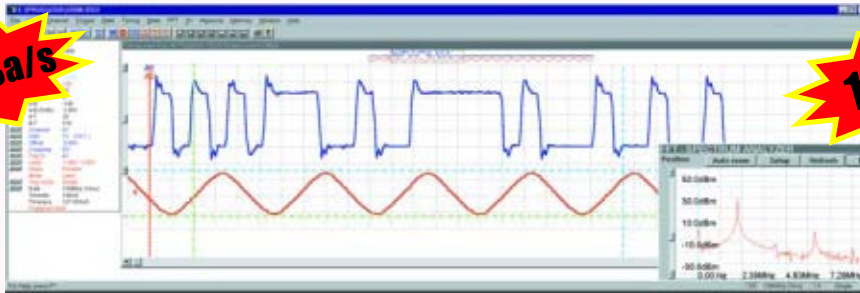
## Digital Oscilloscopes

**NEW!**



- 2 Channel Digital Oscilloscope
- **500 MSa/s** max single shot rate
- 1Mpt sample memory
  - 250 MSa/S (Dual channel) 512 Kpts
  - 500 MSa/S (Single channel) 1 Mpts
- Advanced Triggering
- Only 9 oz and 7" x 3.5" x 1.5"
- Portable and Battery powered
- USB 2.0
- Advanced Math
- FFT Spectrum Analyzer
- Priced at ~~\$950~~ **Introductory Price \$850**

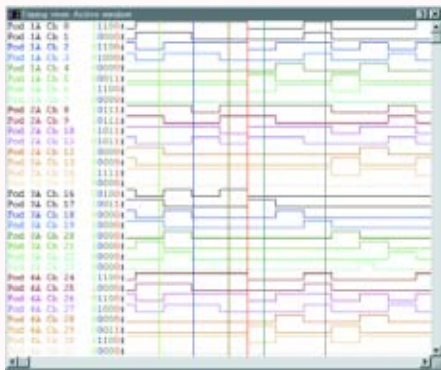
**500MSa/s**



**1Mpts**

Windows Screenshot

## Logic Analyzers



Windows Screenshot



- 40 to 160 channels
- up to 500 MSa/s
- Variable Threshold
- 8 External Clocks
- 16 Level Triggering
- up to 512K samples/ch
- USB 2.0 and Parallel Interface
- Pattern Generator option

LA5240 (200MHz, 40CH)	<b>\$1700</b>
LA5280 (200MHz, 80CH)	<b>\$2350</b>
LA5540 (500MHz, 40CH)	<b>\$2500</b>
LA5580 (500MHz, 80CH)	<b>\$3500</b>
LA55160 (500MHz, 160CH)	<b>\$7500</b>



Link Instruments (973) 808-8990  
17A Daniel Road East · Fairfield, NJ 07004 · Fax (973) 808-8786

[www.Linkins4.com](http://www.Linkins4.com)

# RabbitFLEX

## A New Way To Customize



- Click-to-ship in 5 days!
- Pay only for what you need
- Revision friendly
- Perfect for prototype and production



RabbitFLEX™ is an unique build system that gives you the power to develop custom boards without the hassle and the cost. The RabbitFLEX simple-to-use web interface allows you to choose from numerous options such as digital I/O, analog I/O, serial ports, and Ethernet connections on your custom board. Just configure and buy online and our patent pending manufacturing process will deliver your solution in a matter of days. With RabbitFLEX you will reduce design risk, manufacturing cost, and development time.

Start developing now by ordering the RabbitFLEX Tool Kit and your own custom RabbitFLEX board. Take your solution to the next level.

Configure and Buy Online  
[www.rabbitFLEX.com](http://www.rabbitFLEX.com)

Quick Turn Boards Range From  
**\$149-\$279**

RabbitFLEX Tool Kit  
**\$199**

### Test Drive RabbitFLEX

Build your custom RabbitFLEX board online. Add a tool kit to your order for a complete development system including Dynamic C®.

[www.rabbitFLEX.com](http://www.rabbitFLEX.com)



For a limited time with kit purchase.



2900 Spafford Street, Davis, CA 95616 Tel 530.757.8400  
**Solutions That Work**

# TASK MANAGER

## Agency vs. Contingency

When I was a student in London in 1998, I took an interesting sociology course titled Political Processes and Social Change. In his opening lecture, the course convener asked us a seemingly simple question: Is social change driven by contingencies or by the actions of human agents? Ah, yes, the old Agency vs. Contingency debate. Theorists who focus on the former tend to argue that individual actors such as Franklin D. Roosevelt, Winston Churchill, and Albert Einstein move history. In contrast, proponents of the latter theory argue that contingencies—such as pandemics and major fluctuations in the international markets—drive change.

What do you think, particularly in terms of the path on which the embedded community is traveling? Will the historians and social scientists of the future argue that it was the work of a handful of individual agents—particular designers, researchers, programmers, and corporations—that drove technological change, or will they write that various contingencies—such as macroeconomic changes in the technology sector—had the most impact?

Considering such questions will help you figure out where you fit in the puzzle. Do you believe a handful of engineers will develop the devices that will change the technological landscape, or do you think that exogenous social and economic forces will be main factors? Are you waiting for the next Bill Gates or market crash to determine your access to new technologies and your ability to design new systems?

Perhaps you should consider a completely different theory. The way we see it here at *Circuit Cellar*, the collective spirit and effort of the design community will lead the way. No matter which technologies, personalities, and socioeconomic variables (whether positive or negative) enter the formula for progress, the design community will remain steadfast in its determination to move forward and come up with novel ideas. If a young engineer develops a groundbreaking technology, the community will use it to its advantage. If the market should crash and hardware, software, and financial resources suddenly become scarce, the members of the community will work together to figure out ways around the problems. Sure, the speed at which the community will develop new technologies will change every year, but it's unlikely that any one person or event will derail the train. Are you on board?

It's exciting to see that many of you are now working harder than ever to develop your ideas and present them to your peers. Moreover, we're glad that you are actively addressing the social and environmental issues that are facing us in the 21<sup>st</sup> century. For instance, Abigail Krich, an enthusiastic designer who recently graduated from Cornell University's graduate program in electrical engineering, describes how she built a self-powered solar data logger (p. 12). She uses the system to measure solar insolation levels. This project proves that all of you can design effective systems that better the environment and society at large.

Columnist Jeff Bachiochi tackles another timely issue: alternative power sources (p. 56). He provides you with some tips on leveraging the power of "green" energy. Now is a great time to start thinking about the ways in which you can make your designs more environment-friendly and power efficient.

One last note: We're happy to announce the return of George Martin's Lessons From the Trenches column. This month, he begins a new series of articles about C language (p. 60). His articles will appear every other month.

cj@circuitcellar.com

# CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

**FOUNDER/EDITORIAL DIRECTOR**  
Steve Ciarcia

**CHIEF FINANCIAL OFFICER**  
Jeannette Ciarcia

**MANAGING EDITOR**  
C.J. Abate

**MEDIA CONSULTANT**  
Dan Rodrigues

**WEST COAST EDITOR**  
Tom Cantrell

**CUSTOMER SERVICE**  
Debbie Lavoie

**CONTRIBUTING EDITORS**  
Jeff Bachiochi  
Ingo Cyliax  
Fred Eady  
George Martin  
Ed Nisley

**CONTROLLER**  
Jeff Yanco

**ART DIRECTOR**  
KC Prescott

**GRAPHIC DESIGNER**  
Mary Turek

**NEW PRODUCTS EDITOR**  
John Gorsky

**STAFF ENGINEER**  
John Gorsky

**PROJECT EDITORS**  
Steve Bedford  
Ken Davidson  
David Tweed

## ADVERTISING

860.875.2199 • Fax: 860.871.0411 • [www.circuitcellar.com/advertise](http://www.circuitcellar.com/advertise)

### PUBLISHER

Sean Donnelly  
Direct: 860.872.3064, Cell: 860.930.4326, E-mail: [sean@circuitcellar.com](mailto:sean@circuitcellar.com)

### ADVERTISING REPRESENTATIVE

Shannon Barraclough  
Direct: 860.872.3064, E-mail: [shannon@circuitcellar.com](mailto:shannon@circuitcellar.com)

### ADVERTISING COORDINATOR

Valerie Luster  
E-mail: [val.luster@circuitcellar.com](mailto:val.luster@circuitcellar.com)

Cover photography by Chris Rakoczy—Rakoczy Photography  
[www.rakoczyphoto.com](http://www.rakoczyphoto.com)

PRINTED IN THE UNITED STATES

## CONTACTS

### SUBSCRIPTIONS

**Information:** [www.circuitcellar.com/subscribe](http://www.circuitcellar.com/subscribe), E-mail: [subscribe@circuitcellar.com](mailto:subscribe@circuitcellar.com)  
**Subscribe:** 800.269.6301, [www.circuitcellar.com/subscribe](http://www.circuitcellar.com/subscribe), Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650  
**Address Changes/Problems:** E-mail: [subscribe@circuitcellar.com](mailto:subscribe@circuitcellar.com)

### GENERAL INFORMATION

860.875.2199, Fax: 860.871.0411, E-mail: [info@circuitcellar.com](mailto:info@circuitcellar.com)  
**Editorial Office:** Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: [editor@circuitcellar.com](mailto:editor@circuitcellar.com)  
**New Products:** New Products, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: [newproducts@circuitcellar.com](mailto:newproducts@circuitcellar.com)

### AUTHORIZED REPRINTS INFORMATION

860.875.2199, E-mail: [reprints@circuitcellar.com](mailto: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.

Circuit Cellar® makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled projects, Circuit Cellar® disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published by Circuit Cellar®.

The information provided by Circuit Cellar® is for educational purposes. Circuit Cellar® makes no claims or warrants that readers have a right to build things based upon these ideas under patent or other relevant intellectual property law in their jurisdiction, or that readers have a right to construct or operate any of the devices described herein under the relevant patent or other intellectual property law of the reader's jurisdiction. The reader assumes any risk of infringement liability for constructing or operating such devices.

Entire contents copyright © 2006 by Circuit Cellar, Incorporated. All rights reserved. Circuit Cellar is a registered trademark of Circuit Cellar, Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.



# R8C/Tiny Brings 16-bit Performance to 8-bit Applications

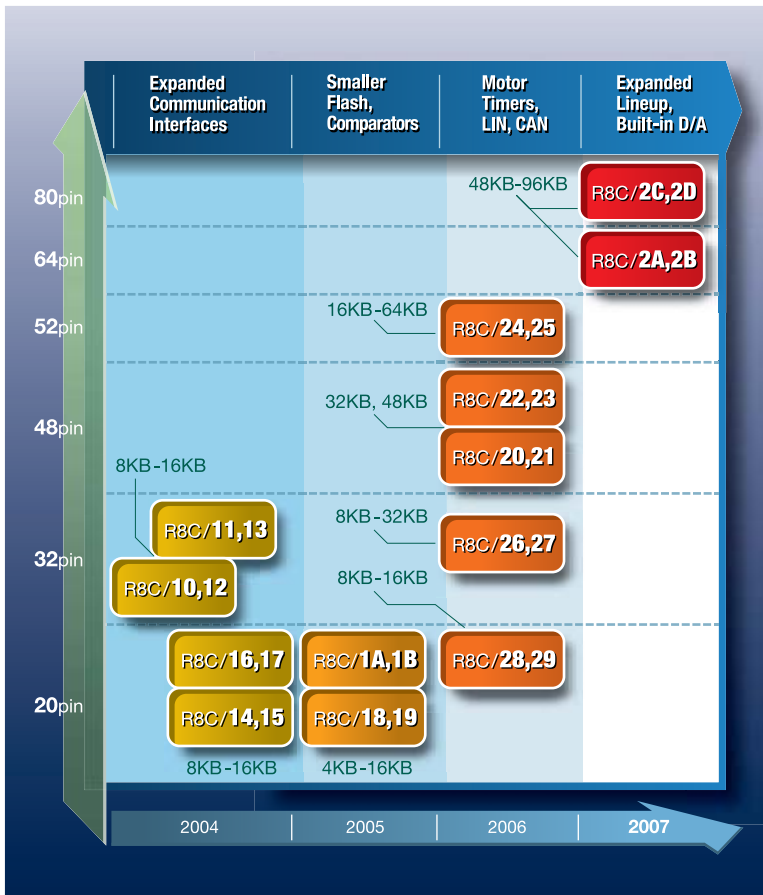
High-performance: 40MHz on-chip oscillator and single-cycle flash memory access

## Renesas Technology

No. 1\* supplier of microcontrollers in the world

introduces the R8C/Tiny Series of microcontrollers. Its powerful 16-bit CPU core running at 20MHz provides the performance never imagined in 8-bit MCUs. R8C/Tiny MCUs high level of integration and peripheral set will enhance your application's functionality while reducing the overall system cost.

### R8C/Tiny Product Roadmap



### HOT Products R8C/25

<b>R8C/Tiny CPU</b>	Program Flash (16KB - 64KB)	RAM (1KB - 3KB)
Oscillation Circuit Main Clock (20MHz Max.)	Data Flash (2KB)	Power-On Reset Circuit
On-chip Oscillators (40MHz, 125KHz)	Low Voltage Detect Circuit	Protect Register
Oscillation Circuit Sub Clock (32KHz Clock)	Enhanced WDT	16-bit motor control Timer (2)
RTC	External Oscillation Stop Detection	8-bit Timer (3)
Serial I/O Clock Sync./ UART (2ch)	SSU/I2C	A-D Converter (10-bit x 12ch)
44 GPIO	Hardware LIN	On-Chip Debug

Package: 52pin LQFP (10mm x 10mm, 0.65mm pitch)

### Top Reasons To Select R8C/Tiny

- High-Performance**
  - 16-bit CPU runs at 20MHz, executing instructions in as fast as 50nsec.
- Scalable**
  - Same core, same peripherals allows easy portability from 20 to 80pins
- High-Integration**
  - Includes 40MHz on-chip oscillator, data flash, power-on reset circuits, several 8- and 16-bit timers, up to 20 channels of A/D, D/A, LIN, CAN and more
- Reliable**
  - Oscillator stop detection circuits, access control of system registers
  - Watchdog timer with on-chip oscillator, programmable low-voltage detect circuit
- World-class Development Environment**
  - Complete software and hardware tools for short development cycle
  - Free 64KB software tool chain

\*Source:Gartner Dataquest (April 2006) \*2005 Worldwide Microcontroller Vendor Revenue" GJ06333



## Get Started Today -

Go online and register to be eligible for a FREE Starter Kit

[www.america.renesas.com/ReachR8C/d](http://www.america.renesas.com/ReachR8C/d)

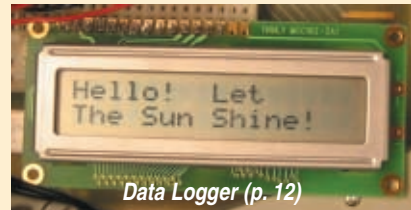


Renesas Technology Corp.

# January 2007: Embedded Applications

## FEATURES

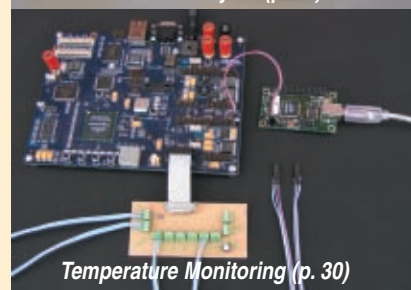
- 12 **Self-Powered Solar Data Logger**  
*Abigail Krich*
- 20 **QuickComs**  
An MC16C/62P-Based RS-232 Analyzer  
*Nick Lott*
- 26 **Atmel AVR Design Contest 2006 Winners Announcement**
- 30 **Multi-Input Temperature Logger**  
*Nial Stewart*
- 40 **Arby**  
An Arbitrary Waveform Generator with a Twist  
*Dhananjay Gadre, Pushkar Sareen, Subodh Prabhu, & Suhas Chakravarty*
- 46 **Voltage Solutions**  
Harness the Power of Voltage Converters  
*Stuart Ball*
- 67 **The Power of Flash**  
Flash Memory Techniques for Your Toolbox  
*Mark Bereit*



Data Logger (p. 12)



RS-232 Analyzer (p. 20)



Temperature Monitoring (p. 30)

## COLUMNS

- 50 **APPLIED PCs**  
**Dive Into the ZigBee Pool**  
An Easy Way to Start Moving Messages  
*Fred Eady*
- 56 **FROM THE BENCH**  
**Green Energy**  
*Jeff Bachiochi*
- 60 **LESSONS FROM THE TRENCHES**  
**Hello World ... Want Cookie**  
*George Martin*
- 78 **SILICON UPDATE**  
**Hot Chips 18**  
*Tom Cantrell*



Go Wireless (p. 50)



What's Hot? (p. 78)

## DEPARTMENTS

- 4 **TASK MANAGER**  
Agency vs. Contingency  
*C.J. Abate*
- 8 **NEW PRODUCT NEWS**  
edited by *John Gorsky*
- 93 **CROSSWORD**
- 94 **INDEX OF ADVERTISERS**  
February Preview
- 96 **PRIORITY INTERRUPT**  
For Want of a Paper Trail  
*Steve Ciarcia*





MYKLE-06

## AVR picoPower Microcontrollers

To meet the tough requirements to modern microcontrollers Atmel® has now combined ten years of low power research and development into picoPower™ technology for AVR® microcontrollers. picoPower enables AVR to achieve the industry's lowest power consumption with 650 nA with a real time counter running and 100 nA in deep sleep.

- What can AVR picoPower do for your design?
- True 1.8V supply voltage enabling operation of all features and core down to 1.8V
  - Minimized leakage current enabling 100 nA Power Down sleep consumption
  - Sleeping brown-out detector enabling full protection with no power penalty
  - Ultra low power 32 kHz crystal oscillator enabling operation at only 650 nA



For more information, check out [www.atmel.com/ad/picopower](http://www.atmel.com/ad/picopower)



## SYSTEM ON MODULE INTERNET APPLIANCE

The **SoM-NE64M** is a 16-bit system on module (SoM) based on the ColdFire MC9S12NE64 processor. This 16-bit 68HC12-compatible processor has an Ethernet MAC and PHY built-in along with two serial ports. It features 64 KB of flash memory, 32 KB of EEPROM, and 8 KB of RAM. Another 512 KB of RAM can be added as an option.

All of this functionality is incorporated on a board smaller than a business card that uses less than 1 W of power. Applications for the SoM-NE64 can be programmed using GNU tools within an Eclipse IDE. Alternatively, the SoM-NE64 can be programmed using CodeWarrior.

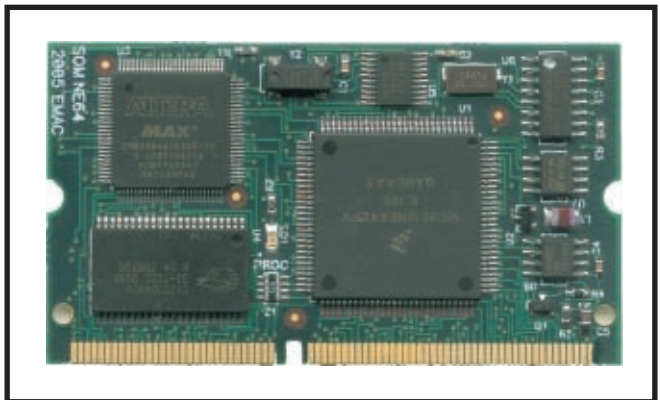
Like other modules in the product line, the SoM-NE64M is designed to plug into a carrier board containing all the connectors and any additional I/O components that may be required. This approach allows the design of a custom carrier board that meets the customer's I/O, dimensional, and connector requirements without having to worry about the processor, memory, and standard I/O functionality. Since the module itself has more functionality built in than many other SoM designs, the carrier board can be much easier to design and produce, lowering cost and time to market.

In addition to the option of the developing a carrier board, one can be purchased off-the-shelf. Off-the-shelf

carrier boards that feature A/D, D/A, MMC/SD card, keypad, LCD, and modem interfaces are currently available. The SoM approach provides the flexibility of a fully customized product at a greatly reduced cost.

The SoM-NE64M is ideal for any web/network data acquisition and control application. The SoM-NE64M costs \$67.

**EMAC, Inc.**  
[www.emacinc.com](http://www.emacinc.com)



## Connect to the wireless world...

...compact solution for wireless monitoring and control



- Ultra-low power 520MHz XScale technology
- Perfect solution for fanless solutions
- Support for GSM/GPRS, IDEN, CDMA modems
- On board GPS receiver
- Support for Linux and Windows CE
- Display controller for TFT and STN displays
- Serial ports for legacy communications
- USB, CompactFlash and PC/104 expansion
- Wide input DC input

ZEUS... a scalable, ultra-low power solution for asset tracking and information display using cellular wireless and GPS technology.

Customer support    Product longevity    Extended warranty    RoHS compliance

**888-941-2224**  
[www.arcom.com](http://www.arcom.com)



**Arcom**  
Think Embedded. Think Arcom.

# NEW PRODUCT NEWS

## NEW IM3000.X ARCHITECTURE-BASED MICROPROCESSORS

The IM3100, IM3300, and IM3900 are new processors based on the IM3000.x architecture. Each version has application-specific soft peripherals optimized for wired and wireless network applications.

This technology has three distinct advantages. Its reconfigurable architecture allows new peripheral functions to be synthesized and downloaded as microcode. Its field-programmable microcoded architecture allows the creation of custom instruction sets well suited for mobile and data communications applications. This results in very efficient, low-power operation. One version has been configured to execute Java byte code and does so with unprecedented efficiency. TCP/IP acceleration and encryption acceleration are included in the existing configuration libraries.

The IM3100 addresses the 802.11 wireless VoIP handset and terminal market with an integrated, narrowband, base band controller, Skype voice CODEC, LCD controller, and media and touch screen user interfaces.

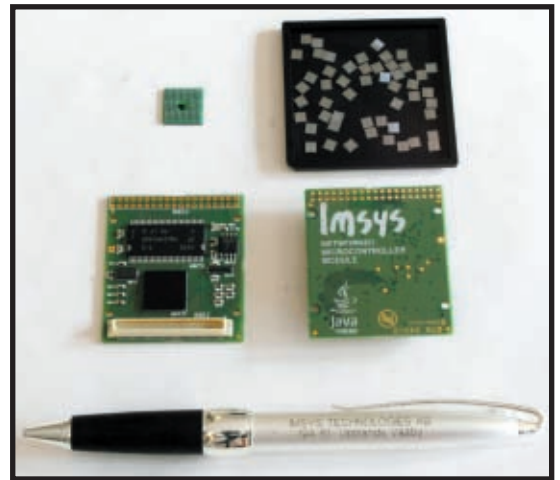
The IM3300 enables the design of an optimized GPS positioning unit for fleet management applications. The GPS base band processor is implemented along with a number of serial and parallel I/O ports for communication to local sensors and the communications unit. The unit also incorporates an Ethernet controller.

The IM3900 is a general-purpose version that brings connectivity and Java together for embedded applications. This configuration constitutes a very efficient platform for program execution in assembly code, C, and especially Java. Ethernet and a multitude of other serial channels are available to provide a platform for communications protocol conver-

sions. The IM3900 implements a full Sun Microsystems certified Java ME-CLDC configuration.

Prices range from \$12 to \$15 in 10,000-piece quantities. The pricing includes licenses for software included in the basic chip configurations.

**Imsys Technologies AB**  
[www.imsystech.com](http://www.imsystech.com)



## Tianma LCDs

**Tianma Microelectronics** specializes in the manufacture of all LCD technologies from TN to TFT. With more than 22 years of LCD experience, we have grown to be China's largest LCD manufacturer. Tianma has teams of dedicated and experienced engineers and sales people worldwide to help you with all of your LCD needs. Our continued success in cell phone, automotive, consumer, and medical industries assure our ability to help you in whatever market you serve.



21138 Commerce Pointe Drive, Walnut CA 91789 USA  
 Tel: (909)468-2839 or 1-800-864-7789 • Fax: (909)468-0868  
 Email: sales@tianma.com • Web: www.tianma.com

## Reliable ZigBee™ Mesh Networks

- Up to 8 times the range of typical ZigBee
- Easy-to-use

\$50 off XBee Professional Developers Kit when you mention this ad\*

	Frequency	Indoor Range	Outdoor Range	Power Output	Mesh Network	Agency Approvals
XBee™	2.4 GHz	30 m	100 m	1 mW	Yes	Global
XBee-PRO™	2.4 GHz	100 m	1.6 km	60 mW	Yes	Global

**MaxStream** A Dig International Company  
[www.maxstream.net](http://www.maxstream.net) 866-765-9885  
\*Offer expires 11/30/2006

# NEW PRODUCT NEWS

Visit [www.circuitcellar.com/npn](http://www.circuitcellar.com/npn) for more New Product News.

## SYSTEM ON MODULE WITH ULTRA-FAST LINUX BOOTUP

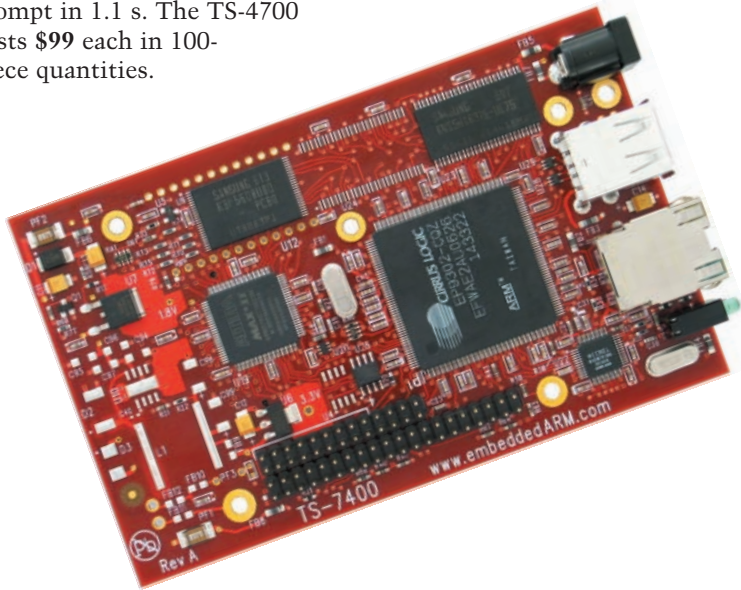
The TS-7400 is a small embedded computer module (system on module) that is designed to provide extreme performance for applications which demand high reliability, fast bootup/startup, and connectivity at a low cost and low power, such as point-of-sales (PoS), vending machines, data acquisition units, and data recorder modules.

The module is based on the Cirrus EP9302 ARM9 CPU, which provides a standard set of on-board peripherals. The EP9302 features an advanced ARM920T 200-MHz processor design with MMU. The TS-7400 includes a standard SD Card socket and a 40-pin header that brings out many interfaces, including audio, GPIO, and ADC. In addition, an one-piece setup with 802.11g Wi-Fi integrated inside a rugged enclosure is available, making this solution ideal for use in wireless sensor network, data acquisition/recorder applications, PoS, vehicle telemetry trans-

mitter units, and more.

The TS-7400 has a tweaked boot-up firmware and Kernel that, along with the hardware accelerated NAND controller and hardware ECC, enables ultra-fast bootup to a Linux shell prompt in 1.1 s. The TS-4700 costs \$99 each in 100-piece quantities.

Technologic Systems, Inc.  
[www.embeddedarm.com](http://www.embeddedarm.com)



## HIGH TECH PROTOTYPE PCBs

*Touch the Future*



- MICROELECTRONICS—LINE WIDTHS DOWN TO 1.25 MIL. (0.00125")
- HDI—STAGGERED, STACKED, AND FILLED MICRO VIAS
- HI-RELIABILITY LEAD-FREE BOARDS
- HEAVY COPPER BOARDS

### SUPPORTING THESE MARKETS

- MEDICAL • MILITARY • SPACE • DOWNHOLE • HIGH RELIABILITY • SECURITY



SIERRA MICRO ELECTRONICS • SIERRA PROTO EXPRESS  
**800.763.7503**  
[WWW.PROTOEXPRESS.COM](http://WWW.PROTOEXPRESS.COM)

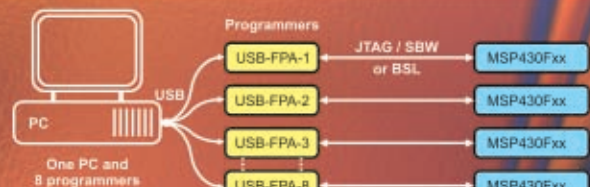
## FlashPro430 GangPro430

USB Flash Programmers  
for Texas Instruments'  
MSP430 microcontrollers.



Reliable and the fastest programmer on the market.  
Perfect for production usage.

- ✓ 60 kB Flash can be programmed in about 2 seconds
- ✓ supports JTAG, Spy-Bi-Wire and BSL interfaces
- ✓ can assign unique serial numbers
- ✓ up to eight programmers can be connected to one PC and program target devices simultaneously



**Elprotronic**  
Incorporated

[www.elprotronic.com](http://www.elprotronic.com)

2,500 NEW  
**NATIONAL SEMI**  
PARTS

5,000 NEW  
**AMP**  
PARTS

2,500 NEW  
**MAXIM**  
PARTS

# Wow! Jameco just added 65,000 new major-brand products!

7,200 NEW  
**FREESCALE**  
PARTS

2,900 NEW  
**VISHAY**  
PARTS

19,000 NEW  
**TEXAS**  
**INSTRUMENTS**  
PARTS

2,800 NEW  
**MICROCHIP**  
PARTS

## The industry's fastest growing product offering!

You know that Jameco's catalog  
always offers over 99% *in-stock*  
*availability*—the best of any elec-  
tronic components distributor...

And now, they have the  
fastest growing product offering  
in the industry!

They've just added another  
*65,000 new parts* to their online  
catalog; and it's everything  
from ICs to passives, optos to  
interconnects, power supplies  
to electromechanical.

## Service & Availability!

As Design Engineers  
know, Jameco offers great  
service, selection and  
same-day shipping!

Now you can get those  
same benefits for even  
more great brands...

## Check out these new and expanded lines:

**Aavid Thermalloy •**  
**Alcoswitch • AMP •**  
**Amphenol Connex •**  
**Atmel • Augat • AVX •**  
**Bourns • Buchanan •**  
**Comair Rotron • Condor Power**  
**Supplies • CTS • Cypress • Dallas**  
**Semiconductor • Fairchild • Freescale**  
**Semiconductor • Grayhill • Intel •**  
**Intersil • ITT • C&K Switches •**  
**Lattice Semiconductor • Lite-**  
**On • Maxim • Microchip •**  
**Micron Technology • Molex •**  
**National Semiconductor •**  
**Panasonic • Philips •**  
**Power-One • Raychem •**  
**Renesas Technology •**  
**Sandisk • ST Micro • Texas**  
**Instruments • Toshiba •**  
**Tyco Electronics • Vishay**  
**Intertechnology...**

Get it here. Right now:

**Jameco.com/CCU**

**JAMECO**<sup>®</sup>  
ELECTRONICS

Great Products.  
Awesome Prices.

# Self-Powered Solar Data Logger

*Abigail designed a microcontroller-based, self-powered solar data logger that uses a photodiode to measure solar insolation levels. The system converts the analog signal to a digital value that's stored in flash memory.*

On a seemingly rare sunny day in Ithaca, New York, the sun delivers about 1,000 W of power per square meter and just begs to be put to some purpose besides browning the backs of the students lying out in the gorges. At up to 15% conversion efficiency, COTS photovoltaics (PVs) can turn that light into electricity. A growing number of Ithacans have heeded the sun's call and installed solar electric PV systems to power their homes and businesses. Ithaca may get 40% less solar insolation than San Diego, but it gets 25% more than Germany, the world leader in installed PV capacity. Tompkins County, where Ithaca is located, has roughly 2.9 W of installed solar capacity per person, which makes it second in the U.S. to only Palo Alto, California.

With the Finger Lakes, ridgelines, and valleys cutting through the region, clouds and fog levels vary significantly from one part of town to another. But weather data is only available from a few select locations. How is a potential PV buyer to know how much power her system will produce unless they can measure the incoming light? And how can a proud PV owner know how her system is performing or detect faults unless she can confirm the conversion efficiency? Expensive commercial data logging systems that cost thousands of dollars do this well, but they are entirely unreasonable for the small system owner.

For Bruce Land's ECE476 Microcontroller Design course at Cornell University (<http://instruct1.cit.cornell.edu/courses/ee476/>), I designed an inexpensive self-powered solar data logger to meet this need (see Photo 1). I built the system around an Atmel AVR STK500 development board that fea-

tured an ATmega32 microcontroller. You can leave the logger (untouched and isolated) in the field to collect data for months or years.

## SYSTEM OVERVIEW

The solar-powered data logger features a photodiode that measures solar insolation levels and converts this analog signal to a digital value that's stored in flash memory. Every time the system logs a data point it also logs a time and date stamp so that the data can be downloaded to a PC and analyzed in the future.

When the system is logging, real-time data is displayed on its small LCD screen. The system displays several useful bits of information: the battery's voltage, the time and date, the length of time the system has been logging, and the length of time it can continue to log before running out of memory.

The logger has a dedicated solar power system that enables autonomous operation. A simple charge controller regulates the charging of a sealed battery (gel cell lead acid) by a small solar panel.

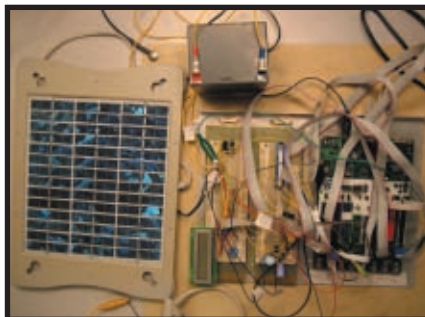
## APPLICATIONS

When planning an off-grid solar elec-

tric power system (one that isn't connected to a larger power grid), the output must be matched closely with the load in order to provide sufficient power without considerable waste. Although the price of photovoltaics has dropped by almost fivefold in my lifetime, it is still quite expensive.<sup>[1]</sup> Good planning and system design ensure that you can provide sufficient power without having to buy more PV than necessary.

The power output from photovoltaics is directly related to the insolation level. Although seasonal and annual average insolation levels for most major U.S. cities are available on the Internet, cloud cover and other weather effects can be extremely localized depending on the topography. Thus, the data for large cities isn't always the same as the data for the smaller towns in its vicinity. In addition, this data is not available for every part of the world. While accurate predictions of power output are important for grid-tied solar electric systems (in which the grid is used as 'storage' for any excess electricity produced and drawn upon for any electricity shortfall), this mainly impacts the return on investment expectations rather than the system sizing.

PV power systems are the only devices currently available for generating electricity without any moving parts. This makes them brilliantly simple and easy to care for, which is a real benefit for homeowners who do not want to spend their weekends greasing bearings and performing regular system checks. But without any maintenance needs or means for visual inspection, it is easy for system faults to go undetected for quite some time.



**Photo 1**—Check out the complete solar data logger system with the PV panel, battery, and a mess of wires.

Large commercial PV installations typically have sophisticated sensors and monitoring software that can detect system faults and activate an alarm when maintenance is needed. These features are usually too costly for residential sized PV systems. The central component of these monitoring systems is an insolation sensor whose output is compared with system power production. When the power production strays significantly from what would be expected given the insolation, an alarm is triggered and maintenance checks can be performed. Although you may not need a fully automated, integrated monitoring system, some means for determining your PV system's efficiency enables you to perform maintenance only when it's necessary and to have peace of mind at other times that everything is functioning properly.

## HARDWARE

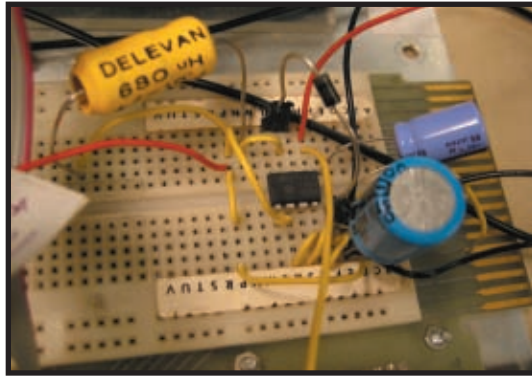
The system schematic is shown in Figure 1. The pins of the STK500 board are depicted along the bottom. Note that the RXD and TXD pins for the RS-232 connection and the data flash are also located on the board.

Port A connects to the photodiode and the switches. Port B also connects with the switches as well as the data flash. Port C controls the LCD and Port D sends data to the RS-232 connection and the field effect transistor. These ports connect to the ATmega32 microcontroller.

## POWER SUPPLY

The logger has a dedicated solar-charging system to enable for autonomous operation. It was built around a 3.2-W solar panel a Volkswagen dealer gave me to accessorize the originally diesel-fueled Volkswagen Golf TDI that I bought and converted to run on vegetable oil. But that car is another story.

Even without much optimization for efficiency, the power needs of the logger are very small. The maximum power drawn by each of the main components is 5 mW for the LCD, 75 mW for the ATmega32, and 75 mW for the photo-



**Photo 2**—The LM2574 step-down switching regulator maintains the voltage supply to the logger at 5 V.

diode for a total of 155 mW. If all components run at maximum power at all times, this would amount to 3.72 Wh/day. This does not account for losses in the voltage regulator or other minor components, but there is plenty of energy available for the system so this is not a problem.

The battery I chose was a 12-V, 5-Ah sealed gel cell lead acid deep cycle battery. This type of battery is the most cost effective when size and weight are not a concern, but safety, ease of handling, and the ability to deep-cycle the battery are. Although a 12-V, 5-Ah battery could provide 60 Wh, draining any battery too low (even a deep cycle battery) can cause damage. Five days of energy storage in Ithaca is considered conservative, but due to the reliability needs of an autonomous system such as this, it is worth having a good factor of safety. Note that 37 Wh of useful storage enables the system to ride through 10 days with no sun, giving plenty of leeway with the battery chosen.

Even with enough storage, it is necessary to be sure that the energy balance of the system is kept positive or the battery will eventually drain. Ithaca averages 2.3 sun hours per day in the winter.<sup>[2]</sup> What this means is that a solar cell rated at 1 W would produce 1 Wh of electricity per sun hour. My 3.2-W solar panel would therefore be able to produce 7.4 Wh (on average) of electricity per day during the darkest time of year in Ithaca if kept at its maximum power point. Because there is no maximum power point tracking in this system, it can be expected that the panel will produce about 5.2 Wh per day in the winter, which still

far exceeds the maximum load expected.

The PV panel, which is rated at 18.8 V at its maximum power point, was designed for trickle charging a 12-V car battery and so could be directly connected to the data logger. However, to prevent battery damage from overcharging, I needed a way to disconnect the panel once the battery was fully charged. Using a BUZ71 field effect transistor and a polling routine, the panel was effectively disconnected when the battery voltage rose above 12 V.

Depending upon the state of charge, the battery voltage will float around 12 V but will not remain steady. The logger components required a constant 5-V power supply. The ON Semiconductor LM2574, a 0.5-A, 5-V step down switching regulator (buck converter) regulates the voltage from the battery to the level needed for the system components (see Photo 2). This regulator has a typical efficiency of 72%, which is much higher than resistance-based voltage regulation.

An unresolved and bizarre result of running the logger off of the solar power supply as compared to the standard AC/DC power supply was that the on-off switch on the STK500 ceased to function. The only way to turn the logger on or off when connected to the solar power supply was to actually disconnect a battery lead.

## PHOTOSENSOR

A Texas Instruments OPT101—a monolithic photodiode and single-supply transimpedance amplifier—is used to sense the incoming solar insolation level. Natural sunlight ranges in intensity from 0 to approximately 1,000 W per square meter, but the OPT101 puts out its maximum voltage at roughly 10 W per square meter of incident insolation. With hardly any light striking the sensor, it reached its upper limit. It was thus necessary to attenuate the intensity of the light striking the photodiode to increase the range over which the sensor could differentiate intensity.

Ideally, this would have been done with neutral density filters, but I didn't

have any on hand. I used the next best scientifically accurate and available tool: electrical tape. Two layers of electrical tape covering the photodiode were found to be quite effective. This solution enabled the sensor to give reasonably reliable readings over the full range of expected intensities (see Photo 3).

The microcontroller's ADC has a maximum value of 255. Multiplying by a factor of four roughly converted the ADC reading to watts per square meter. Ideally, this system would be calibrated and the factor would be more precise than four, but this approximation gave reasonable results with a range of readings from 0 to 1,020 W per square meter.

I used the ATmega32-based STK500 development board because of its integral flash memory and switches. However, if this system were commercialized or rebuilt, it is fairly obvious that the STK500 would not necessarily be used. It has many features that are unnecessary for this project. A far simpler and more com-

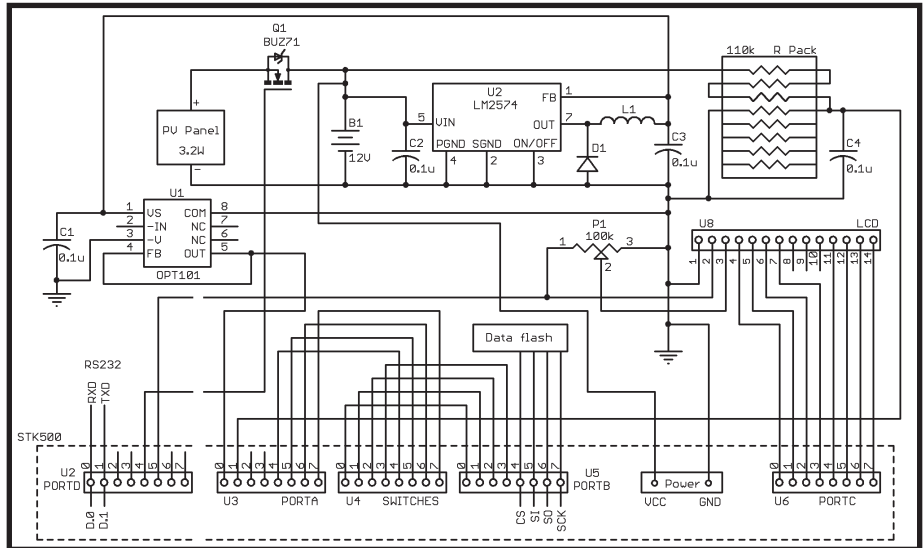


Figure 1—The system's schematic shows the STK500 and its connections with each of the other components.

pact board can be designed.

## PROGRAM

An interrupt-driven program runs the system and enables accurate timing. An interrupt service routine decrements a series of task timers once per millisecond. The main task

in the program calls various subroutines at predetermined intervals when their task timers run out and the appropriate flags are set to enable a task to run. Each of the eight buttons is polled separately once every 30 ms with a state machine to debounce the button as it stabilizes after a transi-

### Microprocessor Modules

All Arcturus Embedded Modules Include:

- Snap-in ~1.7" x 2.4" soDIMM form factor
- 32-bit Embedded Processor
- On board SDRAM, NOR or NAND Flash
- Ethernet and Serial (RS232 and/or TTL)
- Embedded uLinux OS
- Bootloader and Utilities
- Complete GNU Tools (with kit)
- Full TCP/IP stack, Filesystem
- Commercial or Extended Temp Versions

---

**ARM7+DSP C5471**  
ETHERNET + DSP

- Signal Processing, Audio, SIP Telephony Stack Available

**ColdFire MCF5282**  
ETHERNET + CAN BUS

- Equipment Networking, Security and Building Control

**ColdFire MFC5272**  
DUAL ETHERNET + USB

- Router, Internet Appliance, Gateway, Network Bridge

**DragonBall 68VZ328**  
ETHERNET + LCD

- Industrial Control, Machine Interface, Home Automation

COMPLETE  
**KITS**  
AVAILABLE

BUILD PRODUCT  
**FAST**  
GET TO MARKET

**www.ArcturusNetworks.com**

TEL +1.866.733.8647 FAX +1.416.621.0190

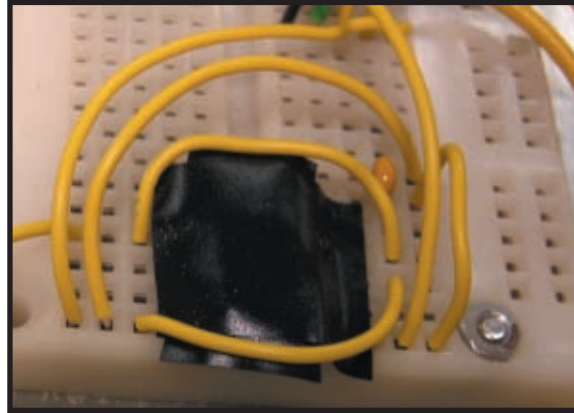
SALES@ARCTURUSNETWORKS.COM



tion. The only button that is not debounced is the LCD Wake button because bouncing is not a concern with this function. The debouncing routine is based on code written by Bruce Land.

Each time the system is turned on or reset, it welcomes the user and guides her through the set-up process (see Photo 4). During set-up, you set the system time and date, select the logging frequency, decide whether to clear any stored data or to continue by appending future data, and indicate when to actually begin logging after the system is in place.

In order to enable data appending after a reset, the memory pointers have to be stored in nonvolatile memory and initialized only when the chip is programmed, not each time the system is reset. The ATmega32 contains 1,024 bytes of data EEPROM memory organized as a separate data space in which single bytes can be read and written. The pointers to the flash



**Photo 3**—The photosensor is covered with two layers of electrical tape to provide a larger range of sensitivity. The inset shows the sensor without the tape cover.

memory as well as a log of how long the system has been logging are kept in EEPROM. During operation, you can reset the time if needed or change the logging frequency without interrupting the data collection. If the logging frequency were changed, it would be impossible to calculate how long the system had been logging unless a running tally had been kept.

A series of flags and state machines

are used throughout the program to prevent erroneous user input from activating a section of code out of order. At any given point in the program, only the relevant buttons are active and their functions change as shown in the button labels in Photo 5.

The battery voltage and the photodiode output are fed into two channels of the ATmega32's ADC. The ADC has a maximum input of 5 V. So, in order to read the battery voltage, it was necessary to use a voltage divider to guarantee that the input to the ADC was within range. Initially, the system sampled only the light level with the frequency at which the user wanted data stored. However, with logging frequencies of 1 min. to 1 h, this did not allow for a satisfying real-time display that showed changes in light intensity. Moreover, it allowed less accuracy if only one sample was taken per stored data point. Instead, the final design

# PCB-POOL®

SERVICING YOUR COMPLETE PROTOTYPE NEEDS

**Price Example:** 16 Sq-Inches (double sided pth)

**2 Days: \$ 90.00**

**8 Days: \$ 22.50**

Standard PCB-Pool Service  
SIMPLY SEND YOUR FILES AND ORDER ONLINE!

**WATCH "ur" PCB®**

Save vital time on design errors in advance of receiving your Prototype. View high resolution photographic images of your PCB during each production stage. Be one step ahead, use our realtime PCB monitoring service.

## WWW.PCB-POOL.COM

**Tollfree USA : 1877 390 8541**

sales@beta-layout.com

**DOWNLOAD OUR FREE PCB SOFTWARE**  
[www.free-pcb-software.com](http://www.free-pcb-software.com)

Industry Quality  
**LEAD FREE**  
Pb, Bi, Sn, Ag, Cu, Ni

ROHS / WEEE conform

# TRI-M SYSTEMS

Proudly distributes:



IDE Flash Drive Carrier Board with Micro SD Interface



SD-IDE-40/44

Available in 40 and 44 pin header configuration  
Support PIO 0-4 and Ultra DMA 3 mode.  
Bootable from Transflash/micro SD.  
Low power consumption.



Advanced Fanless Embedded Controller



AEC-6900

Fanless Design  
Intel® ULV Celeron® 650MHz Processor  
1 PCI (optional 2nd PCI) / 2 PCMCIA Slots  
Wide Range DC or AC Power Input



60 Watt High Efficiency PC/104 PSU



HE104-DX

6V to 40V DC input range  
+5V, +12V, -5V and -12V DC output  
High efficiency up to 95%  
PC/104 compliant

www.tri-m.com info@tri-m.com

1.800.665.5600

HEAD OFFICE: VANCOUVER

tel: 604.945.9565 fax: 604.945.9566



Photo 4—The data logger greets users before leading them through the system setup.

reads the ADC once per second through a task that enables an ADC interrupt routine and starts a conversion. The ADC interrupt automatically switches between the two channels for the photodiode and the battery voltage, reading the battery voltage once for every five times the photodiode is read.

To get an accurate battery voltage reading, the PV panel is disconnected just before the reading is taken and then reconnected immediately after if the battery voltage is below 13 V. If it is at 13 or above, the PV panel is not reconnected to prevent overcharging.

The instantaneous readings are displayed to the LCD screen and kept in a running average over the logging interval. When the logging interval is complete, a data point is stored in the flash memory and the running average is cleared.

To be able to run for a useful length of time, the data logger needed an external memory for storing measured data. The older STK500 boards have flash memory chips built into them (as can be seen on the upper right of the board in Photo 5), making this an obvious choice. Unfortunately, I did not have any instruction set for interfacing with the flash memory. An earlier ECE476 project used this flash memory. Based on the project's code and comments, I was able to learn how the flash memory interface worked.<sup>[3]</sup> The designers used a program named dFlash, which was written by Terje Frostad of Atmel Norway. I e-mailed the Atmel AVR technical support team and received permission to use dFlash in my project as well. The dFlash program provides a set of routines to interface with the flash memory by writing to a 264-byte buffer. This buffer is then written to

one of 1,024 264-byte pages of the flash memory.

Because the routines take byte-size data as input, I had to break the insulation value (stored as an integer) into 2 bytes and then put it back together again when storing or retrieving it. The 2-byte insulation together with the time and date stamp meant that each time the system logged a data point it used 8 bytes of memory. This allowed the system to log 33,792 data points. The logging interval is user-selectable from 1 min. to 1 h. At an interval of 1 min., the system is able to continuously log data for 23.4 days before filling the memory. At an interval of 1 h, the system is able to log for 3.8 years.

One thing I did not realize that caused me a considerable headache was that PortB.4:7 is used for interfacing with the flash. I had initially been using PortB as input for the switches, but I found that three of the switches ceased functioning properly. When I realized that dFlash was reinitializing these pins, I was able to switch these buttons to PortA.

The flash memory can be read either through a buffer or directly, the latter being my choice for the logger. If you were to call for the data to be retrieved before the buffer had filled and written the data to flash, there would be no data to retrieve. Thus, it was necessary to write the buffer to flash memory just before the data retrieve routine was entered as well as when the buffer became full. It was also necessary to give the microcontroller a brief period to finish writing the buffer before the read command was executed or the same problem would occur.

The read command is executed when you press the Retrieve Data button (active only when the logger is stopped). The logger must be connected to a computer using an RS-232 cable with straight-through connection. You should start a simple terminal program on the PC (e.g., HyperTerminal) set to 9,600 bps, no parity, 1 stop bit, and no flow control.

When the Retrieve Data button is pressed, the system prints identifying header rows followed by a row for each data point logged with the time and date at which it was stored. The LCD displays the message "Uploading

# How far will your design take you?





Challenge yourself against other top embedded engineers around the world in **DesignStellaris2006**, proudly sponsored by Luminary Micro, Keil, and Circuit Cellar.


Use any microcontroller in Luminary Micro's Stellaris™ family of ARM® Cortex™-M3 controllers with the ARM RealView® Microcontroller Development Kit (MDK-ARM) to create your design contest entry, and see how far your design will take you!

- No purchase necessary to enter.
- \$10,000 in cash prizes!
- Entry deadline is February 7, 2007.
- Winners will be announced at the Embedded Systems show... Silicon Valley 2007.
- Submit your design today!


FOR COMPLETE DETAILS, VISIT [www.LuminaryMicro.com/DesignStellaris2006](http://www.LuminaryMicro.com/DesignStellaris2006)



LUMINARY MICRO™



KEIL™  
An ARM® Company



Cortex™  
Intelligent Processors by ARM

CIRCUIT CELLAR®

[www.LuminaryMicro.com/DesignStellaris2006](http://www.LuminaryMicro.com/DesignStellaris2006)



THE LM3S811 EVALUATION KIT includes the Stellaris LM3S811 Evaluation Board, an evaluation copy of MDK-ARM, USB cable, documentation, and programming examples.

Data" followed by the time and insolation value presently being uploaded. Upon reaching the end of the data, a footer row is printed to the terminal program and the LCD displays the message "Done Uploading Please Restart." The data in the terminal program on the PC may then be copied into another program for storage or analysis. If an incomplete transmission was made, you may reset the logger and choose to upload the data at the first prompt. The data may be uploaded as many times as the user desires until the memory is cleared.

Initially, the program was written with each of the subroutines that updated a parameter sending its update to the LCD. This avoided updating the LCD more frequently than necessary. However, when power-saving code was incorporated to turn the LCD screen off when the system was idle by driving it with a port pin, the system would freeze every time it went idle. It turns out that the LCD sends and receives messages to and from the microcontroller. If the LCD is off when the microcontroller sends it a message,

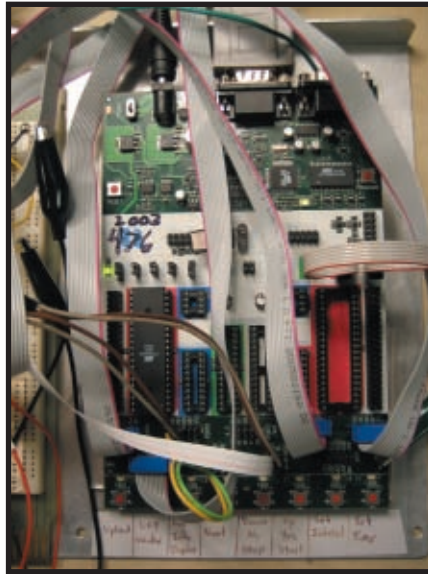


Photo 5—Check out the STK500 board with labels for each of the eight buttons.

it is not able to respond and the microcontroller hangs, waiting. The program thus had to be rewritten to prevent the LCD from being called when the LCD screen was turned off. I did this by writing a single routine that printed to the LCD screen. A state machine and a


variety of flags controlled it so that the correct message would be displayed.

Additionally, there was some difficulty at times with the LCD not making proper contact with the surplus whiteboard I was using. When the LCD lost contact momentarily, it would cause the system to freeze. Thus, I added an extra new whiteboard solely to hold the LCD screen securely because I was not able to solder the borrowed LCD to my project. Another problem I had with the LCD was when the STK ground was not properly connected to the system ground, the negative terminal of the battery. This happened during testing with the STK500 plugged into an AC/DC power supply. When the two grounds were not at equal voltages, the voltage across the LCD was not in its operating range. The positive voltage was coming from the STK port pin and the negative voltage from the battery negative terminal. Once the two systems were joined by connecting to the STK ground pin (or also powering the STK from the battery), the LCD resumed

## Enhance Your Connectivity


The RCM4000 — Ethernet And Much More

- Ethernet with royalty-free TCP/IP stack
- Up to 25 configurable GPIO
- On-board 12-bit A/D
- Up to 5 serial ports
- 512K flash (program), 32 MB NAND flash (data)
- Rabbit® 4000 at 58.98 MHz




**New!**  
Next Generation  
**Rabbit 4000**

**Complete Development Kit**  
**\$149** Regular \$239 Limited Time Offer



**Order Online At**  
[rabbit4000.com](http://rabbit4000.com)

06227



# KEIL™

An ARM® Company

## Microcontroller Tools

**µVision® IDE**

```

graph TD
    subgraph IDE
        A[µVision Project Manager] --> B[C/C++ Compiler]
        A --> C[Macro Assembler]
        B --> D[C/C++ Libraries]
        C --> D
        D --> E[Linker / Locator]
        E --> F[µVision Debugger]
        F --> G[Device Simulation]
        F --> H[Target Hardware]
    end
        
```

**RTOS**

- RTOS Kernel
- TCP/IP Suite  
TCP, UDP, PPP, SLIP  
ARP, DNS,  
Ethernet, DHCP,  
HTTP, FTP, SMTP
- Flash File System
- USB Device Interface
- CAN Interface

**Professional Tools for Over 1,000 Devices**

- 8-bit: 8051 and Extended 8051 Variants
- 16-bit: C16x, XC16x, and ST10
- 32-bit: ARM7, ARM9, and Cortex-M3

**New! RealView® ARM Compiler**

800-348-8051
[www.keil.com/xd](http://www.keil.com/xd)

normal operation.

## IMPROVEMENTS

If I rebuild this system for fun or commercialization, it will need to be packaged in a weatherproof enclosure to allow for outdoor operation. Additionally, since not everyone has a spare PV panel lying around the house, it would be important to increase the system's efficiency to allow for a reduced panel size because this is the most expensive component. This can be done in many ways.

The first way would be to have a maximum power point tracking charge controller rather than the simple on-off switch used in this project. This would keep the voltage of the solar cells at the maximum power point on the I-V curve. It would effectively extract about 30% more power from the panel than allowing the battery to determine the panel's operating voltage. The second way to improve efficiency would be to reduce the power consumption of the system. Because I had plenty of power from my PV panel and sufficient battery capacity, there was no need for this system to do more than turn the LCD screen off when idle. However, it would be possible to reduce power needs further by reducing the chip's clock speed, letting the chip go idle between readings or at night, and eliminating the LEDs on the STK500 board or using another board entirely.

The sun was shining in my eyes as I started working on this article last summer. I think I'll soon have to join the fold and get a PV system of my own to go with the logger. ☑

*Abigail Krich (ajk28@cornell.edu) holds a B.S. in biological and environmental engineering and a Master's degree in electrical engineering from Cornell University. She is now a project developer at Tamarack Energy, a renewable energy development company. She has previously worked at the National Renewable Energy Laboratory's National Wind Technology Center and at Northern Power Systems.*

## PROJECT FILES

To download the code, go to [ftp://ftp.circuitcellar.com/pub/Circuit\\_Cellar/2007/198](ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/198).

## REFERENCES

- [1] R. Wiser, M. Bolinger, P. Cappers, and R. Margolis, "Letting the Sun Shine on Solar Costs: An Empirical Investigation of Photovoltaic Cost Trends in California," LBNL-59282, NREL/TP-620-39300, 2006, <http://eetd.lbl.gov/ea/ems/reports/59282.pdf>.
- [2] Solar Insolation for Major U.S. Cities, Advanced Energy Group, [www.solar4power.com/solar-power-insolation-window.html](http://www.solar4power.com/solar-power-insolation-window.html).

- [3] S. Jean-Louis and C. Pandarinath, "The Big Red Guide," ECE476 Final Design Project, Cornell University, 2004, <http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2004/sj74/main.htm>.

## RESOURCE

Datasheets, dFlash.c, and the User Guide, <http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2006/ajk28/ajk28/index.html>.

**\$51 For 3 PCBs**  
**FREE Layout Software!**  
**FREE Schematic Software!**

- 01 DOWNLOAD our free CAD software
- 02 DESIGN your two or four layer PC board
- 03 SEND us your design with just a click
- 04 RECEIVE top quality boards in just days

**expresspcb.com**

# QuickComs

## An MC16C/62P-Based RS-232 Analyzer

*Although RS-232 is disappearing from new devices, you'll probably have to work with it from time to time. You can connect Nick's M16C/62P-based system to an RS-232 device to determine the correct data rate, wiring, and encoding scheme. You can also use it to monitor a serial link.*

The inspiration for this project came from various sources and situations, including my own debugging of old lab equipment, the need to quickly diagnose a broken serial link in a microcontroller system during the design phase, the need to quickly debug student designs in a teaching lab environment, and the need to assist those without an oscilloscope to determine the communication parameters of old equipment.

A few years ago, a friend of mine who was developing front-end systems for databases was given a project that needed to communicate with some electronic scales. He was given scales and cables, but that was it: no data rates, parity bits, or stop bit information. I was asked if there was an easy way to determine the device's communication settings. I asked if he had a scope. "A what?" was the reply. Eventually, he figured out by trial and error that the device used 1200-7e1. I thought there should be a better way.

For me, figuring out the exact data rate and encoding and wiring scheme for the serial connection on that really old piece of equipment at the back of the lab has always been part of the fun and challenge of being an electronics technician. But then again, not everyone can just run off and grab an oscilloscope or logic analyzer and come up with an answer. Sometimes having to repeat the process 10 times in a single lab session as students learn the ins and outs of working with microcontrollers can be a bit tedious and take

some of the fun out of it. These are some of the reasons why I designed the QuickComs prototype.

With RS-232 quickly being replaced by USB, I hope the nightmare of incorrectly wired appliances and forgotten data rates will be a thing of the past. Unfortunately, RS-232 will still be with us for a long time to come, and as it becomes used less frequently, the exact parity and data rates for devices will be easily forgotten.

### QuickComs PROJECT

QuickComs is a useful M16C/62P microcontroller-based system that can be connected to an RS-232 device to determine the correct wiring, data rate, and encoding scheme. It can be used to monitor a serial link as well as act as an automatic null modem cable. QuickComs can be inserted inline to a serial link or connected to a single device.

This project should be useful to anyone who often works with older scientific or industrial equipment. You'll also find it useful if you need to quick-

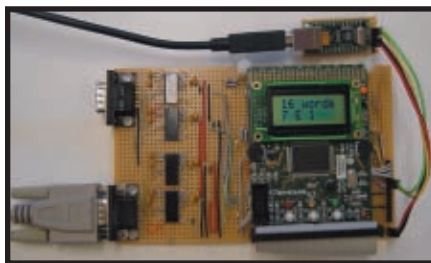
ly troubleshoot a serial connection in a microcontroller system. It has already proved useful for quickly revealing situations in which the system or UART clock was running at an unexpected speed. This is fantastic in a teaching lab when a student's breadboard circuit doesn't work the first time. If you don't have an oscilloscope, this system gives you the power to determine the communication parameters of equipment. You don't need to have an in-depth understanding of the technology. It's a turnkey solution for the front-end developer.

As you can see in Photo 1, the current design is only a prototype. As such, it isn't as portable as a commercial product (nor is it as pleasing to look at). But I imagine it would be quite straightforward to take this design and optimize it for commercial production in a package not much larger than a standard deck of cards (similar to the Atlas range of products from Peak Electronics Design).

### SYSTEM OPERATION

The device has a number of different displays and modes. Each summarizes different information about the serial link. The system operates completely via two of the three buttons: S1, S2, and S3. These are located from left to right across the bottom of the Renesas Technology SKP16C62P demonstration board.

The S2 and S3 buttons are used to navigate the display system. S3 acts as a Next button that moves the system through the modes of analysis. S2 acts



**Photo 1**—I built the prototype using the SKP16C62P demonstration board, some strip board, and a handful of ICs that I had lying around. The process was fairly quick and easy.

as an Enter button for selecting options from the menu.

In the initial display, the device shows the status of the two serial connections DE9F and DE9M, the female and male connectors, respectively. If a cross-wired cable is detected on a port, then "Xvr" will be displayed next to the name of the port. If the port is disconnected or isn't detected, then it will not be shown on the display. If neither port is in use, then a "No Cable Detected" message will be displayed. When at least one port is detected, you can move on to other displays. If both serial ports are being used, then data rate detection and analysis will occur only on the DE9F connector. To detect on the DE9M port, simply remove the DE9F cable.

The voltage display shows the RS-232 voltage levels. These voltages are

measured on the RECEIVE pin of each input. S2 changes the display format to include two decimal places. It's this voltage that is used to detect the presence and polarity of a cable. This information is a useful check for ensuring that the serial link's hardware layer is present and operational.

Monitor mode is used to initiate communication on the USB connection to allow third-party monitoring of the data sent and received on the DE9 connectors. The USB connection uses the same data rate and settings as those detected on the DE9 connectors. S2 is used to toggle the USB connection on or off.

The number of events detected on the RS-232 lines is available as a Display mode. The device has a maximum number of 128 events that can be analyzed. The event memory resets automatically if there is a 1-s pause in data on the

serial line. Pushing S2 can also clear the event memory. When an event is detected, the green LED illuminates.

The period display is used to show the width of the shortest pulse in the event memory. This is used to calculate the data rate. Again, S2 changes the display format to include decimal places. The display will automatically change units between microseconds and milliseconds where appropriate.

In addition, the exact data rate calculated from the bit width is available. As with other displays, S2 changes the display format to include a decimal place. The data rate is displayed in bits per second. This information is useful when a microcontroller has been set up incorrectly or when a clock has gone haywire and is no longer using a standard data rate.

Although the given bits per second

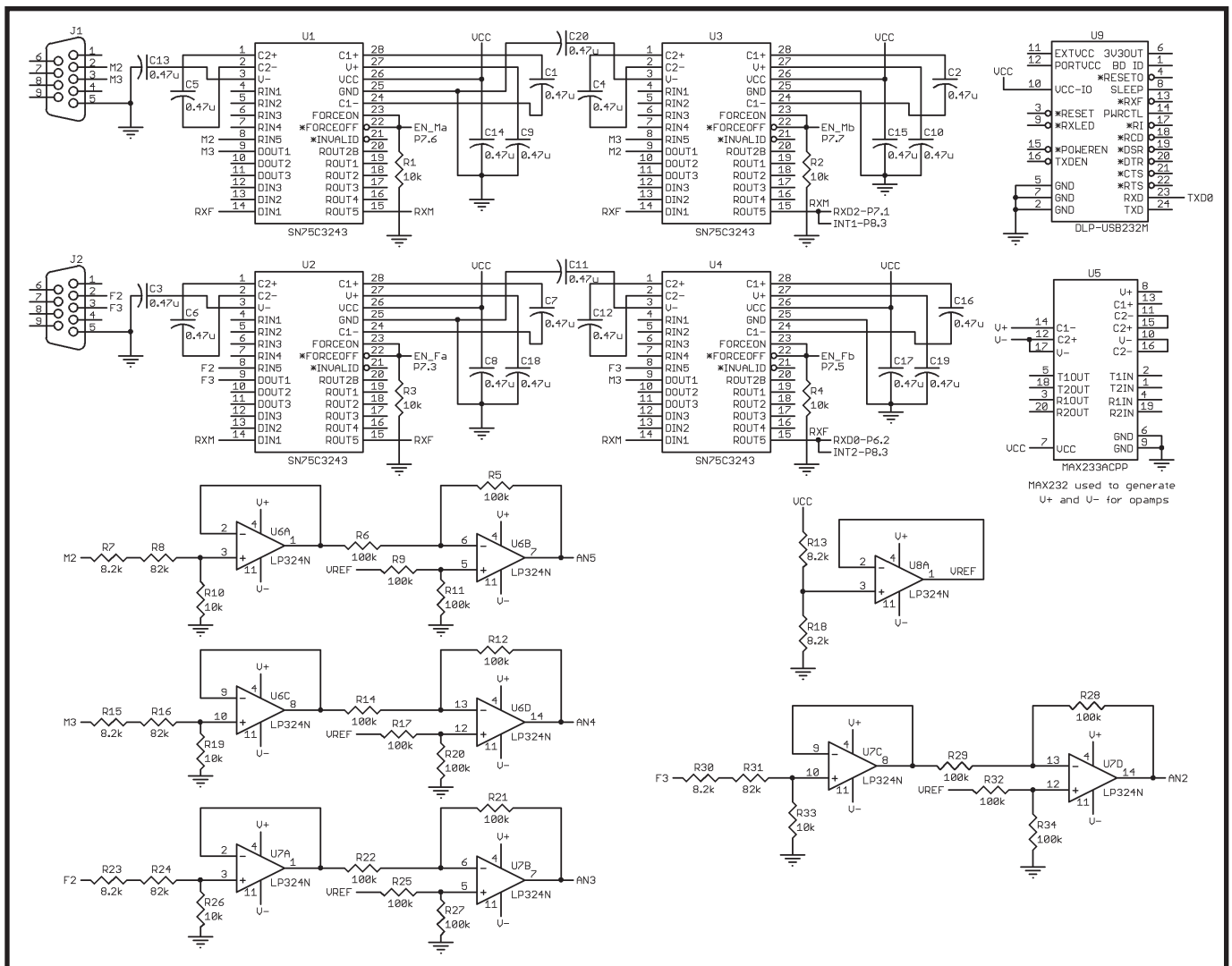


Figure 1—Although there are a number of ICs in this design, it's simply some very basic circuits repeated a number of times.

value is useful, it is often slightly wrong due to timing inaccuracies and it doesn't always give a usable value. The data rate Guess mode calculates the closest commonly used data rate from an internal table of standard data rates. The top line of the display shows how close the measured data is to the guess. A value less than 90% usually means the guess is wrong. This information can be a very valuable ballpark figure when you are beginning to debug a serial link.

If the data rate guess looks good

enough, you can then go on with some analysis of the data in the event memory using the guessed data rate. If no reasonable data can be found, then "NoMATCH" is displayed. If the parity bits can't be determined, the display will read "xx bit," where xx is the number of bits per word including start, stop, and parity bits. The top line displays the number of complete words or characters used in the analysis. If you have a good data rate guess (better than 98%) and a decent num-

ber of events (more than 32 or so), this is usually bang on the money.

## HARDWARE

Renesas Technology's M16CM30626 was a great choice for this design because of its three flexible UARTs and its large memory. I used the SKP16C62P development kit to help reduce development time. I found it useful because it contained all of the basic input and output hardware, and it brought all of the ports out to user-friendly 0.1" headers.

The full-featured development environment provided by the High-Performance Embedded Workshop in combination with Renesas's interactive on-line labs gave me a great head start for learning about this microcontroller family. Using these tools, I could get to grips with the environment and hardware virtually while the real hardware was in the hands of FedEx. This enabled a great start to the project, and when the actual hardware arrived, I was ready to start my specific application development.

As you can see in Figure 1, the circuit is based on a number of RS-232 line drivers and receivers (Texas Instruments 75C3243). Two of these are used on each input and have crossed wiring. When the system starts up, all of the transceivers are disabled. The firmware then looks at the voltages on the signal pins of the connectors and enables the appropriate transceivers.

## TRANSCIVERS & LINE DRIVERS

Two transceivers were used for each RS-232 connection. I used Texas Instruments 75C3243 ICs because they were available at the time. For a future design, I'm considering using a dual or even quad transceiver chip such as the 64C232343 to lower component count.

This system was not designed to be the most inexpensive solution, but I wanted it to be easy to construct. The drivers were chosen simply because I had them on my workbench. If you want to build a similar system, any driver with a high impedance or Shutdown mode would be suitable. In operation, an individual select line enables each chip. At most, only one chip is ever active per channel at one time to avoid driver contention issues.

**You design it.  
We'll connect it to the world.**

**Attention Design Engineers  
Win \$6,000 or More!**  
www.lantronix.com/wirelesscontest

**Serial to Ethernet Device Server™**

**Serial to RS-232 Wireless Device Server**

**PUT THE POWER OF THE WEB IN YOUR PRODUCTS.**

Imagine providing the ability to access, control, even diagnose and repair your products from virtually anywhere... at any time over a network or the Internet.

XPort™ and WiPort™ device servers enable you to quickly build Ethernet or 802.11 b/g connectivity into your designs. With a robust operating system, built-in Web server and full TCP/IP stack, XPort and WiPort have everything you need to bring your products to market with lightning speed.

And they're secure. Both have 256-bit AES Rijndael encryption. WiPort features 128-bit WEP and WPA security.

Lantronix provides the networking expertise, so it's easier than you may think. WiPort is even FCC-certified, so you don't need separate certification. Best of all, design changes are usually minimal or unnecessary.

Call or visit our Web site to get a Development Kit and put the power of the Web in your products.

Visit [www.lantronix.com](http://www.lantronix.com) for your free Device Networking white paper.

**LANTRONIX®**  
Network anything. Network everything.™

©2006, Lantronix, Inc. Lantronix and XPort are registered trademarks, and WiPort and Device Server are trademarks of Lantronix, Inc.

[www.lantronix.com](http://www.lantronix.com) | 800.526.8764



The chips operate in three modes. In High Impedance mode, both chips are disabled. This state is used to examine the voltages present on the communications line with no excitation from the on-board drivers. With the first chip selected, the channel is in Normal mode. In this mode, the pins of the DE9 are connected in standard configuration with pin two corresponding to the RXD (input) signal and pin three as TXD (output). In this configuration, the second chip is disabled. Lastly, in Crossover mode, the second chip is selected and the first is disabled. The second transceiver is wired with opposite connections to the DE9. This will have correct connections for a crossover/null modem cable. To keep the circuitry simple, only TXD and RXD signals were used. Flow control signals such as CTS, RTS, and DTD could have been added, but they weren't used due to time constraints. This set up is repeated for each channel.

The TTL logic sides of the two transceivers are connected in parallel and then cross-wired and connected between channels. This ensures that the two channels have a correct connection. TXD from channel 1 (DE9M) is connected to RXD on channel 2 (DE9F). As a result, there are only two TTL signals on the schematic: RXF and RXM. These signals are connected to RXD1 (P6\_2) and RXD2 (P7\_1) on the demonstration board. This enables you to monitor the communications on each port individually. This crossover of the transceivers on the TTL logic side enables the system to function as an automatic null modem cable with minimal overhead on the microcontroller system. All the microcontroller has to do is simply enable the appropriate transceiver for each channel.

To enable the measuring and analysis of raw serial data, these signals are also connected to two of the external interrupt channels: INT1 and INT2. This enables the system to accurately detect level transitions.

## VOLTAGE SCALING & SAMPLING

In the initial state, all transceivers are off, and the built-in ADC measures the external voltage on the pins. A

separate channel of ADC and op-amps (giving four channels in total) sample each pin voltage.

The voltage on each SIGNAL pin is monitored with some basic op-amp circuitry. This is done to scale them from possible inputs of  $\pm 25$  V to the 0 to 5 V available to the microcontroller. The pin voltages are first scaled using resistors and then buffered using a voltage follower. The voltage is then inverted and shifted. This centers it on the reference volt-

age. The reference voltage is half the microcontroller supply voltage ( $V_{CC}$ ) and is created by a simple resistive divider and a buffer. I initially considered multiplexing these signals through a single input stage, but I decided to go for the brute force approach for two reasons. First, op-amps are generally cheaper than analog multiplexers. Secondly, with the SKP16C62P, I had inputs to burn. In fact, four ADC channels are hardly a big request for any modern microcon-

## Fighting against your PCB-Design Software?

Here's something that will spare your time and your budget!

Boards designed under EAGLE are found in patient monitoring equipment, chip cards, electric razors, hearing aids, automobiles and industrial controllers. They are as small as a thumbnail or as large as a PC motherboard. They are developed in one-man businesses or in large industrial companies. EAGLE is being used in many of the top companies. The crucial reason for selecting EAGLE is not usually the very favorable price, but rather the ease of use. On top of that comes the outstanding level of support, which at CadSoft is always free of charge, and is available without restriction to every customer. These are the real cost killers!



## EAGLE 4.1

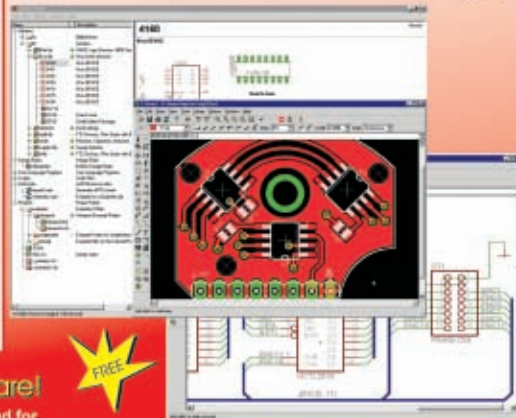
Schematic Capture • Board Layout  
Autorouter

for Windows®  
Linux®  
Mac®

Now  
available  
for Mac!

### Version 4.1 Highlights

- ▶ Powerful library management: e.g. move devices between libraries, base library for packages, generate package variants from other libraries.
- ▶ Dynamic ratsnest during routing process.
- ▶ Copy function in schematic.
- ▶ Rotate components in 0.1-degree steps.
- ▶ Blind & buried vias and pads with off-center drill.
- ▶ User-defined background color.
- ▶ Miter function for (rounded) tracks.
- ▶ Smash for groups.
- ▶ Measure distances between arbitrary points.
- ▶ Choose alternative raster on-the-fly with Alt-key.



### EAGLE 4.1 Light is Freeware!

You can use EAGLE Light for testing and for non-commercial applications without charge. The Freeware Version is restricted to boards up to half Eurocard format, with a maximum of two signal layers and one schematic sheet. All other features correspond to those of the Professional Version. Download it from our Internet Site or order our free CD.

If you decide in favor of the Commercial Light Version, you also get the reference manual and a license for commercial applications. The Standard Version is suitable for boards in Eurocard format with up to 4 signal layers (max. 99 schematic sheets). The Professional Version has no such limitations.

FREE

Prices	Light	Standard	Professional
Layout		199\$	399\$
Layout + Schematic		398\$	798\$
Layout + Autorouter		398\$	798\$
Layout + Schematic + Autorouter	49\$	597\$	1197\$

<http://www.CadSoftUSA.com>

800-858-8355

Pay the difference for Upgrades

CadSoft Computer, Inc., 801 S. Federal Highway, Delray Beach, FL 33483  
Hotline (561) 274-8355, Fax (561) 274-8218, E-Mail: [info@cadsoftusa.com](mailto:info@cadsoftusa.com)

Product is a registered trademark of CadSoft Computer, Inc. in a registered trademark of CadSoft. All other registered trademarks of CadSoft Computer, Inc.

troller these days.

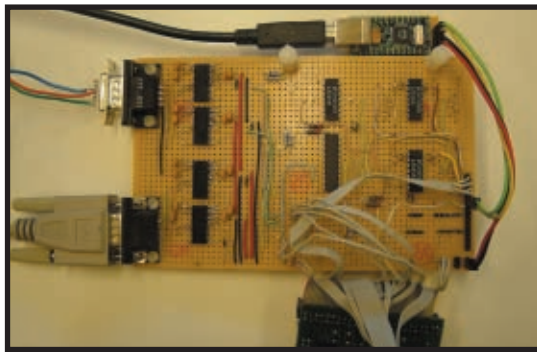
A fifth RS-232 driver was used simply to generate the  $\pm 10$  V for the op-amp voltage rails. This was a MAX233ACPP. I just so happened to have one still sitting on my workbench. It did the job nicely. I didn't need any external parts. For my future design, this will be eliminated. I will use either the existing drivers or a less expensive, more efficient, solution. Using this independent voltage generation enabled me to have all of the 75C3243 ICs in Shutdown mode until I had ascertained the correct connection polarity.

## CONSTRUCTION

I used strip board to mount the components and the SKP16C62P. It was a good choice because the simple external circuitry consists of mostly through-hole components. The transceiver chips were all SMD 1.27-mm pitch small outline packages (SOIC). To mount the chips, I simply ran uninsulated wire through holes in the strip board and then soldered to the pins on the chip sitting on the component side of the board. I was able to use this technique because a large number of the pins remained unused in the design. This has proven to be a pretty robust technique. All of the chips are held firmly in place and have managed to withstand a fairly rough life on my workbench.

I mounted the demonstration board using standard PCB stand-offs over the sampling and scaling section. Photo 2 shows the component side of the board with the Renesas board folded out of view.

I connected the circuitry to the demonstration board using an old 50-way IDC ribbon cable that I had in a box of junk. This was great because it enabled me to quickly and easily disconnect the demonstration board when I wanted to use it in other M16C projects and experiments. Any unused wires were simply tied back in a neat bundle and moved out of the way. If you have a keen eye, you'll notice that the hook-up wire is mostly single core from a salvaged length of twisted-pair telephone network. There



**Photo 2**—When you remove the Renesas Technology hardware from the prototype, you're left with a relatively basic set of electronics.

are a few 0.1- $\mu$ F decoupling capacitors not shown on the bottom of the board.

## SOFTWARE

Data rate analysis is achieved by using a 16-bit timer and information obtained from the external interrupts (INT1 and INT2). The time lapsed between each event is recorded in a buffer using the 16-bit timer. The timer is free running with any overflows recorded between external interrupt events.

The software sets the internal timers to 3 MHz (the main clock divided by eight). This gives a frequency resolution of 333.3 ns. In order to be used for other human interaction events, the timer is set to reset every 5 ms. At this point, an overflow is recorded, LEDs are updated, and a data timeout is checked for.

Although mostly redundant, the logic level after the event is also recorded. The levels are recorded as single bits, whereas the main event time array stores the timer values as 32-bit numbers. The event memory is reset if no events have occurred for 1 s. In its current form, the software has memory for up to 128 events. At a standard 8N1 encoding, this equates to at least 12 characters. This may seem like a very small data set, but in practical tests, it was found to be plenty. The event times are stored as raw clock counts.

When designing the software for the analysis, I aimed to simply mimic the actions I would manually take and automate them. The trouble with this approach was that so much of successfully tackling these problems in real life relies on educated guesswork from prior experience. I was

able to emulate this through the use of a simple data rate table because a standard data rate is used more often than not. (Students are often amazed when looking at a rough 100- $\mu$ s pulse. They tend to say, "Hmm, I bet that's 9,600 bps.")

I thought of using a similar trick for the encoding schemes, but I felt it was better to go for a brute force approach. This makes it possible to find that one-off crazy setting that you would never use yourself. My manual approach is to try different encoding schemes in the order of popularity (based on my experience). I usually start with 8N1, 7N1, 8E1, and so on until I see some data that looks about right. With the speed of the microcontroller, it is fast enough to simply try every valid possibility. Of course, the other difficulty is defining exactly how to quantify "about right" in terms of C code and arrays of data. Luckily, that is something UARTs do every day. It is well documented. I found Jan Axelson's book, *Serial Port Complete*, to be extremely helpful in this respect.

## DATA RATE

The shortest event is easily used to determine the data rate. Selecting the closest value from a table of common data rates then normalizes it.

The event data is then converted from time data to bit length data using the nearest data rate. So, a 300- $\mu$ s event is said to be 3 bits long if the selected data rate is 9,600 bps. This data is then used to look for consistent positioning of start, stop, and parity bits.

## ENCODING

When I first wrote the software for this project, I called it a "statistical analysis." I don't know if this term is really correct, but the idea is to simply try everything and see how often it's right. I then just simply pick the best solution. This type of software is, of course, only as accurate as the sample data. It's prone to being wrong for rather small sample groups. The more astounding thing was its accuracy.

The first part of the encoding analy-

sis is a simple routine that looks for start and stop bits and then counts the number of times it has successfully found a word at a given word length. This is repeated for each word length from seven (5N1) to 12 (9x1). The one with the highest score is then assumed to be the total word length.

After word length has been established, the software assumes that there is only 1 stop bit and it analyzes the data for the parity bit. If it finds a given parity 100% correct for each of the possible words in the event memory, the communication is deemed to be of that parity. If it finds all the words correctly but mixed parity, it determines there to be no parity. The software could easily be extended to look for more than 1 stop bit.

## FUTURE PLANS

As I developed the QuickComs analyzer, I was amazed at how I was able to take a small sample of data, perform some simple analysis, and have a useful device. I was also amazed at the system's tolerance of errors given the simplicity of both the software and hardware.

I recently began developing the all-in-one super RS-232 tool, but sadly, it's still in development. (It's now sitting in some boxes just beside my workbench.) With this latest reincarnation, I hope to integrate all of its existing features with some new ideas. I plan to turn it into a pocket-sized, stand-alone terminal. It will be a technician's best friend. ☺

*After graduating in 2001 with a B.E. in electrical and computer engineering from Canterbury University in New Zealand, Nick Lott worked for three and a half years as an electronics technician for Victoria University of Wellington New Zealand. He spent last year exploring Japan and building electronic systems while living in Tokyo and teaching English. Nick currently lives in London. You may contact him at [nick.lott@gmail.com](mailto:nick.lott@gmail.com).*

## PROJECT FILES

To download code and additional files, go to <ftp://ftp.circuitcellar.com/pub/>

Circuit\_Cellar/2007/198.

## RESOURCES

J. Axelson, *Serial Port Complete*, Lakeview Research, Madison, WI, 2000.

P. Horowitz and W. Hill, *The Art of Electronics*, Cambridge University Press, Cambridge, England, 1989.

Renesas Technology, M16C Family, Renesas Interactive, [www.renesasinteractive.com/renesas/guest/guest\\_index.htm](http://www.renesasinteractive.com/renesas/guest/guest_index.htm).

## SOURCES

**DLP-USB232M USB UART Adapter board**  
DLP Design, Inc.  
[www.dlpdesign.com](http://www.dlpdesign.com)

**LP324N Op-amp**  
National Semiconductor Corp.  
[www.national.com](http://www.national.com)

**M16C/62P StarterKit Plus**  
Renesas Technology Corp.  
[www.renesas.com](http://www.renesas.com)

**SN75C3243 IC**  
Texas Instruments, Inc.  
[www.ti.com](http://www.ti.com)

**Professional Features – Exceptional Price**

**34 Channels sampled at 500 MHz**  
**Sophisticated Multi-level Triggering**  
**Transitional Sampling / Timing and State**

**Connect this indispensable tool to your PC's**  
**USB 1.1 or 2.0 port and watch it pay for itself within hours!**

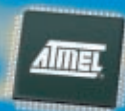
- 500 MHz Sampling / Timing Mode (Internal clock)
- 200 MHz Sampling / State Mode (External clock)
- Multi-level Triggering on Edge, Pattern, Event Count, Group Magnitude/Range, Duration etc.
- Real-Time Hardware Sample Compression
- Qualified (Gated) State Mode Sampling
- Interpreters for I<sup>2</sup>C, SPI and RS232
- Integrated 300 MHz Frequency Counter
- +6V to -6V Adjustable Logic Threshold supports virtually all logic families
- Full version of software free to download
- Mictor adapter available

[www.pcTestInstruments.com](http://www.pcTestInstruments.com)

Visit our website for screenshots, specifications and to download the easy-to-use software.

**Intronix Test Instruments, Inc.**  
Tel: (602) 493-0674 • Fax: (602) 493-2258  
[www.pcTestInstruments.com](http://www.pcTestInstruments.com)





## The Atmel AVR Design Contest 2006 Winners Announcement



The Atmel AVR Design Contest 2006 was an excellent opportunity for designers to work with the Atmel AVR family of flash memory microcontrollers and test their design skills against the world's best and brightest engineers. Last February, designers from all corners of the globe started working with the parts and planning their projects. After thoroughly reviewing each submission, the judges awarded prizes to 12 projects based on their technical merit, usefulness, originality, design optimization, and cost effectiveness.

We are proud to announce that Alberto Ricci Bitti has won the Grand Prize for his innovative WITNESSCAM project. Alberto's amazing design is an automated, self-recording surveillance system that features a VGA CMOS camera, a passive infrared movement sensor, a 1-GB SD card, and an Atmel ATmega32 microcontroller. This easy-to-install, self-contained surveillance system is the perfect solution for any building.

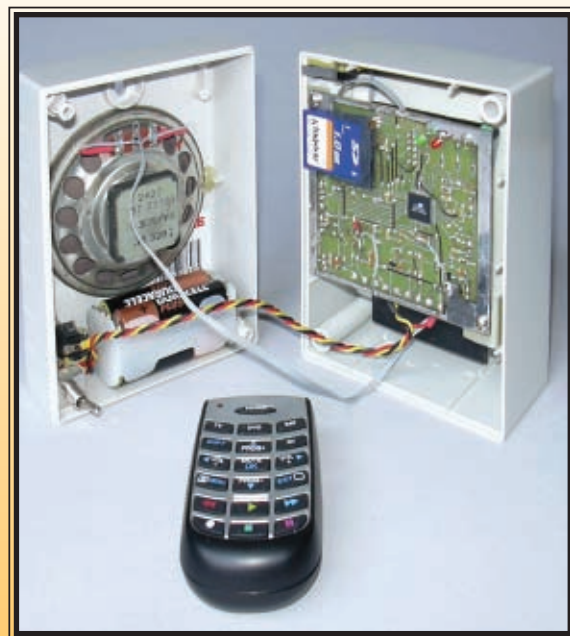
Congratulations to all of the winners!

### GRAND PRIZE

#### WITNESSCAM

The WITNESSCAM is a self-recording surveillance camera that's perfect for the home or office. The innovative, ATmega32-based system features a VGA CMOS color camera, a passive-infrared (PIR) movement sensor, and a 1-GB SD card. The aesthetically pleasing prototype looks like an ordinary alarm detector, but when it detects movement, it silently starts recording. You can control the system with an infrared remote. The interactive camera responds with voice prompts, and the circuit can recognize when the box is open.

*Alberto Ricci Bitti*  
a.riccibitti@iname.com  
Italy



**CIRCUIT CELLAR®**

www.circuitcellar.com

**ATMEL®**

www.atmel.com

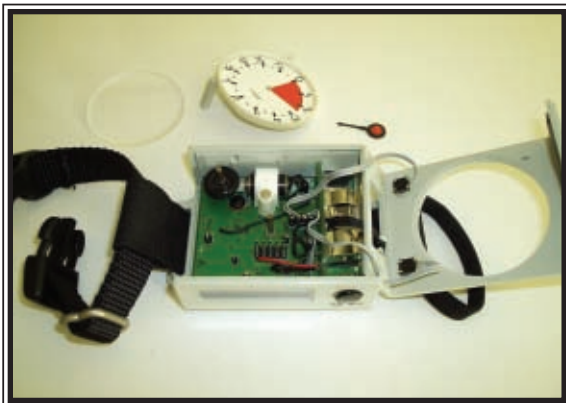
## First Prize

### ATir AVR IR Keyboard Interface

The well-designed ATtiny45-based ATir interface device offers a convenient cross-platform solution to interface an IR remote control to type keyboard macros to a PC. In addition to the microcontroller, the compact system features an infrared receiver/demodulator and a few discrete components. The interface plugs into a PS2 keyboard port on the PC and accepts commands from an infrared remote.



*Steven Savage*  
stevensavage@nls.net  
U.S.



## Second Prize

### Bidirectional Stepper Motor

This interesting ATtiny13-based design takes the principles of current-pulse operated watch hand movement and enables both clockwise and counterclockwise operation. The device, which features a mechanical pointer, provides the altitude indication for a skydiver's altimeter. The pointer mechanism moves in both directions to show altitude on the way up and down.

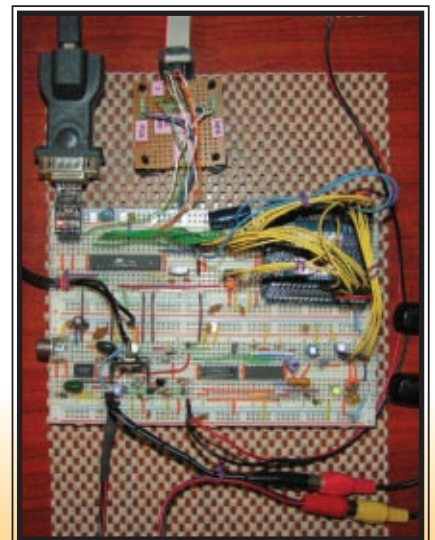
*Chris Belcher*  
stepmotor@ntlworld.com  
U.K.

## Third Prize

### mOtiOn: A Video-Based Motion Sensor

The mOtiOn is an inexpensive video-based motion sensor. It's based on the real-time processing of a video signal using analog preprocessing and optimized image-processing algorithms in an ATmega88 microcontroller. The unique sensor can detect movement in an incoming composite video signal in real time at 30 frames per second. It also conveniently shows where the movement is detected on its video output.

*Naubert Aparicio*  
naubert.aparicio@usa.net  
U.S.





## Honorable Mention

### Talking Calculator

This precise, user-friendly, ATmega88-based talking calculator operates with real numbers. It features the four basic operations (addition, subtraction, multiplication, and division) and several useful functions

(e.g., change of sign (+/-), inverse (1/x), add with memory (M+), and read from memory (MR)).



*Mariano Barron Ruiz*  
ispbarum@sb.ehu.es  
Spain

## Honorable Mention

### Slave Flash Trigger

A camera's external flash can be triggered remotely with its built-in flash. Unlike "photocells," the

ATtiny13-based Slave Flash Trigger is intelligent. It can determine a camera's main flash and synchronize it with the external flash.



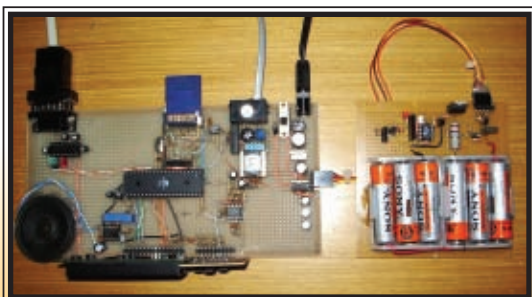
*Aleksander Borysiuk*  
alex\_priv@wp.pl  
Poland

## Honorable Mention

### AVR Phone Recorder and Telephony Platform

This well-planned, ATmega32-based project demonstrates the implementation of a phone line audio and event recorder that was originally designed for technical support call center quality-assurance purposes. The reasonably generic telephony platform can be easily adapted for other applications, such as an answering machine/interactive voice response system.

*Marco Carnut*  
kiko@tempest.com.br  
Brazil

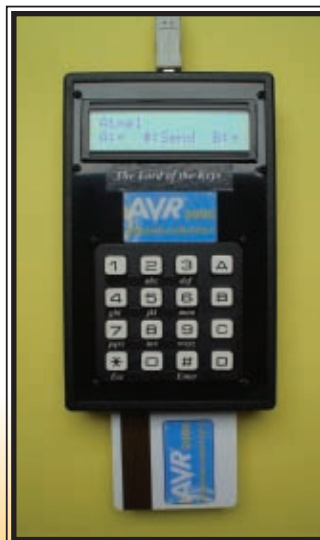


## Honorable Mention

### The Lord of the Keys

Built around an ATmega168 microcontroller, this handy password-managing system enables you to enter, store, and display numerous passwords and usernames. The secure device connects to your PC via a useful software-controlled USB interface.

*Carlos Cossio*  
ccossio@hotmail.com  
Spain



## Honorable Mention

### SAMEgen

The compact SAMEgen is a test generator for the National Weather Service's Specific Area Message Encoding (SAME) coding. The ATtiny45-based system is useful for both the development of decoding circuitry and the testing of SAME-enabled receivers.



*Don L. Jackson*  
don.jackson@ae5k.us  
U.S.

## Honorable Mention

### Doggie 911

The Doggie 911 electronic monitoring system enables dog owners and vets to monitor the patterns of epileptic seizures in dogs. The real-time, ATmega32-based system logs the number of seizures, the duration of the seizures, and the time between seizures so that veterinarians can better diagnose and treat epilepsy.

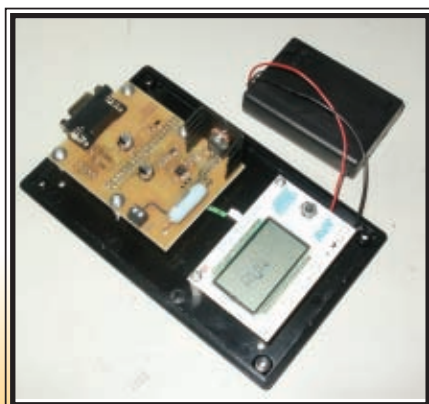


*Steve Lubbers*  
ke8fp@arrl.net  
U.S.

## Honorable Mention

### Dummyload

The simple ATmega169-based Dummyload project simulates loads from 0 to 1,000 mA to help characterize new power supply designs. The system enables you to test at different loading values and test load transients. The microcontroller interfaces with an inexpensive LCD and a joystick to provide a simple user interface.

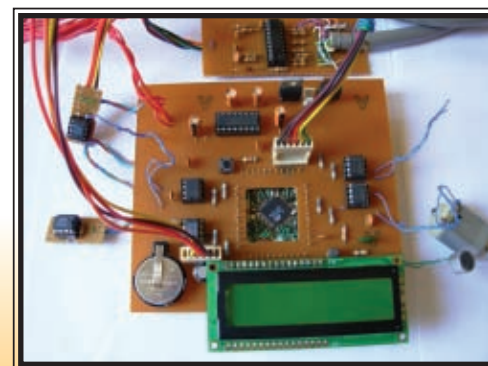


*Kenneth Lumia*  
klumia@adelphia.net  
U.S.

## Honorable Mention

### Automatic Egg Incubator

The easy-to-use ATmega32-based Automatic Egg Incubator facilitates the proper hatching of healthy birds. Two digital thermometer chips serve as dry and wet electronic thermometers. An LCD shows the real-time status of the system, which rotates the eggs and monitors variables such as temperature, aeration, and humidity.



*Niyaz K. Zubair*  
nkz1984@yahoo.com  
India

# Multi-Input Temperature Logger

*Nial designed an FPGA-based multiple-input temperature logger with an innovative USB interface. In this article, he describes the FPGA logic to drive the sensors and the flexible one-pin USB interface used to connect it to a PC.*

Wouldn't it be useful to be able to monitor a number of temperatures simultaneously? I want to be able to measure the temperature in my house, determine the temperature in my refrigerator, verify the internal temperatures in my designs, measure the temperature under the hood of a car I'm tuning up, and more.

I recently came across some inexpensive temperature sensors with an easy-to-drive 1-Wire interface, so I built a system for measuring and logging a number of temperatures. The system can measure more than eight inputs between  $-10^{\circ}$  to  $100^{\circ}\text{C}$ . In addition, it can measure temperatures from a distance (my garage is at the other end of the house). When I interface the system to a PC, I can easily view the data. I kept the logic footprint per sensor low, so I could see how many sensors could be driven from one FPGA.

My background is in digital design (with a very strong emphasis on FPGA design/implementation for the last 12 years or so). As the saying goes, "When your only tool is a hammer, everything looks like a nail." An FPGA seemed like the obvious choice for the project. My immediate thought was to use an FPGA evaluation board, so all I had to worry about was the logic.

In this article, I'll describe the system and my test results. I'll concentrate on the the FPGA logic to drive the sensors and the flexible one-pin USB interface used to connect it to a PC (see Figure 1).

## FPGA BOARD

The logic required to drive

the sensors and implement the USB interface is not that complex. Most modern FPGAs and some larger CPLDs would have done the job. I had an Altera Nios (Altera's soft core processor) evaluation board on hand. This is built around an Altera EP1C20F400C7. In addition to the on-board components required for Nios experiments, it also has two sets of expansion headers that enable an easy connection straight to the FPGA. Although it was complete overkill for what I needed, I pressed it into service (see Photo 1).

## TEMPERATURE MEASUREMENT

As I was flicking through some catalogs, I found a wide range of temperature sensors. I wanted one with a simple digital output so I could drive as many as possible with the aforementioned accuracy and input range. The relatively inexpensive Maxim DS1821 digital thermostat and thermometer (approximately \$6) seemed to meet all of my criteria.

The DS1821 interface is described as

1-Wire, but it seems to use a particular flavor of the 1-Wire interface. My understanding is that most 1-Wire applications allow devices to be addressed on a common interface, whereas the DS1821 allows only point-to-point wiring. It might have been convenient to be able to run a single cable run to remote sensors, but each DS1821 requires only three interface cables, so this isn't a big drawback.

## 1-Wire/DS1821

The 1-Wire interface relies on a data line pulled high with a resistor at some point. All communication is performed by pulling the line down to signal 0 or releasing it to signal 1. When communicating with the DS1821, the FPGA is considered the master and the DS1821 is the slave.

All communication with the DS1821 starts with an initialization sequence where the master pulls the data line low for more than  $480\ \mu\text{s}$  and then releases the line. After a period of time, the slave pulls the line low for 60 to  $240\ \mu\text{s}$  to confirm that it's

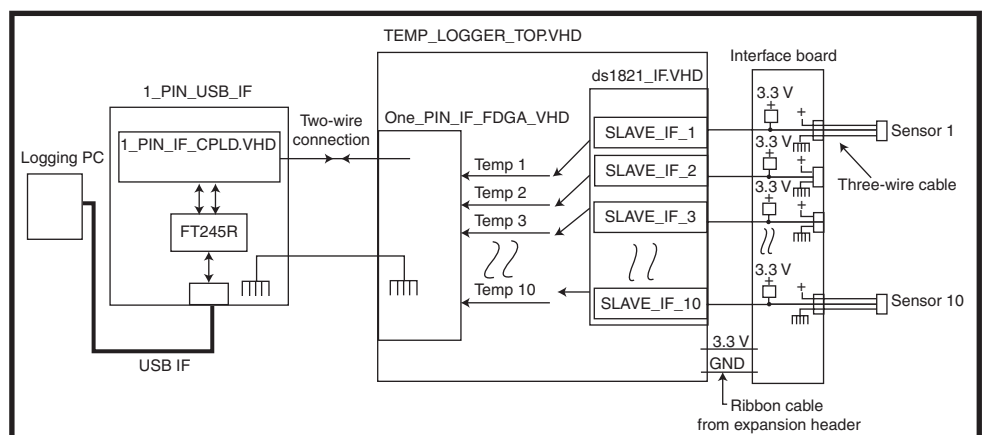
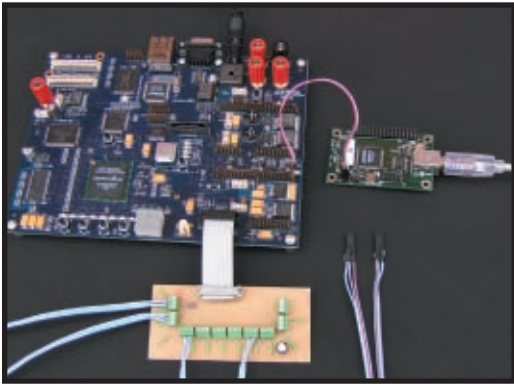


Figure 1—Temp\_logger\_top.vhd is implemented on the FPGA evaluation board. The other two modules are small custom PCBs.





**Photo 1**—Take a look at the entire system. The one-pin interface board is on the right. The sensor interface board is at the bottom. Only four sensors (two are shown) are wired in.

present. The command byte or data byte transfer sequence follows the initialization.

To write a bit to the slave, the master starts by pulling the data line low for 1  $\mu$ s to indicate the start of the bit. To write a zero, the line is kept low. To write a one, the line is released and the pull-up then drags it high. To read a bit, the master again starts by pulling the data line low to indicate the start of a bit. It then releases the line if the slave wants to signal a zero, it must keep the data line pulled low. To signal a one, it does nothing and the pull-up drags it high.

The DS1821 is fairly complex with several different operating modes. I tried to keep things as simple as possible after much experimentation and hair pulling pared down the interface protocol to initiate a conversion and read the result. The DS1821 IF protocol sends the initial sequence, writes 0xEE (Start Conversion Command), waits 0.5 s, sends the initial sequence, writes the 0xAA (read temperature command), reads the result, and goes back to start.

## DRIVING THE DS1821S

Initially, I designed a module that implemented the complete interface to each DS1821. This worked well, but there was a reasonably complex state machine and a large

counter involved, having a dedicated module per device replicates this for each device. Most of this functionality is common across all the devices interfaced. All of the outputs are driven together. The only logic that is different is when device present signal is detected or when the results are being read.

The structure of the design was changed to a master/slave arrangement. The master module (`ds1821_if.vhd`) generates all of the timing and a master output signal. It also generates timing flags to the slave to record if the sensor has replied to the initialization (i.e., if it's present or not) and then to shift in the conversion result. This keeps the logic footprint required for each device interfaced low and allows a single FPGA to interface a very large number of devices (see Figure 2).

The `ds1821_if.vhd` module is split into three main sections. Everything is synchronous and the processes communicate with start and finish flags that are activated for one clock cycle.

## STATE CONTROL PROCESS

The state machine steps through the command sequence. It drives the low-level module with `init`, `wr_byte`,

and `rd_byte` flags. The low-level process signals back that it has finished each task with the `done` flag. The DS1821's operating mode could be changed by changing this process to write and read the appropriate parameters using these commands.

This state machine implements the commands from the high-level state machine, handling all the low-level interface timings and so on. You will notice that all the timing parameters are defined as constants. Initially, I designed the DS1821 interface to run at the FPGA board's 50-MHz crystal speed. When this changed to 80 MHz because of the need to oversample the USB interface, it was easy to change all of the constants to the new values for an 80-MHz clock. (The internal 80-MHz clock is generated with one of the FPGA's four PLLs.)

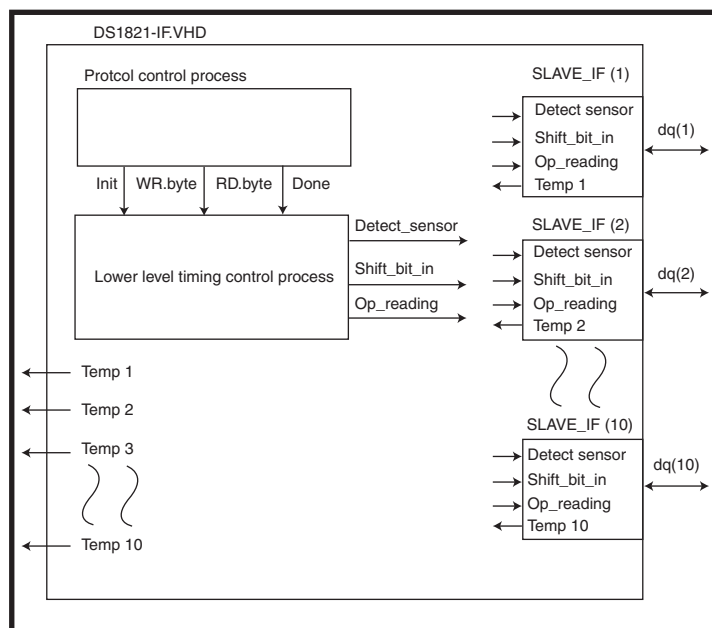
This generates the `MASTER_DOUT` signal that is fed to the data line of each of the sensor data lines:

```
Dq(#) <= '0' (when master_dout = '0') else 'Z';
```

This module also generates the trigger signals to the slave modules interfacing the DS1821s. Every time an initialization is performed the slave gets a `detect_sensor` flag to determine if its sensor is replying or not. When

reading the result byte, the `shift_bit_in` flag indicates to the slaves when to shift the next result bit in, and they should transfer the transient shift register result into the temp output register at the end `op_reading` signals.

Note that `slave_if.vhd` was designed to have a minimal footprint to maximize the number of devices a single FPGA could drive. When a `detect_sensor` flag is received, the status of the data line is sampled and the `sensor_prsnt` output is set if a sensor is detected. The `detect_sensor` flag also



**Figure 2**—A modular approach to the DS1821 interface allows the number of devices interfaced to be easily changed.

resets the internal shift register, which is filled when the `shift_bit_in` flag is activated eight times. When the `output_reading` flag is asserted, the shift register value is registered into the `temp` output if the sensor is present. If the sensor isn't present, the result is set to `0xC9`. (This equates to a temperature of  $-55^{\circ}\text{C}$ .) This last step allows the state of the sensor to be established without having to read the separate `sensor_prsnt` bit. It never gets that cold in Edinburgh, so I thought this was a safe value. This could be removed to reduce the footprint even more.

The slave interfaces and the assignment of the `master_dout` signal to the individual data lines has been done with a `generate` statement. You will see that this easily allows the number of devices interfaced to be increased (see Listing 1).

### DS1821 CONNECTION BOARD

The interface to the DS1821 couldn't be much simpler: a GND line, a 3.3-V power line, and a signal line

**Listing 1**—The DS1821 interface top-level process loops constantly. It reads the sensors via the slave interfaces, instantiated below, which provide the results in an output register `TEMP`. These are then read by the PC over the USB interface at the required rate.

```
slaves_generate:for I in 1 to 10 generate

slave_inst: entity work.slave_if
port map (
  rst           => rst, - in std_logic;
  clk           => clk, - in std_logic;
  detect_sensor => detect_sensor, - in std_logic;
  get_bit       => shift_bit_in, - in std_logic;
  output_reading => op_reading, - in std_logic;
  temp         => temp(I), - out std_logic_vector(7 downto 0);
  sensor_prsnt => sensor_prsnt(I), - out std_logic;
  dq           => din(I) - in std_logic
);

dq(I) <= '0' when (master_dout = '0') else 'Z';
din(I) <= dq(I);

end generate;
```

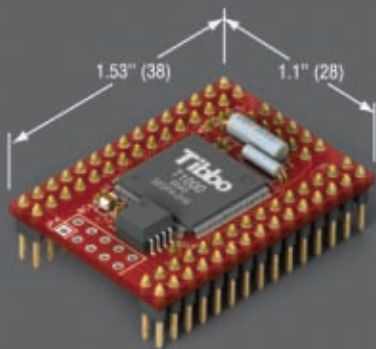
with a 4.7-k $\Omega$  pull-up to 3.3 V. I built a simple one-sided interface board with copper clad board and Press-n-Peel transfer film to allow 10 sensors to be connected (see Photo 1).

To connect the sensors to the interface board, I used ribbon cable split into three wire cables. This was inex-

pensive, although assembling the sensors was fairly fiddly. Small screw terminals were used to connect on the interface board. These work fairly well, although they are fiddly. If a large number of sensors were to be terminated in a semipermanent manner, I would use IDC headers/plugs to ter-

**Tibbo**  
TECHNOLOGY

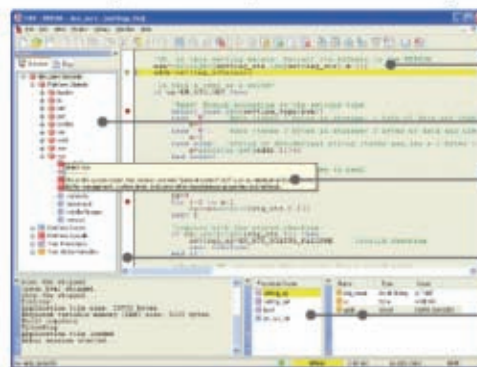
## Build your next automation project around our EM1000 BASIC-programmable Embedded Module



- 50 MIPS CPU
- 100BaseT Ethernet port
- 512K flash disk
- 4x high-speed UARTs
- High-speed parallel slave port
- Real-time clock with backup power
- 49x general-purpose I/O lines
- Development kit available (EM1000-SK)

- Programmable – in BASIC!
- Optimized for real-time applications
- Rich object set
- Built-in webserver
- Event-driven operation
- Sophisticated development environment supports cross-debugging (no ICE needed)

### Code and debug your Tibbo BASIC application using Tibbo Integrated Development Environment (TIDE) software



- Write in familiar BASIC language
- Inspect objects, procedures, and variables
- Code faster with auto-completion and code hints
- Set breakpoints, execute step-by-step, etc.
- Monitor the state of variables and stack

web: [www.tibbo.com](http://www.tibbo.com) email: [sales@tibbo.com](mailto:sales@tibbo.com)



## Back a winner!



Atmel's ARM®-based 32-bit microcontrollers are winners. They have already picked up four prestigious awards from readers of industry leading magazines\*. Why? Because they give you exactly what you want. On-chip Flash memory. USB & Ethernet connectivity. DMA to eliminate internal bottlenecks. Supervisory functions. All this at the lowest possible power consumption and unit price, plus code compatibility across the entire product family. So, make your application a winner by backing a winner: Atmel's AT91SAM Smart ARM-based microcontrollers.

\* EEProductCenter Ultimate Product (Processor & Memory) for Q4 2004 and again for Q4 2005, Embedded Control Europe Gold Award (Micros & DSP) for H2 2005, 2006 EETimes ACE Award Ultimate Product of the Year.

Learn more about our AT91 products by visiting our web site, at [www.atmel.com/ad/at91](http://www.atmel.com/ad/at91) and register to qualify for a free AT91SAM7 development kit. You will also receive an AT91 DVD with extensive product documentation, training material, application notes and code samples.



Check out Atmel's AT91 solutions at [www.atmel.com/ad/at91](http://www.atmel.com/ad/at91)



© Atmel Corporation 2006. All rights reserved. Atmel®, logo, combinations thereof, Everywhere You Are® and others are registered trademarks of Atmel Corporation or its subsidiaries. ARM® is the registered trademark of ARM Ltd. Other terms and product names may be trademarks of others.

Everywhere You Are®

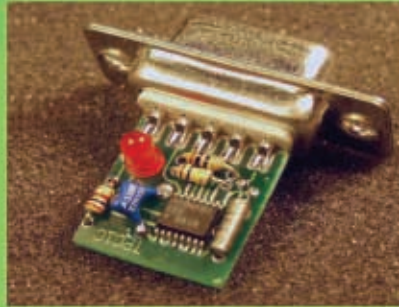
**\$25** EVAL (QTY 1)

**\$7.10** OEM (1K)

# TINY PC!

1/2" FORM FACTOR!  
RUN X86 ASSEMBLY,  
TURBO C OR QBASIC.  
EASY TO USE!

TPC1 is a functional computer complete with CPU, memory and I/O. Not a DIP module, core or "engine". No special cables or development kits to buy. Download programs with XP Hyperterm. Micro-power, run from serial line or battery.



- Built-in FLASH, RAM, EE
  - Up to 6 serial or parallel I/O
  - On-board diagnostic LED
  - PC compatible BIOS routines
  - Hardware Real-Time-Clock
  - Up to 4 channels 12bit ADC
  - Ready to go DB9 connector
- TPC2 (more mem/io) \$50/\$14

## TRUE FRAME GRABBER CREDIT CARD COMPUTER 2

- Color & BW up to 640x480
- Serial/parallel:pic,avr,z80,pc
- Full speed: to 30 frames/sec
- Simultaneous composite out
- Use w/digital CMOS camera
- C, BASIC, Assembler source

oem(1k) **\$27**  
eval kit(1) **\$95**



Add vision at low cost!  
Perfect for inspection, security, robotics. Full frame buffer unlike sx, pci,usb type. Industry std PC104 form factor.



New PLUG-N-GO, no cables/power supply to buy!  
Lo-power RISC cpu 10x faster than PIC, Z80, 8051  
4meg NV mem,ser,par,RTC,4ch 12bit ADC,ISA bus  
Built-in BASIC, Assembly, C compiler included  
Friendly instruction set, unlike PIC or 8051  
CCC2 eval(1)\$75/oem(1k)\$21 CCC1 \$50/\$14

## VOLKSCOMPUTER A CONTROLLER FOR EVERYONE!

PC compatible complete with keyboard & display. Debug programs on the desktop then download to VolksComputer. Super bright LCD. Tactile-click keypad accesses entire alphanumeric char set. 10 bit ADC. Serial (318-2e8) and parallel (378h) ports. VCI: eval \$95 oem \$27 VC2: \$150/\$42



## SERIAL MINI-TERMINAL

RS232 terminal for Stamp, PC, Z80, AVR etc.  
-super low-current, powers from serial line  
-LED backlit LCD, visible in all conditions  
-115.2kbps, DB9 conn, simple commands  
-specify 20 customizable or 16 tactile keys  
eval(1) \$75,oem(1k) \$21.30,w/BASIC cpu \$27



## 640x480 VGA LCD \$27

Controller for most single/dual scan LCDs  
Works with lo-res (160x120, 320x240,etc.)  
Use with PC or SBC, standard VGA BIOS  
Source code demo shows VGA initialization  
Adaptable for other CPUs (i.e. Z80, HC11)  
oem(1k) \$27 evalkit(1) \$95 w/10"LCD \$195



## RFID MODULE \$7

New lower cost unit has longer range and reads more tag types than competitive units.  
On-board power FET drives solenoid directly with card qualify functions. Serial out.  
RFID2 eval (qty 1) \$25, oem (1k) \$7.10

# RS232 TO SD FLASH



HALF THE SIZE, HALF THE COST, TWICE THE VALUE!

Read/Write PC compatible media with Stamp, Z80, AVR, PIC micros. Up to 128 files at once 16 gigabyte max size.

SD1 oem (1k) price \$11.35,  
SD2 \$75/\$21

**\$39.95**  
single piece price

**WWW.STAR.NET/PEOPLE/~MVS**

Alternate: **WWW.GEOCITIES.COM/MEGAMVS**

visit website or  
call for more info  
(508) 792 9507



5yr Limited Warranty  
Free Shipping  
Mon-Fri 10-6 EST

**SERVING THE EMBEDDED  
COMMUNITY SINCE 1979!**

minate a large number of sensors in one operation.

One of the goals was to be able to drive the sensors over a reasonable distance. I unrolled three wires from the full length of my ribbon cable roll (30.5 m/100') and tested to see if the sensor worked, but no go. I changed the pull-up resistor from the recommended 4.7 kΩ to 1.5 kΩ to see if that would improve matters with no more success. On the DS1821, the DATA pin is the middle of the three, so I configured my interface to use the middle of the three wires as the data line. This configuration presents the data line the highest capacitive load. So, as a quick experiment, I changed the data line to use one of the two end cables. Surprisingly, it worked.

## FPGA USB INTERFACE

I'll cover the USB interface portion of the FPGA design later in the article after I've described the one-pin USB interface workings. The FPGA was built with Altera's free web edition of its QuartusII software. The web edition supports development on all devices in Altera's CPLD family, smaller FPGA families, and some of the faster/larger device families. All that is required to replicate the build are the VHDL source files, the Quartus project file (project.qpf) and the constraint/pin allocation file (project.qsf).

The resultant POF file was used to program the EPCS64 configuration PROM on the Nios evaluation board that configures the FPGA at power-on. If you download the project from the *Circuit Cellar* FTP site, you will note the directory structure I tend to use for FPGA development: Altera (where Quartus is run and the FPGA is built), ModelSim (where I run test benches), TestB (test benches written to test designs), and VHDL (the design source directory).

Note that for any significant FPGA development, the ability to simulate designs is

essential. Simulation involves using your design as a subcomponent in a higher-level design to drive the inputs and monitor the outputs. This enables you to test submodules as they are written and debug all of the small functional errors before you get anywhere near hardware. I wrote a test bench that connected the CPLD and FPGA modules and drove the CPLD module's FT245 interface (TestB/tb\_One\_PIN\_IF\_CPLD.vhd). This simulation enabled me to fully test the functionality of the two modules to ensure that they worked together and ensure that they weren't both trying to drive the one-pin line at the same time. This saved debug time and effort and avoided the risk of damage to the devices if both tried to drive the one-pin line in contention during initial debug.

Mentor Graphics's ModelSim is the industry-standard tool for FPGA simulation. It is capable, but it's also very expensive. Symphony EDA's VHDL Simili is a more affordable fully standard-compliant simulator. Xilinx's WebPACK (freely downloadable) also includes a limited version of ModelSim.

## FPGA-PC USB INTERFACE

The facility of a simple PC interface with an FPGA can be extremely useful. It can enable early debugging before other interfaces are finished,

backdoor access to increase debug information, or access to embedded applications with no other direct interface to the device.

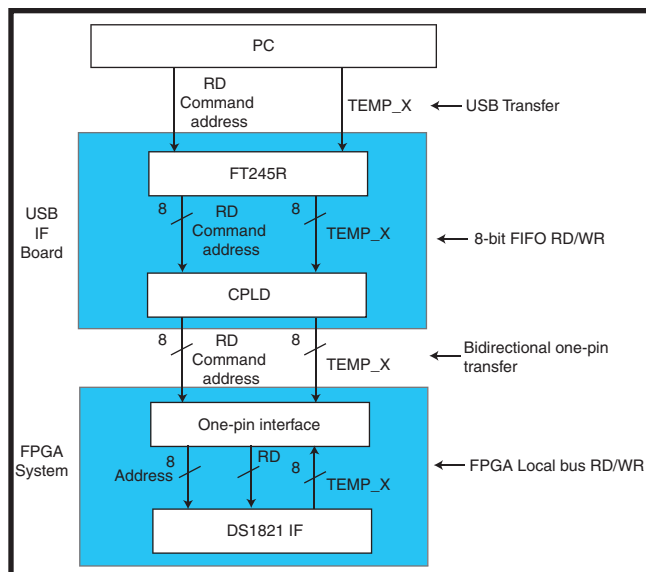
For some time, I wanted a generic USB interface that could be used on any of my FPGA evaluation/development boards and easily adopted in any new designs. But there were a couple of obstacles in my way. First, it would require a dedicated board design/connector pinout for each new application. No two development boards seem to have the same connector pinout. Plus, it's hard to have a generic interface to a number of different boards due to the different header power, ground, and signal pinouts.

In addition, it would require a number of dedicated pins for the interface (a minimum of 12). Normally, this isn't a problem, but that's quite a few device pins to commit for a debug connector that may not be used. Also, where board space is at a premium, adding a connector with grounds (16 pins?) to allow for debugging would occupy a fair bit of board space, especially if a number would be needed (i.e., more than one FPGA on the board).

Implementing a USB interface with an FPGA is not difficult, especially using one of the ubiquitous FTDI interface devices. One drawback associated with FTDI devices is that they currently allow only full-speed USB transfers at approximately 1 MBps. Other options are available to implement a USB 2.0 interface, but these all involve extra design effort. A high-speed version of the FPDI devices would be a welcome addition to their product range. However, raw speed is not really an issue for this project—1 MBps is sufficient.

I have previously designed interface boards around the FT245BM device. The latest FTDI device—the FT245R with integrated termination resistors, a clock generator, and EEPROM—is even easier to use. It seemed like a good basis for this design.

I mulled over how to get



**Figure 3**—Take a look at the data flow through the system (not the physical data paths). The one-pin is a single line. The read temperature command is sent down to the one-pin interface, which selects and returns the appropriate reading.

around the problem of making an interface universal. My initial thoughts were to have a CPLD between the FTDI interface device and a connector header, thereby reprogramming it to route the FTDI pins to the header pins depending on the target. Unfortunately, this doesn't really help because most development boards have different power and ground connections.

I think I was sitting on my usual spot for inspiration (you know where!) when it hit me that I could use the CPLD to serialize the data from the FTDI interface device to the FPGA. Both the CPLD and all modern FPGAs can implement bidirectional data on a single pin! This approach means that all that's needed to connect to the FPGA is a two-pin interface with a signal and a ground connection. The logic required in the FPGA is reduced. The throughput of

the interface is reduced, but for this sort of debug/generic interface, maximum data throughput isn't a design target. Only one FPGA pin is required for this, so it became the one-pin interface.

### ONE-PIN CPLD

With only one data connection between the target FPGA and the USB Interface Board (USBIB), the interface has to operate asynchronously. All serial transfers are byte wide with a single high start bit. The data line is weakly tied inactive low on the USBIB.

All transfers are initiated from the PC and start with a command byte: 0xA0 for a write or 0x50 for a read (only the top 4 bits signify the command). The target address byte is sent next and then data byte for a write operation. These are read from the FT245R FIFO by a CPLD and serially

transferred to the FPGA. For a read operation, the CPLD sends the command and address bytes to the FPGA and then tristates its OUTPUT pin and waits for the data byte back from the FPGA. Again, a single high start bit indicates the start of the transfer. Both devices tristate their outputs when not signaling (see Figure 3).

Although ultimate speed wasn't a goal, I wanted to be able to achieve reasonable data throughput, so a finger-in-the-air data rate of 20 MHz was chosen for the serial interface. I knew that this might prove difficult to meet. It is fairly fast and there's no facility for controlling data paths or impedances when connecting to third-party boards. If this proved impossible to meet, I knew there was always the option of throttling this speed back to a slower rate.

I decided to use an internal 80-MHz clock in both the FPGA and the

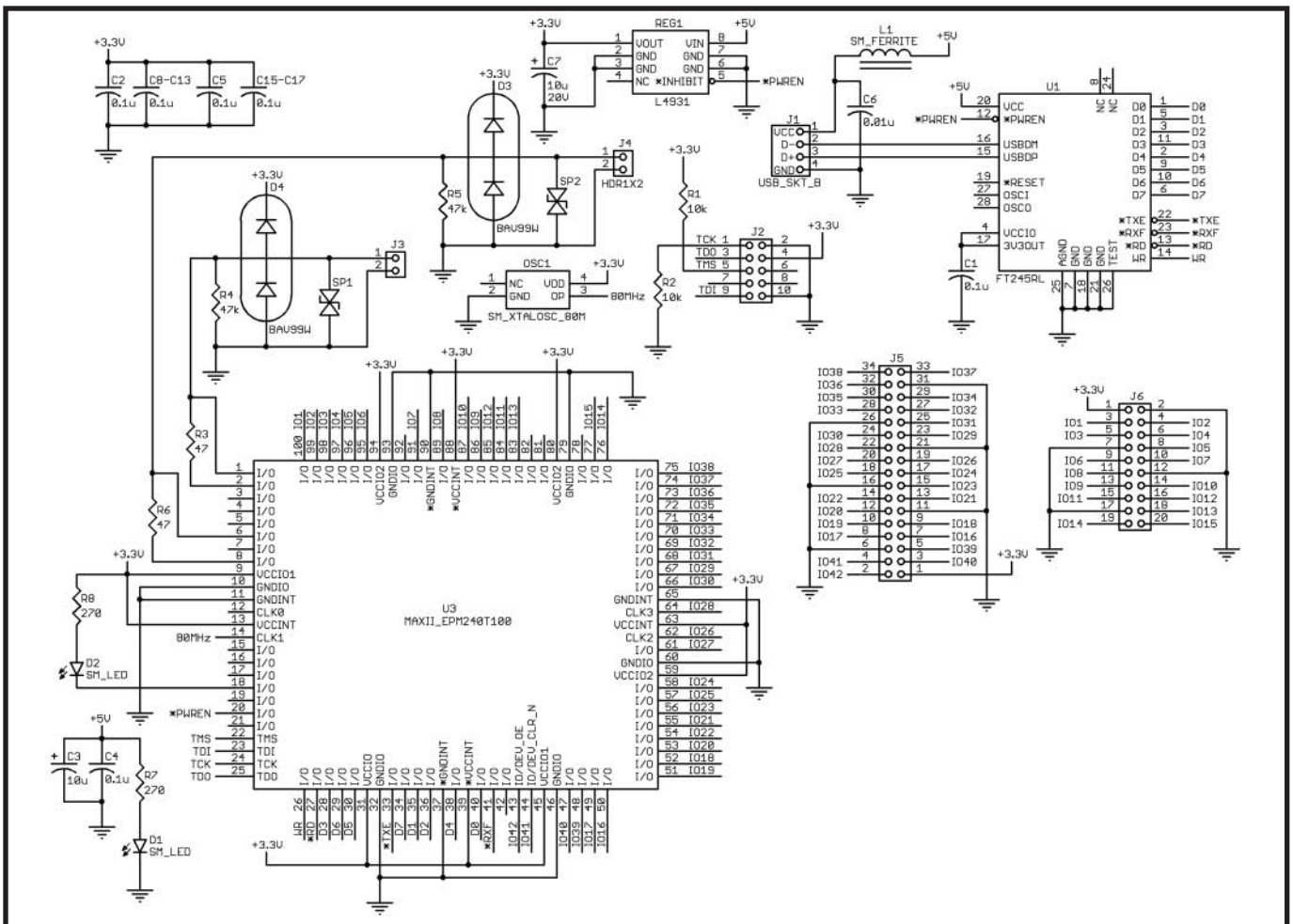


Figure 4—This is a basic implementation of a MAX II CPLD interfaced with an FT245RL USB interface device. The outputs are protected with Littlefuse Pulseguards and diodes to VCC/GND as a belt-and-braces approach to EMC protection. Outputs can be driven with 47-Ω source terminations or straight out.

CPLD for detecting the data and sampling the data. Sampling at 4× data rate enables data to be sampled in a middle region of a bit. There are now two aspects to the USB interface: the USB interface board with the serializing CPLD and the embedded FPGA module.

## USB INTERFACE BOARD

As you can see in Figure 4, there is not a lot to the USB interface board. The FTDI device has so much integrated in it that the USB side of things is trivial.

I built the system around an Altera Cyclone II evaluation board, so I decided to keep with Altera for the CPLD on the USB interface. I chose a MAX II CPLD. I didn't know the final logic footprint of the CPLD functionality when designing the board, so I created a footprint that enables either an EPM240T100 or EPM570T/100 to be fitted.

The 3.3 V for the MAX II device is generated from the USB interface 5-V supply with a local linear regulator. One thing to note is that the USB specification stipulates that the host controller can put any USB device into Suspend mode, where the draw on the interface must be less than 500 µA. In order to meet this requirement, I chose a linear regulator with an enable output. This is driven from the FT245R's PWREN# pin. This only enables power to the rest of the board when the FT245R has undergone USB enumeration and it is not in Suspend mode.

A crystal oscillator provides the 80 MHz. It is positioned as close to the CPLD as possible and connected to one of the dedicated clock INPUT pins.

The interface board must be as robust as possible, so I configured two interface paths. If one CPLD pin is damaged or destroyed, a quick recompile on the Altera tools will enable the other path to be used. In order to protect the input from ESD strikes, a Littelfuse PulseGuard ESD suppressor is positioned as close to the connector pins as possible on the bottom of the board. A BAV99W (two-diode package) protects against over- and under-volt-

Bus ports	Description
Lb_address	Rd/wr address
Lb_d_out	Wr Data out
Lb_d_in	Rd Data in
Lb_wr	Wr Flag
Lb_rd	Rd Flag
Lb_din_valid	Flag in that Rd data is valid. This allows the target being read to delay the interface until it responds with data (usually the next system clock cycle).

**Table 1**—I listed the FPGA one-pin interface local bus signals. These drive the access to the FPGA temperature registers.

age spikes. The capacitance of these two devices is very low (the PulseGuard is less than 0.055 pF and the BAV99W is less than 1.5 pF), so they don't have much effect on the data path.

The configuration of many of the target FPGA evaluation boards is unknown, so it's hard to control the path impedance and terminate things properly for bidirectional communication. Two pins were connected to each output path. One had a straight connection; the other had a source termination resistor of 47 Ω. This enabled experiments to optimize the termination if things did not work.

## CPLD LOGIC

There are three main processes in the CPLD logic. The master process is the protocol process that drives the two processes that interface with the FT245R and drive the serial interface with the FPGA. The FT245 IF process is fairly simple. It processes either a get\_byte or send\_byte flag from the master process to read or write data to or from the FT245R Rx or Tx FIFOs.

The FPGA IF process is simple. It acts on the send\_bits or get\_bits flags from the master process. To send a byte, it is registered into a shift register with a leading one as the start bit. The data is shifted through the shift register every four 80-MHz clock cycles (counted by clk\_count) for 9 bits (bit\_count). A delay is added to the end of a character so the FPGA can distinguish between the end of one and the next start bit.

Note that for both interface process-

es, the asynchronous data or flags are retimed twice into to the 80-MHz clock domain before it is used. This is considered good design practice to protect against metastability.

## FPGA INTERFACE LOGIC

The FPGA interface is simpler than the CPLD with just a master process and a serial interface process. The interface module takes the command, address, and data from the serial one-pin interface and drives a local bus interface (see Table 1).

The master process idles and waits for a command to be received by the serial interface process. A decision is made after a command and address have been received about whether write data is expected or whether the local bus should be driven to read from the target address. If a write command is expected, then the write data is received and then the write is driven out on the local bus for one clock cycle. If a read the target address is read, the serial interface process is then signaled with send\_bits to serially transmit the data to the USB interface board.

The serial interface process acts on either a get\_bits or send\_bits flag from the master process. To receive data, the process waits until a start bit is detected. This is then confirmed and the data clocked into a shift register. Data back to the USB-IB is shifted out as soon as the send\_bits flag is activated. The USB-IB should be expecting it if all is still synchronized.

## ONE-PIN IF TESTING

Using the USB interface board marked the first time I used a MAX II CPLD. When I initially built the USB interface board, I did a quick design that simply flashed the LED to check that I was able to configure the device correctly.

When I first tried to test the one-pin interface, I had baffling problems that I traced to the FT245R interface. I couldn't understand what was wrong until I discovered that the FT245R was permanently asserting the output to indicate that data had been received from the PC. I then realized that the default for unused pins in Altera's

Quartus software (for building the CPLD configuration) is an output driving to GND. This is something to look out for in any new Quartus projects. I had forgotten to change this to “tristate inputs,” and my tests flashing the LED had destroyed the FT245R output.

I built a new board and the interface worked the first time. This is one of the benefits of logic simulation as discussed before!

I had a look at the data on the oscilloscope. It looked fine at first, but there was a signal reflection glitch on the edges. I found that by setting the output drive current on the CPLD pin to a minimum, it improved the shape of the data and almost eliminated the ringing. (Refer to `one_pin_waveform.jpg` posted on the *Circuit Cellar* FTP site for more information.)

## BORLAND C++ BUILDER

I am not a software engineer. I used Borland's C++ Builder (BCB) for this application because it enabled me to build the software quickly without having to worry about any of the details involved with building a Windows application.

One feature of BCB is its ability to include third-party components. This saves you time because you don't have to build your own. As well as displaying and logging temperatures, I wanted to display current temperatures graphically.

A quick search on the 'Net pointed me to several companies that provided thermometer components. I tried the components from Abakus VCL, and they integrated easily. Within 20 minutes, I was able to incorporate its thermometer component in my application. (This would have been quicker if I had known what I was doing.) The results are shown in Photo 2.

## FTDI INTERFACE

FTDI provides royalty-free drivers and software examples for a large number of software development tools and languages. I parsed the C++ builder example and extracted the minimum functionality to drive the interface to the FT245R.

My software is fairly well commented and provides another example of how to drive this device. It's worth downloading and inspecting the software if you're thinking of using one of the FTDI devices. It should be fairly easy to translate this to your compiler/language of choice, especially if you use the other examples on the FTDI web page.

## BASIC APPLICATION

I needed a quick logging application that would enable me to test the system's effectiveness. When the software (`temp_logger.exe`) runs, you have to open a connection to the FTDI device. Press the Connect button on top left. The status label will turn green (“Connected”) or red (“Can't Connect”). If a connection is made, the temperatures are read once every second.

The Turn Logging On button initializes the logging. It also updates the timer component with the value in the Logging Interval entry box. This enables you to vary the logging rate according to the application.

I left in a box and button that enables a write to the FPGA's default register. The value is reflected in the FPGA evaluation board's LEDs. It can be used as a quick test that the interface is working if things don't appear to be operating correctly (useful during debug). The log file (`logfile.txt`) will overwrite any previous versions, so it must be renamed after any logging sessions that will be saved.

As I said before, I'm not a software engineer or graphical interface designer. But this does the job well enough to perform valid logging runs.

## SYSTEM TEST RESULTS

One of my goals for this design was to see how long it would take central heating to fill my hot water tank. I wanted to be able to minimize the amount of time my heating system is on.

My hot water tank comes with about 1" of foam insulation. I found that it was fairly easy to bore out a 1-cm plug with a piece of tubular steel. I bored seven holes (equally spaced down the cylinder) and then

attached eight sensors. I tied one cable to the hot water outlet pipe at the top (see Photo 3). I then ran several log runs to monitor the boiler's temperature as I filled a bath, took a shower, emptied the tank of hot water, and turned on the heating system. The log files for these runs are posted on the *Circuit Cellar* FTP site. Take a look.

## USEABLE SYSTEM

The system works well. The inexpensive DS1821 sensors seem to be accurate and fairly easy to interface. I drove a sensor over 100' of ribbon cable. Refer to Maxim's web site for information about driving the 1-Wire interface over long distances. You can probably improve upon it.

The FPGA logic footprint (per DS1821) is 17 logic elements. The smallest family in Altera's Cyclone II family has 4,608 logic elements. The number of devices that could be driven from most FPGAs will probably be limited by the number of available I/O pins. (The total output current would need to be examined.) This could be hundreds of devices.

The highlight of the exercise was the development of the one-pin interface. The interface provides a flexible way of getting a user interface into an FPGA using only one pin that's useable on most evaluation/protoboards.

There are a couple of disadvantages of the current implementation. If the connector is placed on the target board incorrectly, then the CPLD and FPGA outputs will be driving to ground and potentially damage them. So, polarized connectors with a lock tab should be used to stop this from happening. Only 256 addresses can be accessed, although the bottom bits of the command word could be used to expand this to 4,096 addresses. The data throughput is fairly low. This wasn't a design goal, but if a higher bandwidth is needed, the interface protocol can be changed to enable burst reads or writes from the target.

As usual, this design exercise took much longer than expected. I would like to thank my wife Karen for her support. She entertained our two girls as I worked on this project! ☺



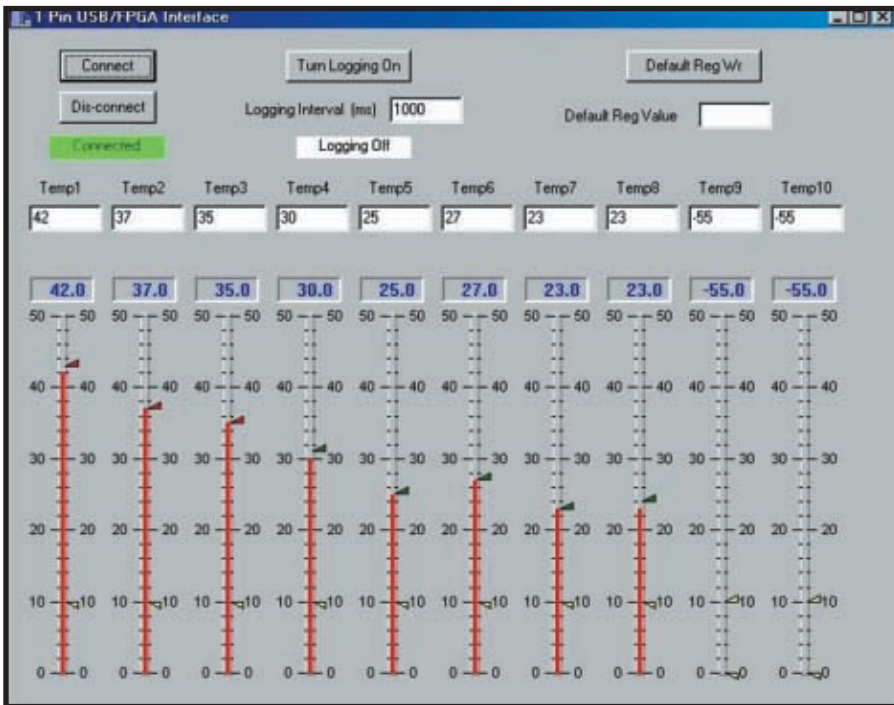


Photo 2—The logging application shows the water temperature at different heights in my hot water tank. Note the obvious temperature gradient down the tank.

Nial Stewart earned a B.Eng. and an M.Sc. at Queens University in Belfast. After working for several years in the telecommunications field, he now runs an FPGA/hardware development consultancy business in Edinburgh. In his spare time, Nial enjoys playing field hockey, messing about in his

garage, and spending time with his family and friends. You may contact Nial at [nial@nialstewartdevelopments.co.uk](mailto:nial@nialstewartdevelopments.co.uk). Type "Circuit Cellar" in the subject line to avoid spam filters.

## PROJECT FILES

To download code and additional files, go to [ftp://ftp.circuitcellar.com/pub/Circuit\\_Cellar/2007/198](ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/198).

## SOURCES

**Abakus VCL**  
A.Baecker  
[www.abakusvcl.com](http://www.abakusvcl.com)

**Cyclone II evaluation board and MAX II CPLD**  
Altera Corp.  
[www.altera.com](http://www.altera.com)

**FT245R IC**  
Future Technology Devices International  
[www.ftdichip.com](http://www.ftdichip.com)

**PulseGuard ESD Suppressor**  
Littelfuse  
[www.littelfuse.com](http://www.littelfuse.com)

**DS1821 Digital thermostat and thermometer**  
Maxim Integrated Products  
[www.maxim-ic.com](http://www.maxim-ic.com)



Photo 3—I waited until my wife was out before I "installed" the system as I thought cutting the foam might have been messy! Luckily, it was easy to bore out small plugs and mount the sensors against the copper cylinder.

# Pololu Robotics & Electronics



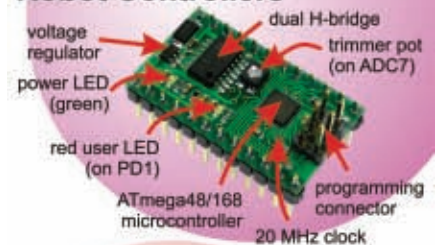
**Robot Kits**  
Line followers  
Robot arms  
Hexapods  
Chassis

**Mechanical Components**  
Gearboxes, servos  
Wheels, ball casters



**Motion Control**  
Servo controllers  
Motor controllers

## Robot Controllers



## Solder Paste Stencils

Use our low-cost solder paste stencils to quickly assemble your surface-mount designs. **From \$25.**



**Custom Laser Cutting**  
**From \$35**

Cut your own custom chassis, front panels, and more!

**1-877-7-POLOLU**  
**[www.pololu.com](http://www.pololu.com)**

6000 S. Eastern Ave. 12D, Las Vegas, NV 89119

# Arby

## An Arbitrary Waveform Generator with a Twist

*The Arby is an arbitrary waveform generator that can easily generate any waveform. The eZ80F91 microcontroller-based system enables you to input waveform requirements in the form of an object-based program written in any text editor and transmitted via the Internet.*

This article is about a system that was never supposed to be presented on its own. The system was intended to be a part of a much bigger project. But it ended up garnering more peer recognition than the larger system.

Originally, we had intended to design and showcase a CCD camera controller, but it soon became clear that the waveform generation portion of the controller was the trickiest part of the project. So, out of compulsion rather than by design, we decided to work on a full-blown waveform generator (an arbitrary one at that). That decision led to the birth of the Arby, which is a Zilog eZ80F91 microcontroller-based arbitrary waveform generator (see Photo 1). The system features a built-in pseudo-compiler that executes the waveform codes downloaded on it using its Ethernet interface.

### WHY ARBITRARY?

Enter any self-respecting electronics laboratory and you'll probably see a waveform generator lying in a corner. As the name implies, it's a device that provides a waveform whose parameters can be programmed. The parameters are generally frequency, amplitude, offset, and so on. Such a device is very useful for experiments and tests, which require a particular signal as a stimulus. One good example would be electronics and communications-related tests.

Generally, waveform generators are function generators, which means they have a few standard and periodic waveform functions (e.g., sinusoidal, sawtooth, or triangular) built into them.

An arbitrary waveform generator, on the other hand, lets you specify a highly custom waveform that doesn't need to be periodic. The signal value at any point of time will be according to your specifications. The application for such a waveform is where you need to test a system by stimulating it with a non-periodic or quasi-periodic input signal.

We made good use of our waveform generator when we needed a CCD chip to test our CCD camera controller. Rather than buy an expensive CCD, we decided to mimic its behavior. We accomplished this by programming the waveform generator to provide a sample waveform—the kind a CCD chip generates once it captures an image.

We realized that we couldn't compete with the specifications of commercial arbitrary waveform generators from companies like Tektronix and Agilent. So, we designed a low-cost alternative that features a general-purpose microcontroller

for waveform generation instead of dedicated, expensive hardware. We fondly refer to our system as the poor man's arbitrary waveform generator!

### SYSTEM IN ACTION

All of the action revolves around a Zilog eZ80Acclaim! evaluation board. Among other things, it has an Ethernet interface built into it. It has a TCP/IP stack and readymade APIs for implementing the applications layer of the TCP/IP stack. We realized that we could provide a fast PC link to the system via the Ethernet without having to go into the nitty-gritty of the standard. This enables us to feed in our waveform requirements directly from a web browser on a PC rather than having to push buttons on the device.

Once we submit our requirements in a predetermined format, the microcontroller processes them and provides the desired 16-bit digital output



**Photo 1**—Take a look at the Arby with the programming interface used during the project development stages. The system generated the waveforms on the logic analyzer.

on the GPIO pins. A simple DAC then converts the 16-bit output to its analog counterpart.

The software APIs supplied by Zilog make the implementation of protocols like HTTP very easy. This means we can host a web site on Arby with user controls and status information. Thus, the system is a truly Internet-enabled device that is definitely a first for any kind of device in this class. The biggest benefit is remote operation.

Figure 1 shows the entire system. The flow of control is from the remote user to the final output waveform.

## INITIAL DEVELOPMENT

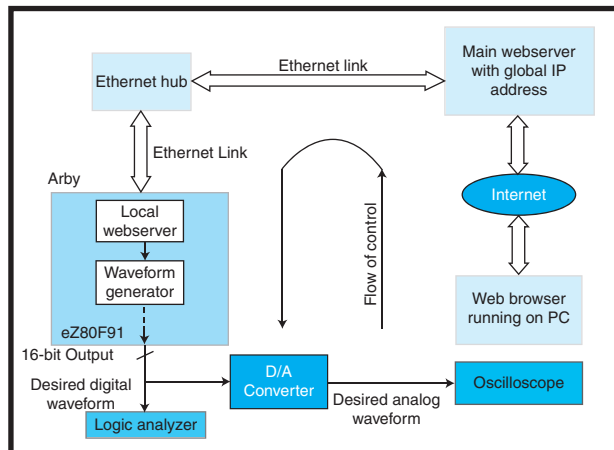
To generate the waveform, each of the 16 GPIO bits must be toggled in a particular sequence. This sequence must be either dictated by the digitized form of a sampled analog waveform or simply by a purely digital waveform. The limitation of such a microcontroller-based approach is that the frequency of the waveform generated is limited by the amount of time the CPU takes to execute the most basic assembly instruction. This time includes the time required to fetch the instruction opcode and any operands from the memory and the time required to decode the instruction. This is why a purely hardware-based solution scores better in terms of performance.

We faced an even bigger hurdle. The format in which the user specified his waveform requirements allowed basic waveform blocks to be chained and nested to make a more complex waveform. Things like looping (i.e., repetition of a basic block) were also allowed. This resulted in a more compact waveform specification and reduced our effort. The downside was that this complex chain of waveform blocks had to be decoded on the fly in the microcontroller core to find out the time interval between two consecutive toggles of a GPIO pin. This operation meant extra work for the CPU, which would increase the minimum possible interval between two toggles, hence decreasing the maximum possible

frequency of the generated waveform.

Our first approach was to use the on-chip timers to perform the task of interrupting the CPU at suitable intervals of time. Between two such interruptions, the CPU would drive constant values on each of the GPIO pins. This interval of time and the constant value was dependent on the kind of waveform and therefore had to be calculated from the input requirements. This meant programming the timers with a particular timeout value, letting them run, and again configuring them with a new timeout value once they had timed out. Thus, even though the time-counting operation was being done through the hardware timers (as opposed to a `do...while` loop in software), the overhead of programming the timers after each timeout affected the toggling rate adversely. Figure 2 shows this with the help of a 1-bit waveform.

The next line of thought was to reduce the input requirements to a sequence of time-value (state) pairs. Whereas the previous approach did the time interval calculations during the waveform generation, this one did a sort of preprocessing and stored the result in the memory. With the preprocessing done, the waveform generation could proceed



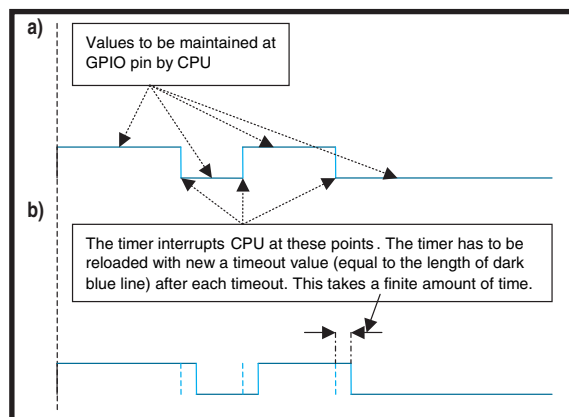
**Figure 1**—You don't need to be near the physical device to program the waveform. Using a global/LAN IP address, you can download either a prepackaged or a new waveform developed from scratch.

without the overhead of having to calculate the intervals. However, this approach needed huge amounts of memory to store the results of the preprocessing. The reason being that even a short user specification—which had some looping and nesting in it—was equivalent to thousands of time-value pairs. The memory required for a decent sized waveform exceeded what we had on chip and onboard. With heavy hearts, we discarded this promising but impractical solution.

## SURVIVAL OF THE FITTEST

Just when things were looking gloomy, one of us thought of a compromise between the first and second approach, one that was quite practical. We could decode the input specifications, and instead of time-value pairs, we could generate the opcode for an optimized delay loop in assembly corresponding to the required time interval. This would be done for each time interval. Once the decoding of the entire specification was complete, the CPU Program Counter could be made to jump to the location where the opcodes were stored and the processor would execute them.

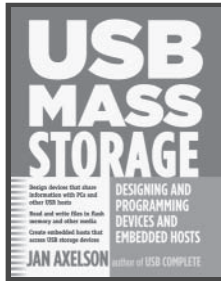
The advantage associated with this approach was that during the generation of the waveform no calculation needed to be done to extract the length of the time interval because that had been done already. Yet the



**Figure 2**—Take a look at the required waveform (a) and the actual waveform (b). The dashed lines show where the transition should have taken place. You can see the differences that arise from the expected behavior in any waveform generator that decodes user-described waveform patterns on the fly.

## ADD MASS STORAGE

### TO YOUR DESIGNS



### USB Mass Storage Designing and Programming Devices and Embedded Hosts

Jan Axelson

ISBN 1-931448-04-3 \$29.95  
Lakeview Research LLC www.Lvr.com

From the author of *USB Complete*

**USB/ETHERNET DAQ**

**LabJack UE9**

Available now for only ...  
**\$429** qty 1

USB/Ethernet Data Acquisition & Control

- \* USB 2.0/1.1 and Ethernet
- \* 14 analog inputs (12- to 16-bit)
- \* Stream input data up to 50 kHz
- \* Use with C, VB, LabVIEW, etc.
- \* Includes DAQFactory Express
- \* Operates from -40 to +85 deg C
- \* 2 analog outputs (12-bit)
- \* 23 digital I/O
- \* Up to 2 counters (32-bit)
- \* Up to 6 timers
- \* Approx. 3" x 7" x 1"

LabJack Corporation, Colorado, USA  
 info@labjack.com, (303) 942-0228

www.labjack.com

ANDRE LAMOTHE'S

## XGAMESTATION

DO-IT-YOURSELF VIDEO GAME SYSTEM

Inspired by the Atari 2600, Apple II & Commodore 64!

SX52 @ 80 MIPS

**INCLUDES:**

- Assembled XGS Micro Edition Unit!
- Complete Development Kit!
- Tools, Demos & Utilities!
- eBook on Designing the XGS Console!
- Cables and Power Supply Included!

WWW.XGAMESTATION.COM

(925) 736-2098 SUPPORT@NURVE.NET

**Listing 1**—*UART.wav* demonstrates how a complex waveform like the UART sequence may be easily represented using *WavCom*. The example includes a subroutine-type call for a child chain that handles all of the data bits. For different data patterns, only the data chain must be modified, while the header for start/stop bit remains the same.

```
% UART.wav

% Main subroutine mimicking a UART
chain 0:
    remain 1 120; % stop bit
    remain 0 20; % start bit
    call 1 1; % data bits generated through a separate
chain
loopend; % repeated endlessly

% Child subroutine handling the data bits
chain 1:
    remain 1 40;
    remain 0 20;
    remain 1 20;
    remain 0 20;
    remain 1 40;
    remain 0 20;
end;
```

memory requirements weren't as huge because the results of the preprocessing were stored not in the form of basic time-value pairs but as loops written in assembly (their opcodes) whose iteration count was fixed. Thus, the core waveform generation code became a self-modifying one, altering itself to the needs of the waveform.

### LANGUAGE GAP

While developing the Arby's core, we also faced an immediate challenge in providing a mechanism to specify the waveform required at the user abstraction layer. Inside the waveform generator device, the waveform exists as an optimized loop in assembly. It can't be expected of the end users to write the assembly program because they may not be conversant with eZ80 assembly. Anyhow, using assembly to specify a complex chain of arbitrary sequences would be too tedious and would defeat our purpose of designing an easy-to-use device.

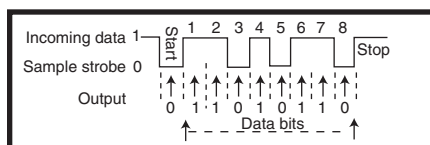
Thus, we were led to implement the waveform at the end-user abstraction layer using a different platform than

that of its inner implementation. None of us were conversant enough with existing compilers such as C++ or lex and yacc to include it in our project. Thus, we arrived at the next best option (which seems to be the best in hindsight) of relying on a pseudo-programming language of our own that would have very limited syntax but support all possible branching, loop, and control instructions in specifying a waveform.

### BIRTH OF WavCom

The end result was a bare minimum compiler—lovingly christened *WavCom* (short for waveform compiler)—with a total of five keywords, three delimiters, and supporting instructions such as wait, loop, jump, end, nested subroutine calls, recursion, and even in-line annotations! To ensure minimum memory usage in the device, the waveform parser was separated and instead integrated with the Java applet-based user interface (which later on acquired the name of *WavCom* instead). This solution also led to minimum data transmission over the Ethernet interface because only the code portion of the waveform (with comments and extra spaces removed) was transmitted to the device.

In the lingo of *WavCom*, each program should have a master chain that completely describes the required waveform. The master chain can in turn invoke further many chains. The master chain indicated as chain 0 is a



**Figure 3**—A generic UART waveform is easy to generate. Using a separate subroutine for data bits allows any variations in data sequence to be easily integrated into the master chain sequence.

functional equivalent of the `main()` function in C/C++ syntax. For the master chain there is an option of having a `loopend` ending in which case Arby will continue generating the master chain in an infinite loop. In all other cases, the Arby acts like a one-shot waveform generator. For driving a value on a GPIO bit, the `remain` keyword is used. Some of the keywords require parameters that are specified in the WavCom file as hex values. Listing 1 generates a UART waveform as shown in Figure 3.

The eZ80 assembly code posted on the *Circuit Cellar* FTP site is generated by Arby for the UART WavCom code. A complete listing of semantics adopted in WavCom is shown in Table 1.

## WavCom TALKS JAVA

Java was chosen as the UI platform because it's easy to use, versatile, and platform/browser independent.

Although we initially had drawn up some grandiose plans of building a paintbrush-like application for drawing waveforms on the web, but barring that, it was clear that the role of a user interface was limited to transmitting the waveform code to the device.

With the development of WavCom (the waveform language), the UI had to adorn the role of waveform parser as well to remove comments and any extra delimiters.

## COMMUNICATION GAP

At the end of the day, the waveform generation occurs at the device abstraction layer. A medium is required to communicate the waveform specifications from the user abstraction layer. In traditional waveform generators, a manual mechanism is provided to control the wave parameters. With the advent of the PC, the RS-232 interface

started to be built in for such devices. In our case, Ethernet serves this role. At a maximum speed of 100 Mbps, our project would be an underutilization of this medium, but the ingenuity of our idea lies in the applications that are enabled by using Ethernet instead of a short RS-232 link, remote on-site debugging being one of them. We will cover this in detail in a later section.

At the device abstraction layer, we used the APIs provided in ZTP—an off-the-shelf TCP/IP stack implementation from Zilog—to build the application layer without having to know much about the internal TCP/IP stack. On the other side, a web-browser-based Java applet was used to open a socket connection to the Arby device from the user end, thus bridging the communication gap.

## DESIGN BLOCK

We used a microcontroller-based

Keywords	Syntax/behavior	Description	Example usage
chain:	chain <chain_num>:	Indicates start of chain definition. All chains are identified by numbers, with the number zero being reserved for the master chain.	% master chain chain 0:  % ordinary chain chain 23:
end	end;	Indicates end of a chain definition. With end, the master chain is executed only once.	chain 11: . . end;
loopend	loopend;	An alternate ending available only for the master chain. With loopend, the master chain is looped into indefinitely by the Arby.	chain 0: . . loopend;
remain	remain <hex_value> <delay_us>	Direction for the output of Arby to acquire a value for the specified delay.	% for a bit remain 1 100; % for a byte remain 0x1234 20;
call	call <chain_num> <iteration_cnt>	Invokes a chain for the specified number of times. Invoking a master chain is not allowed. The master chain can be looped into only by using the loopend keyword.	% call #1 thrice call 1 3;  % call #9 once call 9 1;
%	% Your comment goes here!!!	Used for annotation	%I'm a comment
;	<line contents>; (Line delimiter)	Needs to be present at the end of each line except chain declarations. Text following this is treated as a comment.	end; comment!
,	Word delimiter*	This is what all white spaces/tabs are translated to during the compilation stage.	%User wrote call 4 3; %WavCom said call,4,3;

\*Indicates semantics internal to WavCom. These are hidden from the end user, and they are included only during the compilation stage.

**Table 1**—Take a look at all of the semantics adopted in WavCom along with their syntax, description, and sample usage. All multiple non-code word delimiters are replaced with a comma internally to reduce data size required for transmission over Ethernet. These basic semantics cover most of the major functionalities that can be required in describing a sequential waveform such as looping, subroutine call, and repeating.

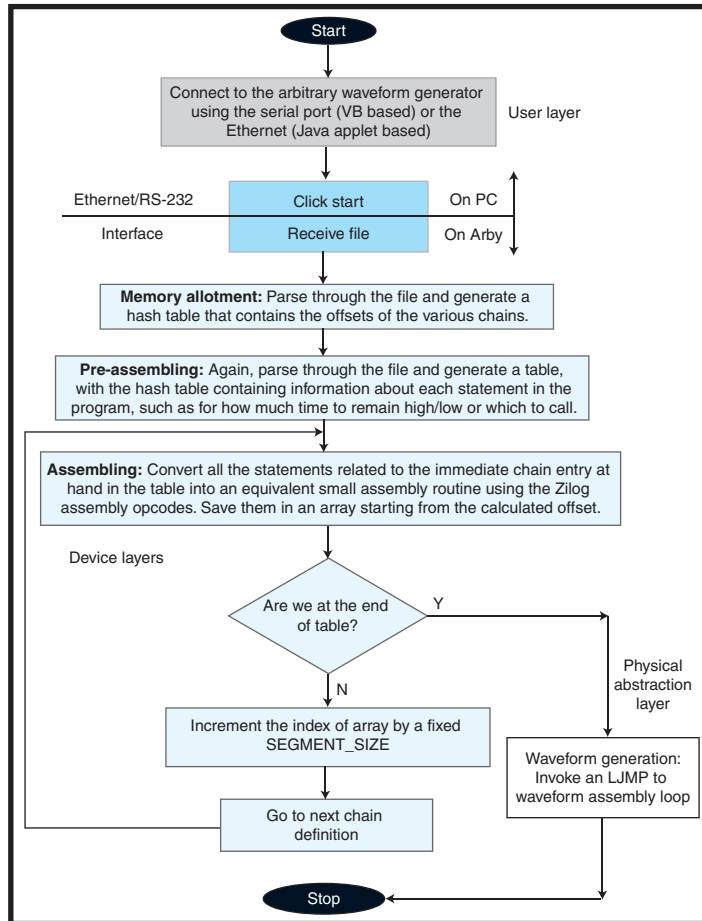
approach to waveform generation (albeit with a twist), so we don't have much to say about the hardware. The eZ80F91 kit is at the heart of the design.

The project includes three different abstraction layers. At the user abstraction layer, the WavCom UI accepts a waveform file from end user, parses it, and transfers the parsed waveform code over the Ethernet medium to the device abstraction layer (eZ80F91). The device abstraction layer then compiles the waveform code to generate an eZ80 assembly equivalent of the waveform. The physical abstraction layer is then brought into context by invoking a JP (long jump) command to the base address of the waveform assembly loop.

An interesting feature to note here is that during the actual waveform generation, the physical abstraction layer is isolated from the remaining layers. Thus, the Arby can be made to generate the previously "downloaded" waveform in a loop without any connection to the outer world. Newly shipped versions can be preprogrammed with a basic test wave pattern.

## DATA FLOW

Figure 4 shows how the system functions. The diagram shows the user, device, and physical abstraction layers, respectively. In our case, Ethernet is the chosen medium of communication between the first two while the latter two abstraction layers physically exist in the same chip. It is the different forms in which waveform exists inside them, which merits their classification as separate abstraction layers. The second box represents the interface between user and device abstraction layers that may be Ethernet or RS-232 (as supported by eZ80F91 microcontroller).



**Figure 4**—Take a look at the data flow at various abstraction layers inside Arby. There are different abstraction layers: user, device, and physical. The Ethernet/RS-232 interface provides connectivity between the user and device layers.

The data flow starts with the user loading the WavCom UI with a waveform file. This waveform file describes the exact waveform as desired by the end user. The UI parses the contents of this file and converts it into a datastream for transmission over Ethernet. On the other end of the Ethernet link, the Arby recovers chain information and allots memory for each chain. In doing so, the shortest possible assembly code equivalent to the functionality described in the corresponding statement is calculated. Using this information, a hash table is generated which correlates chain numbers with corresponding offset addresses in the memory. Finally, using the hash table, the complete waveform (as specified in the master chain) is converted into an equivalent eZ80 assembly routine. The hash table is required to replace its corresponding assembly routine for every chain called inside the master chain.

Once all assembling has been done, a long jump (JP mnemonic in eZ80) instruction is invoked to jump to the base address of master chain routine. Thereafter, the Arby becomes insensitive to any data transmitted over Ethernet as the assembly routine takes over. Once in this mode, Arby can be restored to programming mode only through a physical reset.

## APPLICATIONS

The Arby can be used for a variety of applications like any other arbitrary waveform generator, mostly to simulate use case signals which can either be serial data with glitches or noise, or a modulated carrier wave or a feedback from a control system (like that of an antilock braking system, etc.). Besides these applications, which are common with other waveform generators in the market, this AWG has some unique applications

because of its Ethernet-enabled interface and programming language style way of defining the waveform. We present two such examples to illustrate our point.

## ARBY—ANYTIME, ANYWHERE

The Ethernet-enabled interface enables you to work from a distance and can be embedded in devices that are installed remotely. (Embedding is possible because of the low cost.) This can be used in remote onsite debugging.

Let's say a manufacturer has installed this as a testing aid (of course connected to the appropriate pins and with appropriate multiplexing) in his expensive device and sells the pack to a customer. Just a day later, something goes wrong and the device does not work. Instead of shooting off a tech support person onto the site, the manufacturer can perform few preliminary tests using the Internet to find out that a certain pin has been left disconnected by

the user. The problem is thus resolved without moving anyone an inch closer to the customer's place.

Now let's suppose that there's a faulty device module that the manufacturer must replace and a tech support person is on the way. While on his way, the support person uses his Wi-Fi-enabled laptop to run an extensive set of tests to verify that all modules sharing either interface with the faulty module are working well. This is done by replacing the faulty module with its corresponding waveform pattern for each of the shared interfaces. By the time the support person reaches his destination, he has an all-knowing smile on his face.

## OO WAVEFORMS

The programming style of specifying the waveform gives incredible easiness in specifying complex waveforms, which may be tiresome to specify otherwise. You can actually deal with waveforms in terms of real objects (as in other programming languages). This creates a higher layer of abstraction while dealing with waveforms as com-

pared to the earlier way.

If you need to make a small package for Morse code, you can simply define few small constructs of small waves (chains) and call them dots and dashes. Then these could be used to make a little complex construct of letters and numerals. Once this is there as a library, you can seamlessly write a top-level construct (waveform) and call these letters and numerals in order to have any desired message transfer. If the physical layer of transmit changes—say, the message is now transmitted through amplitude modulation instead of digital logic—the top-level construct will remain unchanged with only the library requiring changes. ☒

*Dhananjay Gadre (dvgadre@nsit.ac.in) is an assistant professor in the Electronics and Communication Engineering Division at the Netaji Subhas Institute of Technology in New Delhi, India.*

*Pushkar Sareen (pushkar.sareen@gmail.com) was a student in the Electronics*

*and Communication Engineering Division at NSIT, New Delhi, from 2001 to 2005. Pushkar works for a reputable firm in the semiconductor industry.*

*Subodh Prabhu (subodh.prabhu@gmail.com) was a student in the Electronics and Communication Engineering Division at NSIT, New Delhi, from 2001 to 2005. Subodh works for a reputable firm in the semiconductor industry.*

*Suhas Chakravarty (suhas.chakravarty@gmail.com) was a student in the Electronics and Communication Engineering Division at NSIT, New Delhi, from 2001 to 2005. Suhas works for a reputable firm in the semiconductor industry.*

## PROJECT FILES

To download code and resources, go to [ftp://ftp.circuitcellar.com/pub/Circuit\\_Cellar/2007/198](ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/198).

## SOURCE

**eZ80F91 Microcontroller**

Zilog, Inc.

[www.zilog.com](http://www.zilog.com)

# Data Acquisition & Control Computer

## iPac 9302

- Cirrus Logic EP9302 ARM9 200 Mhz Processor
- Floating Point Math Engine
- 2 USB 2.0 Host Ports
- SD/MMC Flash Disk Slot
- 40 Digital GPIO Lines
- 1 10/100 Base-T Ethernet port
- 5 channels of 12 bit A/D & 3 PWMs
- 1 RS232 & 1 RS232/422/485 Serial Port
- Battery Backed Real Time clock/calendar
- Linux, CE, & .Net Micro Framework

The iPac has enough I/O for demanding applications & with a size of 3.5" x 3.8" it can fit almost anywhere. Please contact us for more information.

Since 1985  
OVER  
22  
YEARS OF  
SINGLE BOARD  
SOLUTIONS

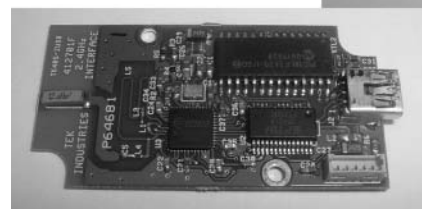
**EMAC, inc.**  
EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: [www.emacinc.com](http://www.emacinc.com)



- ~ Completely Self-contained
- ~ Ceramic antenna included on PCB
- ~ Battery Clip also included on board
- ~ Standard CR2450 Coin Cell Battery
- ~ One Z-TAG and one Z-USB included
- ~ Cables, Cases and additional models sold separately

Z-Radio Engineering  
Development  
Hardware Platform  
Compliant w/  
Microchip ZigBee Stack



**\$49.95**  
Plus shipping  
and handling

TEK Industries, Inc.  
48 Hockanum Blvd.  
Vernon, CT 06066

860-870-0001 Fax: 860-870-0003  
[www.tekind.com](http://www.tekind.com)  
[www.tekrfidshop.com](http://www.tekrfidshop.com)  
[sales@tekind.com](mailto:sales@tekind.com)



# Voltage Solutions

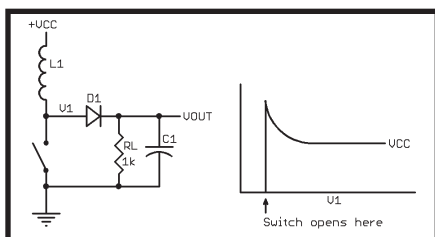
## Harness the Power of Voltage Converters

*Adding voltages to your designs doesn't have to be a complicated task. Stuart explains voltage converter technology and describes the basic principles behind the use of voltage converters in simple designs.*

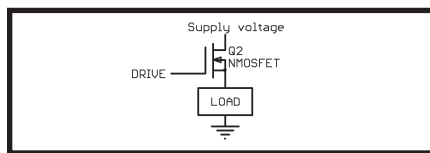
Every once in a while, your designs need a voltage converter to produce a voltage higher than the available supply voltage. For example, you may have a high-side MOSFET driver that switches the supply voltage to a load such as a motor (see Figure 1). Because the SOURCE pin of the MOSFET will be at the supply voltage when the MOSFET turns on, the MOSFET will require a gate drive voltage higher than the supply.

Another possible use for voltage converters is to supply a negative voltage to an op-amp in a single-supply system or a positive voltage exceeding the supply voltage to an op-amp. Other uses include generating bias voltages for RS-232 or other interface devices.

One way to generate a positive voltage is shown in Figure 2. Here, a switch connects one side of an inductor to ground. When the switch closes, current builds in the inductor. When the switch opens, the energy stored in the inductor produces a positive volt-



**Figure 2**—This is a simple switching converter. When the switch closes, current flows through the inductor. When the switch opens, the “flyback” voltage rises higher than the supply. This voltage can be rectified and filtered to provide a DC voltage exceeding the supply.



**Figure 1**—A high-side MOSFET that switches the supply voltage to a load requires a gate drive voltage higher than the supply.

age spike that exceeds the supply voltage. Obviously, a practical circuit will not use a mechanical switch; it will use a transistor instead.

In many circuits, such as a transistor driving a relay coil, this voltage can be high enough to destroy the transistor. It is usually clamped to the supply or to some other voltage through a diode. However, if the voltage is rectified with a diode and filtered with a capacitor, the result will be a positive voltage higher than the supply. The magnitude of the resulting voltage depends on the characteristics of the inductor, the speed of the transistor switch, and the frequency and duty cycle of the input signal.

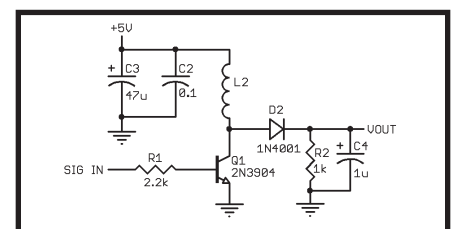
### PRACTICAL CIRCUIT

Figure 3 shows an NPN transistor replacing the switch. The base of the transistor is driven by a pulse train to turn the transistor on and off. When the transistor is on, the low side of the inductor is connected to ground. When the transistor switches off, the low side of the inductor produces the positive voltage spike that is the source for the output voltage. This is a simple concept, but there's a problem. What does the input signal look like, and what

kind of inductor should you use?

I made measurements to determine this information. For this article, I used three types of inductors with the circuit shown in Figure 3. The input signal was adjusted (frequency and duty cycle) to obtain maximum output voltage into a 1-k $\Omega$  load, which corresponds to the maximum power available.

I tested the circuit using 1-mH and 200- $\mu$ H inductors. Two different 1-mH inductors were used. One was a small inductor about the size of a 0.5-W resistor. The other was a large inductor (like the kind you used to be able to buy at RadioShack) that was about 2" long and 0.5" in diameter. Table 1 shows the results.  $T_{HI}$  and  $T_{LO}$  are the high and low periods of the input signal.  $V_{OUT}$  is the output voltage, and  $P_{OUT}$  is the power delivered into the 1-k $\Omega$  load resistor. Two conclusions can be drawn from this table. First, the 200- $\mu$ H inductor requires a higher drive frequency for maximum output. In general, smaller inductors provide faster response to load changes, which is part of the reason they are used in



**Figure 3**—This practical switching converter uses an NPN transistor to switch the inductor current on and off. The frequency and duty cycle of the input signal control the amount of power delivered to the load.



commercial voltage converters. But the 1-mH inductor is probably more suitable to a system where the input may be driven by a microcontroller.

The second result you see in the table is that the larger 1-mH inductor is capable of producing more power than the smaller 1-mH inductor, even though both are the same value. The larger inductor has larger windings and an air core. The smaller inductor is able to get high inductance in a small package by using a ferrite core. When used in a voltage booster circuit, losses in the core of the smaller inductor prevent it from storing as much energy as the larger inductor, so the resulting power output is smaller.

Many commercial power inductors made for power switching circuits will also use a ferrite core. However, they are designed to avoid saturation at the rated power levels so the problem exhibited here does not occur.

For a given frequency, the duty cycle of the input signal also affects the output. For example, using the smaller inductor but driving it with a 10- $\mu$ s high period and 8- $\mu$ s low period results in an output voltage of 8.23 V. For maximum power output, the high input period will last just long enough for the inductor to reach maximum current and then the transistor would be turned off.

Adjusting the duty cycle either side of the optimum value will reduce the output voltage. However, if the high portion of the signal waveform is too long, the transistor wastes power by drawing current through the inductor after it has saturated. For maximum efficiency, the output voltage will be controlled by making the high period shorter than optimum, thereby not allowing the inductor to reach maximum current (and store maximum energy).

Other factors affect the output power as well, including the transistor's turnoff time. MOSFET transistors turn on and off very quickly and have minimal losses in the ON state if the gate is driven with sufficient current, which is why commercial switching supplies use them. In the circuit shown in Figure 3, some increase in output power can be obtained by using

a smaller resistor for R1 or placing a 180-pF capacitor across R1. The capacitor increases the transistor switching speed by allowing the base charge to be added and removed more quickly. The cost is increased current requirements for the circuit providing the input signal.

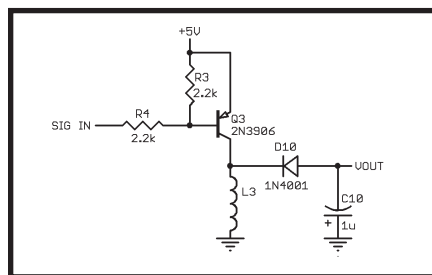
## NEGATIVE VOLTAGES

Figure 4 shows a version of the same circuit that produces a negative voltage that's suitable for biasing an LCD or driving an op-amp. This circuit is a mirror of the previous circuit. A PNP switch connects the inductor to the supply. When the transistor turns off, a negative voltage is produced that is rectified and filtered. Note that the output filtering capacitor has the positive terminal grounded.

The negative voltage generator uses a PNP transistor, so the output characteristics will be somewhat different. For a given inductor, the available power may vary due to transistor switching speeds. If you are attempting to produce symmetrical positive and negative voltages (say, for an op-amp), you'll want to be sure you can get the voltage you need from both versions of the circuit.

## OUTPUT CIRCUIT

The output capacitors in these circuits are 1  $\mu$ F, which is suitable for testing. Some applications will need a larger capacitor. For example, driving the gate of a MOSFET will typically require significant current for fast switching. In that case, you'll want a



**Figure 4**—A PNP transistor is used to generate negative voltages. The circuit is the complement of the positive generator, switching the inductor to the supply and rectifying a negative voltage peak.

Inductor	T <sub>HI</sub>	T <sub>LO</sub>	Frequency	V <sub>OUT</sub>	P <sub>OUT</sub>
Small	13 $\mu$ s	5 $\mu$ s	55.5 kHz	10.4	108 mW
Large	13 $\mu$ s	4 $\mu$ s	59 kHz	15.6	240 mW
200 $\mu$ H	3.5 $\mu$ s	1.2 $\mu$ s	212 kHz	15.7	246 mW

**Table 1**—Take a look at the voltage boost inductor duty cycle measurements. This is the high period of the input signal. T<sub>LO</sub> is the low period. V<sub>OUT</sub> is the voltage across the 1-k $\Omega$  load resistor. P<sub>OUT</sub> is total power into 1- $\Omega$  load resistor (milliwatts).

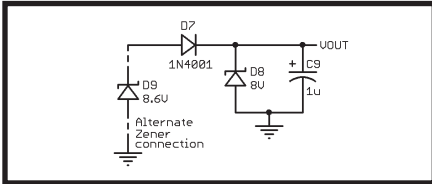
larger capacitor to provide the high initial current.

The 1-k $\Omega$  load resistor was used in the test circuit as an easy way to determine the optimum frequency and duty cycle for each inductor. However, in most practical circuits, you will want to generate a specific voltage, not a specific power into a fixed load. The simplest way to do this is to place a Zener diode across the output (see Figure 5). The Zener diode will always be drawing current from the output capacitor. A commercial switching supply or DC/DC converter will monitor the output voltage and adjust the duty cycle of the drive signal to maintain a constant voltage.

As you can see in Figure 5, you can also place the Zener diode from the transistor collector to ground, but you will need a Zener diode that is about 0.6 V higher to compensate for the forward voltage drop across diode D1. Of course, the negative voltage generator would have the Zener diode reversed (in either configuration) with the cathode connected to ground.

A note of caution here: The 1-k $\Omega$  load in the prototype circuit was used because it draws enough current to effectively clamp the output voltage to a safe value. However, with the right combination of transistor, inductor, and drive signal, it's possible to generate voltages high enough to destroy almost any transistor. Unless you really know what you are doing, I do not recommend operating these circuits without a Zener diode or some other voltage-limiting device on the output.

You may notice that the Zener diode doesn't have a current-limiting resistor. In this case, the Zener diode is used both to prevent transistor damage and to regulate the output. The circuit shown doesn't generate sufficient power to damage a 0.5- or 1-W Zener diode. However, if you want to



**Figure 5**—Adding a Zener diode to the output is a simple way to produce a fixed voltage. This is not the most efficient method because it wastes some power. You can also attach the Zener diode at the collector of the drive transistor (from the anode of D7 to ground).

generate significantly higher power, you may want a high-voltage Zener from the transistor collector to ground to prevent transistor damage and a low-leakage regulator at the output to generate the final voltage (see Figure 6). The transistor-protecting Zener diode needs to be a large enough value to prevent transistor damage and low enough to avoid exceeding the maximum input voltage of the regulator. At higher power levels, it makes more sense to use a commercial voltage booster circuit or a feedback circuit to control the voltage by varying the duty cycle.

## INPUT DRIVE

The test circuit here was driven with a 555 timer that's similar to what you might use in an embedded application. The circuit was also tested with the OCR1B output of an Atmel ATmega8515 microcontroller. This is an output capture signal from an internal 16-bit timer and can be adjusted to produce varying frequency and duty cycle outputs.

You could also drive it from the PWM or timer output of about any microprocessor. In fact, you could use the microprocessor to implement a full feedback-controlled regulated supply by monitoring the output voltage with an ADC and adjusting the PWM duty cycle accordingly. However, in most practical applications, the amount of CPU resources needed makes it more practical to use an off-the-shelf solution if you can't live with a simple Zener limiter.

## CHARGE PUMP

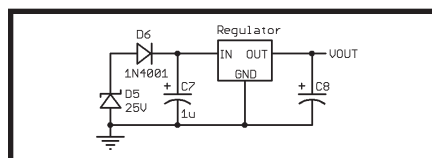
The inductive voltage booster has some problems, including the EMI generated by the switching circuit and the possibility of destroying the tran-

sistor if things don't work right. In addition, the circuit will draw power even when unloaded, which may be a problem for battery-operated equipment. It is possible to build a voltage booster using capacitors instead of inductors.

Figure 7 shows a voltage doubler. This type of circuit has been used for years to generate high voltages in televisions. The way this circuit works is that when the input signal is at ground, C6 charges to the supply voltage (minus the forward drop across D4). When the input signal goes high, the negative end of C6 is at the signal value (typically about the supply voltage). Because C6 is still charged, the positive end is at 2× the supply voltage, which causes C5 to charge through D3. After several cycles of the input signal, C5 will be charged to about twice the supply voltage (minus the diode drops).

One of the circuit's advantages is that the output voltage can't go out of range. It will be only twice the supply voltage. However, a drawback is that half the output power has to come from the input signal. To generate 1 W of output power, the signal has to be able to drive adequate current into the input, which translates to a driver that can both source and sink significant current. For this reason, the voltage doubler is typically used to drive circuits that require very low current, such as bias voltages or supply voltages to CMOS op-amps.

The circuit shown, using 1-μF capacitors and driven by a source with a 100-Ω series resistor, produces 8.5 V with no load and 7.3 V with a 10-kΩ load. The driving signal level drops when the load is applied. So, using a voltage doubler to produce any significant current requires a driver that can maintain the drive voltage with full load applied to the doubler output.



**Figure 6**—For higher power applications, use a transistor-protecting Zener diode followed by a regulator. This improves the circuit's efficiency.

Since the voltage doubler depends on the amplitude of the drive signal to produce the output voltage, it is important to provide a drive signal that swings as close as possible to the positive supply. This will produce the maximum output voltage.

Capacitor sizes depend on input frequency, although larger capacitors are not an issue as long as the signal source can provide sufficient current to charge them at the drive frequency. The circuit was tested with 1-μF electrolytics and the output was the same over a wide range of signal frequencies (about 10 to 100 kHz). One advantage associated with the charge pump approach is that the circuit is more forgiving of the input signal. Just give it a signal with a 50% duty cycle in the appropriate (wide) range, and you will get full output. These circuits are very easy to drive from the timer output of a microprocessor, although you will need high-current drivers to generate any significant power. You can increase the output voltage of this circuit by using Schottky diodes, which have a lower forward voltage drop than the standard rectifier diodes shown.

The circuit can be used to provide voltage that is higher than any available positive voltage by the value of the logic supply (minus diode drops). For example, if the anode of diode D4 were connected to 20 V instead of 5 V, the output voltage would be about 24 V with a 0- to 5-V input signal. The only catch is that the parts have to be sized for the voltage, and the driving source has to be able to sink the current needed to charge the input capacitor from the increased voltage, even during system start-up.

## ISOLATED VOLTAGE SOLUTIONS

Sometimes you need to generate a voltage that's isolated from your control system. An example of this would be interfacing to another system that operates from an unsafe high voltage or from a system in which there is significant difference in ground potential (enough to make a common ground connection difficult or impossible). In such a case, you need an isolated voltage converter.

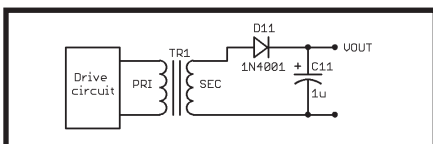
The key component of isolated volt-

age converters is a transformer (see Figure 8). The transformer isolates the voltage-generating side of the circuit from the load because there is no direct connection between the two. The switching supply in your PC is of this type, except the circuitry is fairly complicated. The most impractical part of designing your own isolated voltage converter is the transformer itself. Most switching supplies use custom transformers with multiple windings to maximize efficiency.

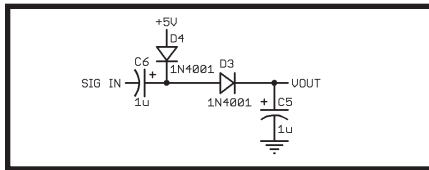
In general, isolated supplies are easier bought than built. If you must roll your own isolated supply, you can use the voltage converter circuit in Figure 3 to drive the primary. But finding a suitable transformer may be difficult. It may be easier to use an off-the-shelf transformer and drive it with a sine signal, either synthesized from the control microprocessor or using discrete transistors and op-amps. This will not be as efficient, but it is probably the easiest way to get an isolated voltage using a standard transformer.

Keep in mind that isolated transformers are not perfectly isolating. The transformer has a breakdown voltage that is a function of the core, the winding insulation, and other factors. If the voltage between the primary and secondary exceeds this value, the transformer may break down and the primary and secondary sides will be connected, either permanently in the form of a short, or momentarily in the form of an arc.

The other issue with transformers is capacitance. There is significant capacitance between the primary and secondary sides of a transformer. Generally, the larger the transformer or the lower the operating frequency for which it's designed, the higher the capacitance. As a consequence, high-frequency energy (e.g., RF energy) may be transferred from the secondary side



**Figure 8**—An isolated voltage converter nearly always uses a transformer to isolate the driving source from the load.



**Figure 7**—This voltage doubler circuit will produce approximately twice the supply voltage at the output. It is most suitable for low-current applications.

back to the primary.

## OFF-THE-SHELF SOLUTIONS

The purpose of this article has been to describe the basic principles of using voltage converters in simple designs. Off-the-shelf solutions are also available (e.g., DC/DC converters from Wall Industries). For driving MOSFETs, there are integrated circuits with internal charge pumps such as International Rectifier's IR21xx series. These devices provide both gate drive circuitry and the ability to drive a high-voltage MOSFET from a lower voltage signal source.

Off-the-shelf charge pumps (e.g., the Maxim MAX1682) are also available. These devices use actively switched transistors instead of passive diodes for better efficiency. The MAX1682 can produce up to 45 mA of current.

## ADD VOLTAGE

As you've learned, adding voltages to your designs is not a complicated task. By selecting the proper components, you can match a voltage booster circuit to your power and drive requirements. Even if you never roll your own voltage booster, you now understand the trade-offs and issues involved in designing one. ☐

*Stuart Ball (stuart@stuartball.com) is an engineer at Seagate Technologies with more than 20 years of experience working with embedded systems. He earned a B.S.E.E. from the University of Missouri-Columbia and an M.B.A. from Regis University in Denver. Stuart has written three books about embedded systems (www.elsevier.com/newnes).*

## SOURCE

IR21xx IC series  
International Rectifier  
www.irf.com

# CIRCUIT CELLAR

Designer's Notification Network

Circuit Cellar design contest entrants have received thousands of valuable development tools and product samples. Because of their contest participation, these engineers receive advance e-mail notice from Circuit Cellar as soon as new samples become available. Now you too can benefit from this early notification.

Welcome to the Designer's Notification Network. Print subscribers are invited to join the Network for advance notice about our new sample distribution programs.

Find out more at  
[www.circuitcellar.com/network](http://www.circuitcellar.com/network)



# Dive Into the ZigBee Pool

## An Easy Way to Start Moving Messages

*You don't necessarily need a ZigBee stack to move small messages from point A to point B. Fred shows you a simple way to get moving with the ZMD wireless sensor starter kit.*

I've become a student of IEEE 802.15.4. The more I learn about it, the more fun I have with it. I recently found a really good IEEE 802.15.4 development kit that puts all of the elements of IEEE 802.15.4 together in an easy-to-swallow capsule.

I'm an absolute fool about anything Ethernet. And based on the way things are going, I'm going to the mad house with IEEE 802.15.4 as well. It's time to take ZigBee and the IEEE 802.15.4 standard seriously as more and more big-time electronic corporations are putting money and effort behind marketing real ZigBee solutions. I really get tired of reading the same old words about what ZigBee is good for. However, even though every ZigBee player is pounding in the way that ZigBee will fit perfectly into such things as home automation and factory floor sensor networks, you've got to give it to them because what they're bellowing about is true. The concept of ZigBee coupled with IEEE 802.15.4 is a beautiful thing.

Speaking of beautiful things, the ZMD wireless sensor starter kit is just that. I recently had the opportunity to test drive the ZMD 900-MHz radio development kit, which includes the Silicon Laboratories C8051F120 microcontroller and associated programming/debugging tools. The most surprising addition to the kit is a 30-day version of Daintree Networks sensor network analyzer (SNA). I say surprising because I didn't expect to see what I saw after I loaded it up. Needless to say, I was compelled to write about my

experiences. So, I'll shut up and get on with describing the wireless sensor starter kit and all of its sidekicks.

### ROLL CALL

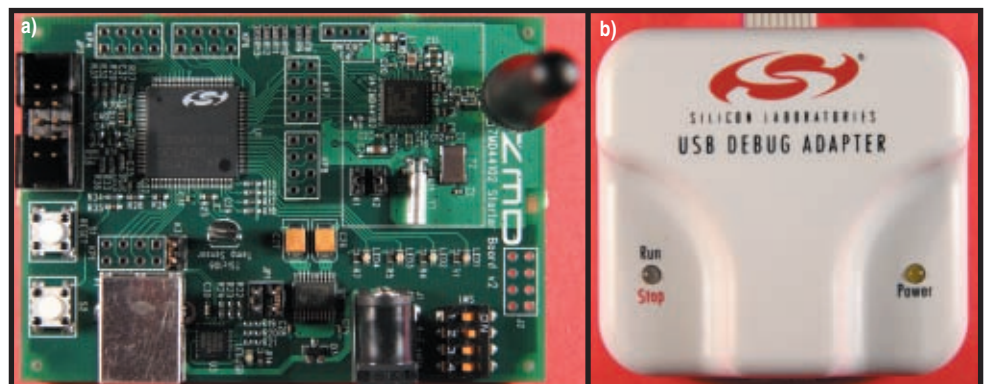
I have the starter kit bundle version of the wireless sensor starter kit. That simply means that my kit includes an extra ZMD44102 starter board that can be employed as a sniffer capture module or an extra IEEE 802.15.4 node. The ZMD44102 starter board contains a C8051F120 microcontroller with an associated JTAG interface and, of course, a ZMD44102 IEEE 802.15.4-compliant/ZigBee-ready single-chip 900-MHz transceiver.

Because the word "sensor" is synonymous with ZigBee and IEEE 802.15.4, the ZMD folks included one of their TSic 106 precision temperature sensors on each of the ZMD44102 starter boards. The Silicon Laboratories portion of the kit bundle includes the C8051F120 microcontrollers, Silicon Laboratories's driver package for the Keil PK51 C compiler, and a Silicon

Laboratories USB debug adapter. A ZMD44102 starter board and the USB debug adapter are shown in Photo 1.

A 4-KB demonstration version of Keil's PK51 C compiler is included with the starter kit bundle. However, you're going to need the full version of the Keil PK51 C compiler to fully utilize the supporting C source code that comes with the starter kit. For those of you that want only to evaluate the ZMD radio or the C8051F120 microcontroller, all of the C source code examples are also provided as ready-to-program hex files that you can push down into the C8051F120 using the included USB debug adapter and Silicon Laboratories flash memory programming utility.

The Daintree Networks SNA package is a professional IEEE 802.15.4 packet sniffer. Everything you need to know about a transmitted IEEE 802.15.4 packet is available via the SNA's application windowpanes. Fortunately, the SNA demonstration included with the sensor starter kit is



**Photo 1a**—As development kit boards go, this one is pretty clean. The ZMD44102 doesn't need much of anything hung on its pins to do its job, and the same can be said of the Silicon Laboratories C8051F120 microcontroller. **b**—This little device was also a pleasant surprise. It was very easy to integrate into the Keil  $\mu$ Vision3 scheme. And, it's simple and it works.

the professional version, which enables you not only to see the bits, but also to visualize a ZigBee-enabled IEEE 802.15.4 network.

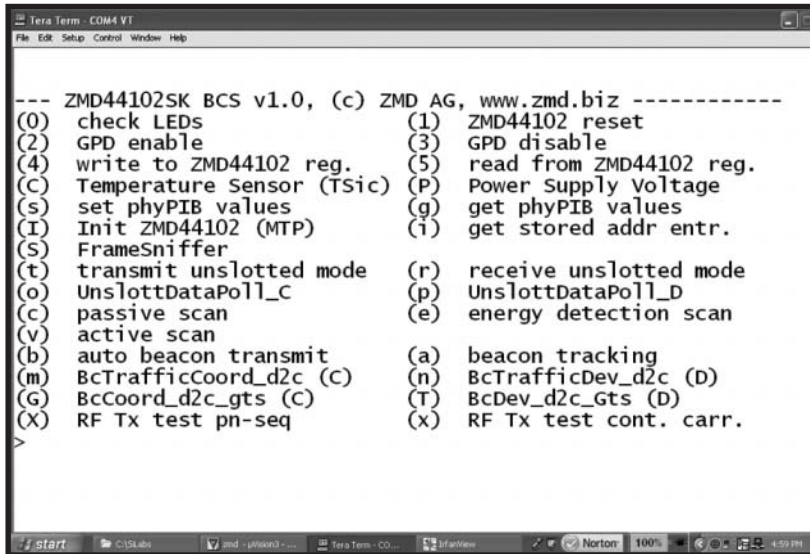
The ZMD44102 CMOS transceiver is an 868.3/902-to-928-MHz band single-chip multichannel IEEE 802.15.4-compatible system-on-chip device. The low-power 40-Kbps ZMD44102 transceiver is also equipped with a complete IEEE 802.15.4-compliant PHY and a thin MAC

that is implemented in hardware. Most IEEE 802.15.4/ZigBee transceivers use a SPI portal to communicate a host microcontroller.

The ZMD44102 employs SPI, but it is also capable of passing data and commands over a parallel interface. Another interesting feature of the ZMD44102 is the use of dual clocks. The 24-MHz system clock is used to clock the digital core and a separate 32.768-kHz clock, which supports Sleep and Power Down modes, acts as an RTC. Both clocks, in one manner or another, support a set of IEEE 802.15.4-oriented timers that are integrated into the ZMD44102's HW-MAC.

I've been reading and rereading the IEEE 802.15.4 standards document. It is refreshing to see that the wireless sensor starter kit documentation uses the IEEE 802.15.4 standard's nomenclature in the datasheet and user manual language. I'll do the same within this column when I can.

The IEEE 802.15.4 stuff combined with the ZMD44102 stuff makes for days of enjoyable reading. The good news is that the ZMD44102 part and its associated documentation are very logical. If you pay attention (no speed reading), you can read it all in a dark room as the proverbial light bulb will be illuminating a bunch.



**Photo 2**—This is a playground. We can do anything we want to do here with little fear. For instance, we can transmit a raw IEEE 802.15.4 packet and receive it at the other end node if we desire.

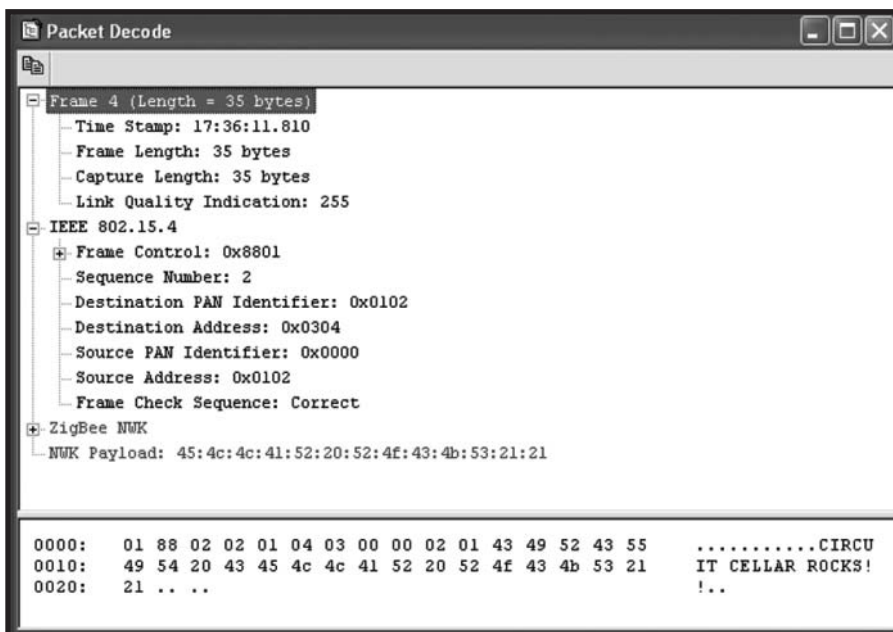
thing set up and ready. The demo application code and Silicon Laboratories drivers that came with my wireless sensor starter kit were designed to be used with Keil's  $\mu$ Vision2 environment. I have the latest version of the Keil PK51 C compiler, which is built around  $\mu$ Vision3. I found that the Silicon Laboratories USB debug adapter needed to have the  $\mu$ Vision3 drivers loaded in order to turn on the recognition of the USB debug adapter's USB interface from

and was pleasantly surprised when the menu in Photo 2 appeared in my Tera Term Pro window.

With the PK51 C compiler/USB debug adapter/BCS source code integration behind me, I moved on to the next task of preparing the SNA and the IEEE 802.15.4 frame capture hardware. I decided to use the third ZMD44102 starter board as the frame capture hardware for the SNA. A pre-compiled and ready-to-load Daintree driver hex file was included with the

within the Keil PK51 C compiler's debug utility's set-up windows.

Once I got the PK51 C compiler to recognize the USB debug adapter, I performed a test compile of the basic communication software (BCS) source that came with the wireless sensor starter kit. Everything came out lovely, and I punched the  $\mu$ Vision3's Debug button to push my new BCS hex file into the C8051F120's flash memory. I clicked Go



**Photo 3**—I remember doing this with raw Ethernet frames. It is really interesting what sniffers try to do with the crap they sometimes may get. Even with the obvious 01-02-03-04 number sequence, the Daintree Networks SNA still gives a good picture of what's really supposed to be going on here.

## LET'S DANCE

I had no problems getting every-

ZMD wireless sensor starter kit. All I had to do was load up the Silicon Laboratories stand-alone flash memory programming utility, which also was a component of the wireless sensor starter kit, and drop the SNA driver code into the C8051F120 flash memory on the newly tasked IEEE 802.15.4 capture module.

There's plenty of good coding fodder to be found in the BCS source. Thus, my goal was to ultimately build a regulation IEEE 802.15.4 packet using BCS functions and send it along its way from one of the ZMD44102 starter boards. While the bits were flying, I captured them all with the SNA.

## BITS IN THE WIND

I decided to throw caution to the wind (along with a few bits) and use the BCS "transmit unslotted mode" to fling a message out of the ZMD44102's cute little 900-MHz rubber ducky antenna. I recorded the event for posterity with the SNA and posted the resultant capture in Photo 3. There is no official ZigBee network. The visualization feature of the SNA couldn't draw up a pretty picture of the network. Besides, there really wasn't a network at that point because I was shooting IEEE 802.15.4 skeet with an SNA shotgun.

If you analyze the contents of Photo 3, the actual frame length calculation, which is gathered from the PHY protocol data unit (PPDU) header area, is correct. Including the pair of CRC bytes that are not shown at the end of the hex dump, I count 35 octets in the hex dump area of Photo 3. How about you?

The Link Quality Indication byte is maxed out at 0xFF (255), as it should be. The IEEE 802.15.4 capture module is less than 1 m from the transmitting station. It also looks like the Frame Control word was meant to be accurate because the Destination Addressing and Source Addressing modes are 16 bits in length. All of the addresses are obviously simply enumerated.

The SNA goes totally whacky on the text message because it thinks this is stuff destined for the network (NWK) layer of a ZigBee stack. In fact, there is absolutely no ZigBee in this gaggle of bits at all. Everything you see is actually IEEE 802.15.4 PHY and

**Listing 1**—The ZMD wireless sensor starter kit source code set is very well written. The use of IEEE 802.15.4 terms makes the code self-documenting. Everything you want to do with a ZMD44102 can begin with the functions contained with the BCS source code. I inserted the root ZMD function code inside of the `plmeSetRequest` and `cmdUnslottedTX/cmdUnslottedRX` functions for clarity.

```
void cmdUnslottedTx (void)
{
    unsigned char ack;
    unsigned char channel;
    unsigned char tx_data[125];
    unsigned char payload_length = 0;
    unsigned int destpanid,destaddr;
    destpanid = 0x5050;
    destaddr = 0x4444;
    channel = 0x01;
    ack = 0x01; //request an ACK from the receiver
    tx_data[0] = 0xAA;
    payload_length = 0x01;
    plmeSetRequest(phyCurrentChannel, &channel);
    /* FROM plmeSetRequest/START
    switch (PIBAttribute)
    {
        case phyCurrentChannel:
            if (*(UINT8*)PIBAttributeValue > 10) return PHY_INVALID_PARAMETER;
            phyPIB.phyCurrentChannel = *(UINT8*)PIBAttributeValue;
            zmdWriteReg(PHY_CHANNEL, phyPIB.phyCurrentChannel);
            break;
        /* FROM plmeSetRequest/END
        zmdUnslottedTx(ack, tx_data, payload_length, destpanid, destaddr,
        AM_SHORT_ADDR);
        /* FROM zmdUnslottedTX function/START
        if (addr_mode == AM_SHORT_ADDR)
        {
            zmdWriteReg(MHR_TX_DST_PAN_ID_1, LOW_BYTE(dest_pan_id));
            zmdWriteReg(MHR_TX_DST_PAN_ID_2, HIGH_BYTE(dest_pan_id));

            zmdWriteReg(MHR_TX_DST_ADDR16_1, LOW_BYTE(dest_addr16));
            zmdWriteReg(MHR_TX_DST_ADDR16_2, HIGH_BYTE(dest_addr16));

            zmdWriteReg(MHR_TX_SRC_ADDR16_1, LOW_BYTE(macPIB.macShortAddress));
            zmdWriteReg(MHR_TX_SRC_ADDR16_2, HIGH_BYTE(macPIB.macShortAddress));

            //SET HIGH BYTE OF FRAME CONTROL WORD
            zmdWriteReg(MHR_TX_FC_2, FT_DST_ADDR_MODE_16 | FT_SRC_ADDR_MODE_16);
        }
        if (ack == ACK_REQUEST)
            //SET LOW BYTE OF FRAME CONTROL WORD
            zmdWriteReg(MHR_TX_FC_1, FT_DATA | FT_ACK_REQUEST | FT_INTRA_PAN);
            //Assemble the packet with the length byte first,
            //then the rest of the packet
            zmdWriteTxFifo(payload_length, payload);
            //SEND IT!
            zmdWriteReg(MAC_CTRL, MC_TX_ON);
            /* FROM zmd UnslottedTX function/END
    }

void cmdUnslottedRx (void)
{
    UINT8 channel;
    UBYTE data_buffer[aMaxPHYPacketSize];
    UBYTE i,j;
    channel = 0x01;
    plmeSetRequest(phyCurrentChannel, &channel);
    while (1)
    {
        if (RX_FAILED == zmdUnslottedRx(&data_buffer, TRUE))
            break;
        else
        {
            wait_ms(5);
            {
```

(continued)

Listing 1—Continued.

```
{
    for(j=0;j<20;j++)
        i = data_buffer[j];
    }
    P3 = data_buffer[10];
}
}
```

MAC related. All that was done here was stuff the Frame Control bits, make up some addressing words, stick a text message behind it all, and cram it into the ZMD44102 transmit FIFO. The ZMD44102 added the necessary encapsulation—preamble, start of frame delimiter (SFD), and CRC—and pushed the entire mess out the antenna pipe.

## NO HARM, NO FOUL

There is absolutely nothing wrong with what I did. As long as the receiver and the receiving application know what to do with the incoming frame, who cares about its format? For instance, the receiver could have simply parsed or counted through the frame control and addressing fields of the IEEE 802.15.4 frame I sent and picked out what it wanted to use of the text message. Or, if there were more than one receiver, the address information could also have been parsed and logically analyzed to determine who the message was really aimed at. That's essentially what a ZigBee stack does, but it does it quite a bit more elegantly.

Let's be a bit more elegant as well. I say you build up a simple unslotted network and send a meaningful byte or two over it. The term "unslotted" means that the little ad hoc IEEE 802.15.4 network will not use beaconing, which means there will be no special, exclusive or repetitive time slots allocated for data transfer timed to a recurring beacon signal. In other words, if the coast is clear, fire when ready.

All you'll really need to do is to determine if you want an acknowledgement for the transmission, identify some specific transmitter and receiver PAN and node addresses, specify the length of the data you wish to send, and plug in your data. To be honest, the ZMD folks have done most of the work. Once you have nailed down who and what, all you

have to do is call upon some of the functions included within the BCS.

OK. You definitely want an acknowledgement because that will force the receiving MAC to generate some ACK traffic. Starting from outside in, let's designate a WPAN address of ASCII "PP," which equates to a 16-bit WPAN address of 0x5050. The transmitter (source) address will be ASCII "SS" or 0x5353, and you'll set the receiver (destination) address to ASCII DD (0x4444). I failed to mention that it may be nice to choose a channel in the 915 MHz ISM frequency band to operate on. So, let's go with channel 1.

## CODING THE SOURCE NODE

The initialization of the ZMD44102 starter boards is taken care of up front in the main function of the BCS. So, rather than fix something that isn't broken, I left it intact. I did, however, make some changes to the unslotted transmit and unslotted receive routines you see in Listing 1.

The cmdUnslottedTX function code in Listing 1 is rather obvious as to what was done considering you and I set forth the addressing scheme and selected the operating channel beforehand. Most every IEEE 802.15.4 and ZigBee implementation uses the 16-bit addressing modes rather than the IEEE 64-bit addressing modes. The less stuff you're sending, the more power you're saving.

In the IEEE 802.15.4 specification, the PHY is defined with integral data (PD-SAP) and management (PLME-SAP) service access points (SAPs), which are logical portals that pass primitives between the PHY and MAC sublayers. Primitives are simply data structures that contain information that can be used and built upon by the neighbor sublayer the primitive is passed to.

The plme portion of the plmeSetRequest function stands for PHY layer management entity. The PLME, as it is called in the IEEE 802.15.4 spec, is responsible for coordinating any management functions associated with the PHY. In this case, setting the current operating channel is a PHY management function. Because the PHY's PLME-SAP and PD-SAP are the only SAPs that join the MAC and PHY sublayers, the MAC will always pass the final management or data primitive to the PHY sublayer, and any data the PHY needs to pass up the stack will always flow through the MAC.

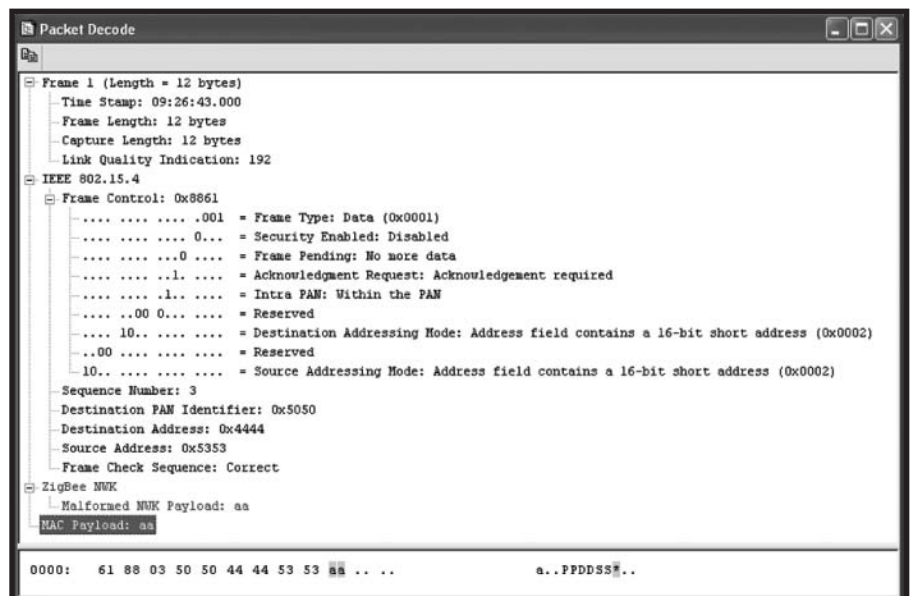
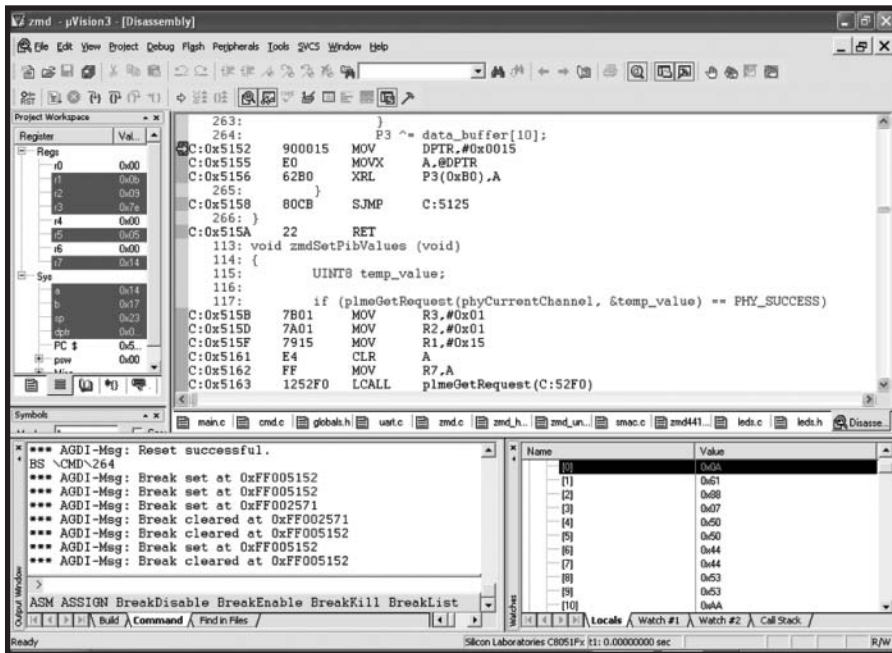


Photo 4—Here's that IEEE 802.15.4 packet we've been going after. Compared to a typical 802.11b packet, this packet is simple, short, and sweet.



**Photo 5**—I listed the hex bytes here just in case you have trouble reading the `data_buffer` array contents: `0x0A 0x61 0x88 0x07 0x50 0x50 0x44 0x44 0x53 0x53 0xAA`. The first byte represents the packet length. The `0x07` is the sequence number, which matches up with the acknowledgement sequence number in Photo 6.

Almost all of the IEEE 802.15.4 packet building is done within the `zmdUnslottedTX` function. The frame control word is being assembled in the correct order inside of the `zmdUnslottedTX` function. The correct build order is done here for clarity as the ZMD44102 has the ability to build an IEEE 802.15.4 frame by combining data from dedicated frame function registers (the ZMD44102's frame control registers, sequence number register, etc.) and frame data stored in general-purpose registers.

The frame control word resides inside of the MAC protocol data unit (MPDU) header, which is identified by MHR (MAC header) in the IEEE 802.15.4 specification. I added the `FT_INTRA_PAN` bit to the low byte of the Frame Control Header because there will be no intra-PAN communications in our simple little IEEE 802.15.4 peer-to-peer network. I also specified the `macPIB.macShortAddress` value in the MAC PIB PAN information base (PIB), which in the case of the BCS is kept inside the `mlmeResetRequest` function.

If you follow the IEEE 802.15.4 rulebook, the logical location of the MAC PIB is inside of the MAC layer management entity (MLME), which is part of the MAC sublayer. As you study

the BCS source code, you will find that some of the elegance associated with the IEEE 802.15.4 specification has been converted into logic that is more efficient in the real world. For instance, many official IEEE 802.15.4 request primitives by rule will cause the generation of an official IEEE 802.15.4 confirm primitive that has to be passed to complete the operation. If an IEEE 802.15.4 confirm primitive operation can be simulated with a sim-

ple return of a value, the BCS will jump all over that. Listing 1's coup de grace is the final assembly of the packet length followed by the packet payload into the transmit FIFO just before the packet is fired down the antenna trace.

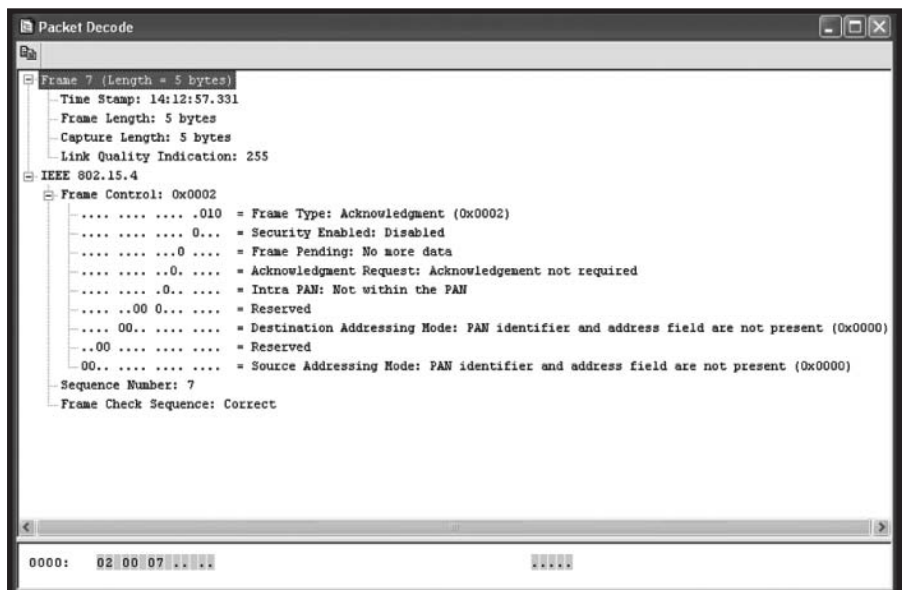
Photo 4 shows what happened when I kicked off the transmission code. The packet length byte is picked off by the PHY. Note that I coded only a `0x01` for the payload length. The ZMD44102 and BCS took care of the rest. I turned on the intra-PAN bit in the frame control word, which effectively eliminated 2 bytes (source PAN identifier) from the addressing fields. Everything in the addressing fields matched the predetermined values, and a single byte of data is sitting just in front of the pair of CRC bytes.

The SNA knows that NWK information should immediately follow the addressing fields and tries its best to make some sense of it. You know that the data payload is in the NWK spot and apparently so does the SNA as it reported a malformed NWK payload.

The transmitter node works. So, let's go work on getting the receiving node up and running.

## CODING THE DESTINATION NODE

The receiver node code can be seen in the lower portion of Listing 1 beginning with the `cmdUnslottedRX` function. The Boolean variable



**Photo 6**—We don't have ACK smarts above the MAC layer to generate any messages. Thus, this acknowledgement message was generated by the ZMD44102's MAC.



auto\_ack is indeed true because things get turned over quickly to the zmdUnslottedRX function. Thus, the very first thing that is done to the receive configuration is to turn on the auto acknowledgement bit and allow the link quality indication (LQI) word to be stored behind the packet in the ZMD44102's receive FIFO. Then, the ZMD44102's transmit success interrupt trigger is disabled.

The receiver gets activated and as soon as some valid data (good packet CRC) gets through to the receive FIFO. Control is returned to the cmdUnslottedRX function. The 5-ms wait is there to allow time for the link to turn around and the receiver MAC to post and transmit an acknowledgement message. I allowed 5 ms here, but it really only takes about 3 ms for this to happen. The zmdGetRxPacket function has already transferred the data from the receive FIFO to the data\_buffer array. So, all I have to do is parse through the data\_buffer array and do as I please. I know that the actual data payload is at offset 0x0A as there is a packet length

byte at offset 0x00 of the data\_buffer array. Just to make something physical happen, I dump the payload data byte into the receiving ZMD44102 starter board's LED I/O port. You can see the contents of the data\_buffer array after the reception of a packet from our transmitting node in Photo 5.

Take a look at the Sequence Number in Photo 6. It matches up with the sequence number in Photo 5. I twisted all of the necessary buttons and knobs (the acknowledge request bit set in the frame control word and the AutoAckEnable bit set in the receive configuration) to allow the ZMD44102's MAC to generate the 5-byte acknowledge message.

### HIT THAT EASY BUTTON

I've just shown you that you don't really need a ZigBee stack to get small messages from point A to point B. The ZMD wireless sensor starter kit bundle is an excellent way to get your feet wet if you want to dive into the ZigBee pool. And, you may find that ZigBee may be overkill for your WPAN application. No matter which

way you decide to go, the ZMD wireless sensor starter kit bundle takes the complication out of IEEE 802.15.4. It won't take you long to see that IEEE 802.15.4 isn't complicated. It's embedded. ☐

*Fred Eady (fred@edtp.com) has more than 20 years of experience as a systems engineer. He has worked with computers and communication systems large and small, simple and complex. His forte is embedded-systems design and communications.*

## SOURCES

**Sensor network analyzer**  
Daintree Networks  
www.daintree.net

**PK51 C Compiler**  
Keil  
www.keil.com

**C8051F120 Microcontroller**  
Silicon Laboratories, Inc.  
www.silabs.com

**ZMD Wireless sensor starter kit**  
ZMD America  
www.zmd.de

## "I NEED TO STAY AHEAD JUST TO KEEP UP"

### It's MY Show!

Scott McCurdy is the manager of design and business development for Jardon Engineering, a PCB design layout service bureau. Scott needs to stay ahead of technology changes in the industry just to keep up with his customers. He notes "the PC board industry is constantly evolving as packaging density and complexity gets more challenging every year. I depend on my visits to the shows and Designers Day to explore new technologies and tools to help my PCB designers do a better job for our end customers. If we don't participate in these types of events, we are doomed to fall behind in technology."

Scott McCurdy  
Manager of Design and Business Development  
Jardon Engineering, Inc.

Suppliers, services, OEMs, new products and technologies, new contacts and old friends. IPC Printed Circuits Expo®, APEX® and the Designers Summit bring together all segments of the industry with a full program of technical sessions, courses and standards development along with plentiful networking opportunities on and off the show floor.



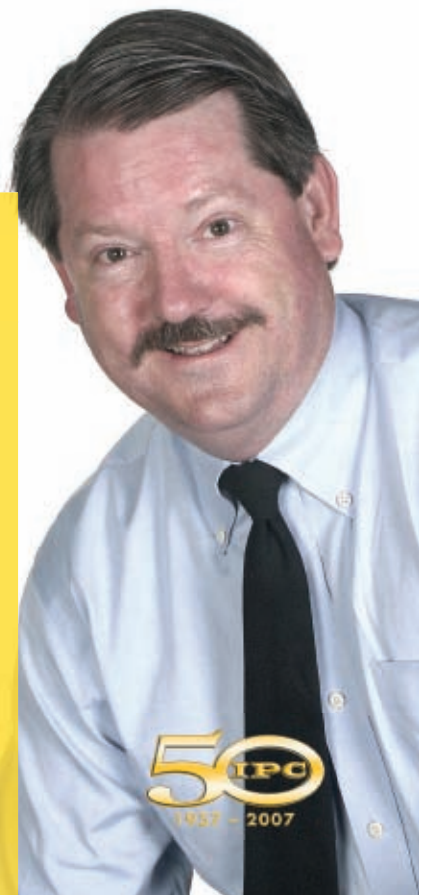
**Saturday – Thursday FEBRUARY 17 – 22, 2007**

Los Angeles Convention Center

**FOR MORE INFORMATION**

contact 877-472-4724 (U.S./Canada) or shows@ipc.org

**www.GoIPCShows.org**





# Green Energy

*What better time than now to get serious about the ways we approach our power and energy consumption needs? Jeff takes a look at “green” technology and describes a few alternative power sources.*

**R**adio has been with us for many years now. You’ve seen the move from AM to FM to satellite. Music might be the most widely broadcast content, but talk shows and news are easy to find anywhere on the dial. Although you might think of AM and FM as local, stations like the British Broadcasting Company (BBC), Voice of America (VOA), and National Public Radio (NPR) are listened to around the world.

As a teenager, I carried around a transistor (radio that is) so I could listen to the local AM rock and roll stations. Two local stations—WDRC and WPOP—had a fierce competition for listeners. Today, WPOP is gone and WDRC is a talk radio station, although WDRC FM competes in the mainstream music wave. Each night, I’d fall asleep as I listened to tunes via a single earpiece. (AM is monaural.) And so began my dependence on Eveready batteries. At the time, much of the money I earned by returning bottles and doing odd jobs went to buying either batteries or plastic models in support of my habits.

Radio broadcasts have the potential of spreading educational information to underdeveloped countries, so the radio can serve as a lifeline for those in need. However, most countries don’t have the distribution networks that many of you take for granted. Most

undeveloped countries don’t have local convenience stores; nor can many of those living in such places afford a luxury such as batteries.

How can we help these nations develop without creating additional consumers of the Earth’s precious and limited oil supplies? This is an opportunity for us to begin teaching the world’s future users about renewable energy.

British inventor Trevor Baylis was the first to take the windup mechanism mainly used in the clockwork and toy industries and apply it as an alternative power generator for a transistor radio. Realizing the potential, the Freeplay Energy Group was

formed in 1998 to take this idea to the next level. The group started with an initial grant from the British government and investments from big business. Since then, the group has remained at the forefront of this new self-sufficient electronics industry. Refer to the Freeplay Foundation sidebar for more information.

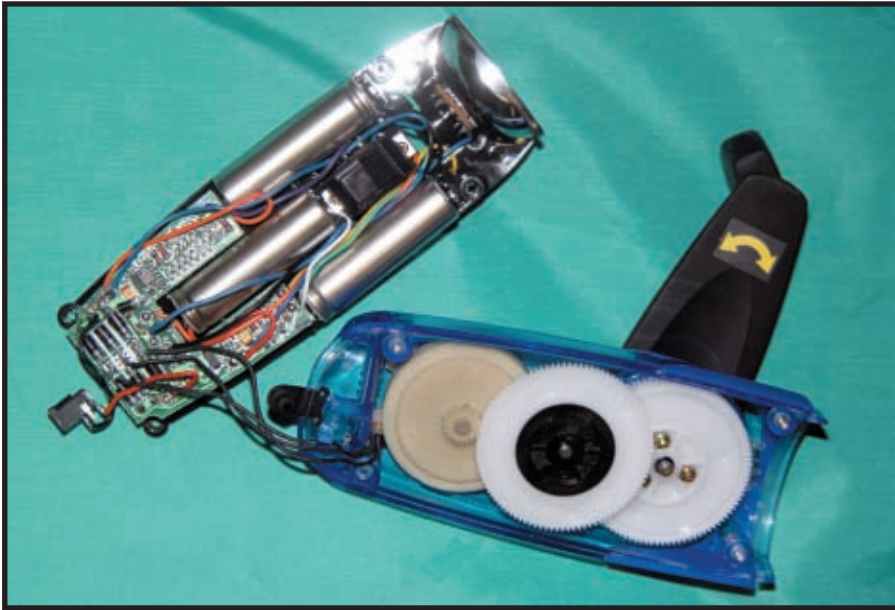
## SELF-GENERATED POWER

Today, Freeplay has numerous products in three areas of interest: radio, illumination, and energy production. The original products involved the winding of a steel spring, which in turn would unwind and drive a DC generator. Today’s products work by cranking an alternator. The alternator creates approximately 6 V of three-phase AC. This output supplies the charging of the three NiMH batteries.

I recently measured the current requirements and charging currents available to the Freeplay Ranger AM/FM radio (see Photo 1). Normal listening at a hefty audio level requires approximately 30 mA (4.8 V, three AAA cells). Based on an operating current of 30 mA at 4.2 V, the Ranger radio uses 126 mW. A good AAA NiMH cell will discharge in about 12 h at a rate of 30 mA. The solar cell on top of the Ranger is capable of supplying 30 mA on a clear day (if it’s oriented correctly). The alternator is capa-



**Photo 1**—Freeplay’s Ranger radio has multiple charging sources. Its AAA NiMH batteries can be charged from the AC adapter, the internal alternator’s cranking handle, or an internal solar cell. The operating time is about 5 to 50 times the cranking time (depending on crank RPM and playback volume).



**Photo 2**—Freeplay's LED flashlight enables you to keep its batteries charged with the standard internal alternator. Illumination has a high and low setting for its seven white LEDs to help conserve batteries.

ble of producing a pretty hefty current. Some minimum effort must be maintained to sustain sufficient output current. An indicator will illuminate when adequate output is produced.

I measured approximately 100 mA of charging current at this minimum indication. By cranking harder, I was able to produce up to 2 A of charging current. Of course, you must apply more effort to produce the higher current.

The actual charging rate depends on the amount of continuous energy that you can maintain. AAA NiMH batteries have a size C of approximately 600 mAhr. It is recommended that a NiMH cell be rapid charged at no more than 1 C. Specifications suggest that 40 min. of cranking would charge the batteries in 40 min. This may severely overtax the cells. (When rapidly charging NiMH batteries, it's suggested that you monitor their temperature so you can to prevent overheating and electrolyte leakage.)

### BTU, HORSEPOWER, & kW

The British thermal unit (BTU) is widely used in the United States in discussions about heating and cooling. It is defined as the amount of heat (energy) required to raise the temperature of 1 lb. of water by 1°F. There are a few ways to look at the energy of 1

BTU. For instance, 1 BTU is equal to 0.00039 hp and 0.00029 kWh. These same units of energy can be compared as consumables when referenced to 1 h.

Consumers who pay for electricity are familiar with kilowatts and kilowatt hours. My last bill showed I used 1,054 kWh of electricity during a 30-day month. I used an average of 35 kWh per day (1,054 kWh per 30 days), or 1.5 kW per hour. At that rate, the power company charged (for generation) me approximately \$0.10 per kWh. (Note that the delivery charge for this energy was approximately \$0.06/kWh.) My last fuel oil delivery was \$2.39 per gallon.

Fuel oil is about 151,000 BTU per gallon. That gallon of fuel oil converts to 44 kWh (151,000 × 0.00029). At \$0.10 per kWh, \$2.39-per-gallon oil is worth \$4.40 (44 kWh × \$0.10) to the power company for electricity generation. Of course, it doesn't pay \$2.38 for a gallon of oil and conversion efficiency isn't 100%, but it does serve as sanity check.

The price of gasoline happens to be down as I write this column. Although it was higher than \$3 for most of the summer, I paid \$2.80 for a gallon last week. The price of a gallon of gas is important to drivers, but it really boils down to miles per

gallon (as all of the owners of large SUVs know). The last fill-up for my van was 17.5 gallons. If I drive 370 miles, that's 21 MPG (370 miles per 17.5 gallons). At \$2.80 per gallon, that's about \$0.13 per mile (\$2.80 per 21 MPG).

If a gallon of gasoline is equal to 125,000 BTUs, then it's 36 kWh (125,000 BTU × 0.00029 kWh/BTU). However, the efficiency of gasoline engines is only about 30%. So, we end up with about one-third of the available energy, or 12 kWh. Using this value, the cost is \$0.24 per kilowatt-hour (\$2.80 per 12 kWh).

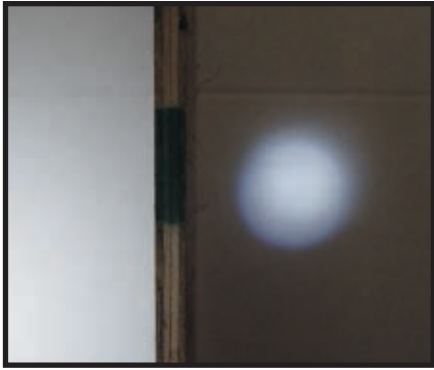
Better yet, my van requires an energy value of 571 watt-hours per mile (WhPM). Note that 12 kWh per 21 miles is 571 watt-hours per mile. Using the watt-hours per mile measurement (or even MPkWh) makes it easier to compare gasoline vehicles to electric vehicle alternatives or energy usage in general.

### EFFICIENCY

Talking about energy conversion is all well and good, but so much depends on the efficiency of energy conversion. A 30% conversion efficiency for gasoline vehicles is pretty darn awful. Thankfully, you have much higher efficiencies when dealing with moving electrons. Some charging systems can get better than 90% efficiency. Although linear supplies can be designed to reduce losses to a minimum, efficiencies are generally much lower than when using digital techniques.



**Photo 3**—When shaken, an internal magnet slides back and forth through the coil. The coil produces voltage pulses as the magnetic field cuts through the coil's winding. This voltage pumps up a supercapacitor, which is the power source for a single LED.



**Photo 4**—You can clearly see the difference between the Freeplay flashlight’s illumination (left) versus the toy shake-to-charge flashlight (right).

Let’s take a look at the alternator used in most of Freeplay’s charging systems. You must measure the energy input (manual labor) and energy output (charging current) to get handle on the kind of efficiencies that can be expected.

The setup to record energy input is a simple lazy Susan (a ball bearing platform) mounted to a plywood base. The item to be monitored is attached so that the crank’s centerline coincides with the centerline of the lazy Susan. The free rotating top has an arm attached to it. I used an adjustable bookshelf bracket as the arm because it has holes spaced 0.5” and is labeled every 1”. This enables a scale to be mounted at various points on the arm. The scale measures the force on the arm as the device being tested is operated (e.g., when the handle is cranked).

To measure energy output, the voltage and current are monitored at the NiMH batteries as the alternator is cranked. There is an initial resistance to alternator rotation where the magnetics are aligned. After breaking this fundamental bond, the cranking force drops and remains minimal, requiring about 35 inch-ounces before any current is measured.

As you can see in Figure 1a, the output current is fairly linear with the force applied to the crank handle. The voltage potential applied rises along with the current output. Using the graph, you can calculate the

energy being applied to the NiMH batteries at various output levels. At a force of 40 inch-ounces (or 0.2 ft-lbs), the energy output is 150 mA × 4.4 V or 660 mW. Also, at 40 inch-ounces of force, I had to maintain about 80 RPMs on the crank (see Figure 1b).

By knowing the force exerted and the crank’s RPMs, you can calculate the power input:

$$\text{Horsepower} = \frac{33,013 \text{ ft-lb}}{\text{minute}}$$

and

$$\text{Horsepower} \left( \frac{\text{feet-lbs}}{\text{minute}} \right)$$

$$= \text{torque (lb/feet)} \times \frac{\text{linear distance}}{\text{minute}}$$

Thus,

$$\text{Horsepower} = \frac{0.2 \times \frac{\text{linear distance}}{\text{minute}}}{33,013 \text{ feet-lb}} \frac{\text{minute}}{\text{minute}}$$

or

$$\frac{0.2 \times \text{linear distance}}{33,013}$$

You can substitute:

$$2\pi \times 1 \text{ Radius} \times 80 \text{ RPMs for linear distance}$$

The result is:

$$0.2 \times 2\pi \times \frac{80}{33,013} = 0.003 \text{ horsepower}$$

or

$$0.003 \times 0.735 = 0.0022 \text{ kW (2.2 W)}$$

The input energy expended to produce a NiMH charging rate of 150 mA (660 mW output energy) is 2.2 W. The system’s efficiency is output energy

per input energy. In this case, it’s 660/2,200, or 0.3 (30%).

## CRANK IT UP

Although the alternator used in Freeplay devices has a current output that’s much greater than the optimum charging current, it doesn’t seem to have any limiting protection for the NiMHs. It looks like a good cranker could heat the rechargeables up to a point at which they may be damaged. I say “a good cranker” because cranking for an extended period of time can really fatigue those muscles in your forearms. At 600 mA, it would require about 1 h of continuous cranking to totally charge the NiMHs back to a fully charged state. That isn’t something I’d be interested in doing.

Another Freeplay product is the Sherpa X-Ray LED flashlight (see Photo 2). This product uses the same alternator arrangement, but the storage device is three AA batteries. AA NiMH batteries have approximately double the capacity of AAA cells. The light source is seven white LEDs. The power switch has two settings: high and low. On the high setting, all of the LEDs are driven with about 22 mA of current (a total of 154 mA) with each LED having a 12-Ω series resistor. On the low setting, the potential across the load (LED/resistor sets) drops down by about a 1 V, thereby reducing each LED’s current to about 5 mA (a total of 35 mA). Because the currents in the flashlight are two to 10 times the Ranger’s current, you can expect the working life of a charge to be much shorter than that of the radio and take a good deal more cranking to get fully recharged.

## TOO GOOD

Usually, if it sounds too good to be true, it probably isn’t true. You may have seen another type of “green” flashlight on the market. The “shake-to-charge” flashlight uses the same principle as the alternator except it uses a single coil and a sliding magnet. You throw the magnet through the coil by shaking it. The current produced from the mag-



**Photo 5**—On average, each American is responsible for 4 lb. of garbage each day.

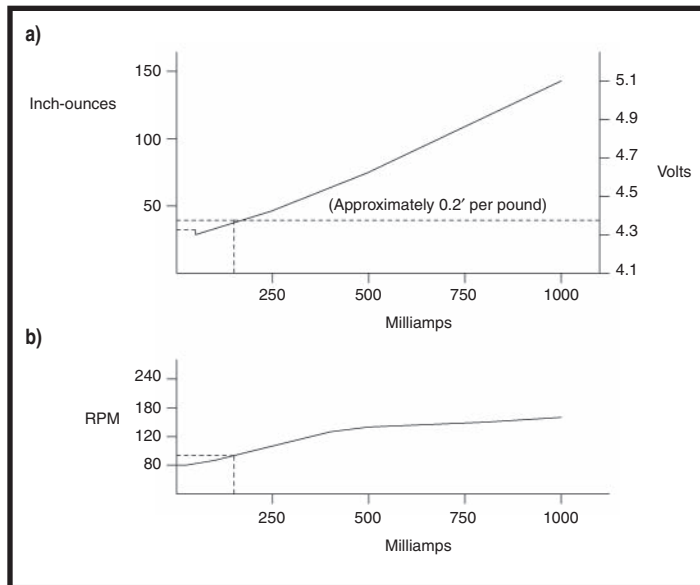
netic field passing through the coil is applied to a super capacitor (0.33 F) instead of a rechargeable battery (see Photo 3). This device places a single white LED across the capacitor when powered via the push button.

After an initial charge/discharge cycle, I shook the flashlight. The LED output went from good to wimpy in about 1 min. Calculating the charge held in a 0.33-F capacitor, I found that I could expect the initial measured current of 20 mA to discharge the capacitor from 5.5 to 2.5 V in about 30 s. This calculation ( $dt = C \times dv/I$ ) might be correct for a constant current; however, the current drops along with the potential, so the time is extended to point where the light output is unusable.

You won't change a tire on a dark road using one of these. Photo 4 shows a white background illuminated by the crank charge and the shake charge flashlights in a dark room. I placed a center divider between the two to improve the contrast.

## GO GREEN

The alternatives available for pow-



**Figure 1**—Take a look at the effect on the alternator's charging current used in the Freeplay Ranger radio as the force on the cranking handle changes (a). Voltage and current measurements are used to calculate output power. The bottom graph (b) shows the same charging currents versus the cranking handle's RPM. The force and RPM values are used to calculate input power.

ering devices in emergency situations or in locations where no sources of power exist are limited. Don't expect too much. Even good designs can require a good deal of physical energy to become worthwhile. This project really enabled me to appreciate how much energy is in a set of batteries (and how I take for granted the convenience of purchasing batteries at any store).

You can basically save energy in two ways. You can improve the efficiency of how you transfer energy

(e.g., recharging the state of a device). You can also improve the efficiency of how the device uses the energy. Such improvements are happening today. For instance, you can replace incandescent bulbs with energy-saving fluorescent bulbs. You can also use switching supplies instead of linear ones to raise the efficiency of power conversion.

Freeplay Energy has a jump on the rest of the market. It may not be using the most efficient technology, but it's being used today to help our world. "Green" is a term that will not go away anytime soon. How much of the 210 million tons of waste per year

has your name on it (see Photo 5)? ☒

*Jeff Bachiochi (pronounced BAH-key-AH-key) has been writing for Circuit Cellar since 1988. His background includes product design and manufacturing. He may be reached at [jeff.bachiochi@circuitcellar.com](mailto:jeff.bachiochi@circuitcellar.com).*

## SOURCES

### Ranger AM/FM Radio

Freeplay Energy  
[www.freeplayenergy.com](http://www.freeplayenergy.com)

## Freeplay Foundation

As a natural extension of the group's work, Freeplay Energy founded the Freeplay Foundation in 1998. The foundation was created to facilitate sustained access to information and education for the poorest of the poor, especially children, women, refugees, and the disabled. Its purpose is to raise awareness of the role of radio broadcasting and communication in developing countries, disaster areas, and regions of conflict. It also researches opportunities where appropriate and alternative sources of energy can be applied to improve the lives of people in developing communities, especially children living on their own.

The foundation provides new, practical energy solutions and is dedicated to ensuring sustained access to information and education through radio broadcasts.

Its self-powered radios provide sustainable access in several key areas. The foundation is committed to disseminating educational resources and help (e.g., long-distance learning programs to increase literacy and knowledge of the environment). It is also committed to promoting useful healthcare-related information (e.g., information about hygiene, immunization, first aid, family planning, and disease prevention and care). It provides assistance during emergencies such as evacuation information.

The foundation is also committed to peacemaking. It promotes conflict resolution, reconciliation, healing, and cross-cultural understanding. Finally, it is dedicated to providing information about topics ranging from water conservation to farming techniques.



# Hello World ... Want Cookie

*George is on a mission to walk you through the process of creating an embedded system, writing code in C language, and then testing the design. In this article, he takes a fresh look at C language and points you in the right direction.*

“Hello World.” That’s traditionally the first message you send out the system output port when you bring up a new C project. In keeping with my *Sesame Street* days, I usually change that first message to “Want Cookie.” Since we all know that embedded systems are a bit out of the ordinary, I want to make a statement that this new system is special.

I’d like to present a series of articles that will walk you through the process of creating an embedded system, writing the code in C, and bringing up that design using one of several evaluation boards available. I’ll be using low-cost evaluation boards for 8-, 16-, and 32-bit microprocessors. These evaluation boards come with an integrated development environment (IDE), editor, C compiler, assembler, linker, locator, downloader, and debugger. The typical price for an evaluation board and tools is \$100 or less. Some of these tools are limited in capabilities, but if you run into that limitation then I’ve accomplished my task and you’ll be ready for the next step. This won’t be a cookbook approach to creating C projects, but I hope instead to lay out a plan for you to follow for any project simple or complex.

I’ve noticed that several designers I talk with don’t use C. These are experienced designers who turn out great systems. When I ask why not C, the main reason stated is that they just never had the time in their busy schedules to sit down and learn how it all works. Also, many hardware engineers understand the hardware, the logic, and even assembly language, but

they also never had the formal software training (exposure) to feel comfortable with a high-level language. Well, now’s your chance. I promise this will be painless, right up until when we start capturing user requirements. But isn’t that part of our real job description anyway? I also think that the student readership will get some benefit out of these articles because they will probably parallel some class work in their embedded system workshops.

I’m going to use several different evaluation systems. These systems will range from very tiny 8-bit devices to full-blown 32-bit powerhouses. I don’t want to concentrate on any one system in particular. After I lay a foundation in what to do, I’ll keep moving among these systems to show you what happens and what is possible with each. That way you can see the differences and feel comfortable selecting the best system for your particular design situation.

One of the benefits of using C is to divorce the code writing from the underlying hardware. This lets you, as designer, move among the processor offerings more freely. Each microcontroller has its significant features, and you should be able to identify what they are and how valuable those features would be in completing your design.

## GET STARTED WITH C

In the early 1970s, Dennis Ritchie developed C language for the Unix operating system. Since then, it has spread to other operating systems, and today it’s one of the most widely used languages. Over the years, C has influenced the

development of various other languages (e.g., C++). It is now the most commonly used language for writing software.<sup>[1]</sup>

C language is well documented, so I won’t waste ink with a detailed language lesson. We’ll pick up the language as we design and build our systems.

Brian Kernighan and Dennis Ritchie’s book, *The C Programming Language*, is the first reference you should review in order to familiarize yourself with the language. This is not perhaps the best book, but it’s certainly a good reference and the one you can use as a basis for our endeavor. I’ll reference it as we explore the language.

As I was gathering information about development systems, I was speaking with Tim Shannon who is in technical sales at NetBurner. He provided me with a list of books that he said would be helpful to individuals starting with C. He especially recommended Greg Perry’s *Absolute Beginner’s Guide to C*. More about NetBurner in later articles. Also, be sure to consider used books about the C programming language.

## EMBEDDED SYSTEM

First, let’s describe an embedded system. It’s not a PC, but we’ll use a PC to do most of the work. An embedded system might be described as a stand-alone computer with a power supply, inputs, and outputs. Once it’s powered up, an embedded system executes its program continually. A calculator is a good example of an embedded system.

But another characteristic of our embedded system is that it is of a real-

time nature. For this set of articles, “real time” can mean blinking a light at approximately a 1-s interval or anticipating close of work on Fridays and powering down unnecessary equipment. It doesn’t have to be lightning fast. Just as I wouldn’t characterize a PC as an embedded system, I would, however, put iPods, PlayStations, and Xbox systems in the embedded category. Heck, even a PC, when it’s running flight simulator, is an example of a real-time embedded system.

So, let’s define a real-time embedded system as: a CPU with an input (or inputs), an output (or outputs), a power supply, and some real-time requirements. The real-time requirement will keep us on our toes and not get sloppy with CPU resources and bloated code.

## FIRST DESIGN

For our first design, let’s start with a switch for input and an LED for output. Our first design is one that turns the LED on whenever the switch is pressed and off when the switch is released. There should be no humanly perceptible delay between switch action and LED operation. Simple enough, here goes.

In order to develop the system I’ve just described, we would need to install the development tools, create a project in the environment, write the code that represents our design, and then compile, download, and test that code. Let’s just skip to the fun part and zero in on the code requirements.

Today’s tools are very sophisticated, and we’ll get into how to set them up in a later article. But for now, take my word that the tools will have created a project and in that project directory will be a file called `main.c`. All C source code files have the extension “`.c`.” In `main.c`, you will also find a procedure named `main`, written as `void main(void)`, or

something similar. `main` is the procedure that starts the C code execution. The CPU is started at power-up. Several assembly language routines are executed to configure the system and create the environment needed for your C program to execute. More about all of these later.

## C OVERVIEW

A little about C so we can start to implement our design. C is a compiled language as opposed to BASIC, which is an interpreted language. A compiled language takes source files and then compiles, assembles, links, and locates them to produce an object file. This object file is loaded for execution or debugging. So, we will be continually editing, compiling, and debugging our code as we go through the development process.

C is a procedural language used to write operating systems. It can produce code as efficient as assembly language. As a requirement for that efficiency, we need to inform the compiler all about what we are intending to do. Let’s start with variables. C assigns a type to each variable. An 8-bit variable is a `char`. It can be used to hold numeric values from `-128` to `127` using two’s complement notation. Another variable type is `unsigned char`. And that can hold numeric values from `0` to `255`. In assembly language, we never distinguished between these types, and thus we have to keep track of what we were doing and be responsible for the results. Pop quiz, with unsigned subtraction, What do the CPU’s carry and borrow flags represent? C takes care of this, and we’ll see just how powerful this becomes as we get more complex design issues.

The next larger type is a 16-bit entity, and let’s just say that’s an `int`. I’m tentative here because the NetBurner uses a CPU that is a 32-bit design. So

`ints` for that system will probably be 32 bits. Don’t worry. I’ll show you an easy way out of this pickle. Along with `ints`, we have `unsigned ints`. Just as `ints` would let you represent `-32768` to `32767`, `unsigned ints` let’s you represent numbers from `0` to `65536`. There are more types and we’ll get to them later.

Statements in C are separated by a semicolon. You can place several statements on a single line, but that can become less clear. For now, keep to one statement per line. So, `int a;` saves space for the variable `a`. How much space? Well, 16 bits worth. Just as `char b;` reserves 8 bits for the variable `b`. And the following code defines `a`, `b`, and `c`, sets values into `a` and `b`, and then adds them and places the result into `c` (see Listing 1).

Just where are these variables stored? In RAM—unless you tell the compiler to place them elsewhere. The CPUs we’re using and C rely heavily on the stack. Each procedure uses space on the stack for locally defined variables. These variables are created when the procedure is called and destroyed when the procedure is terminated. You can define a variable to be in RAM permanently (static) and available to all procedures or in the procedure’s stack and lost after a procedure is completed.

Take a look at Listing 2. Lines 1 and 2 define the 16-bit variables `a` and `b`. Standard C has comments in `/*...*/`. But here is one place I suggest you deviate for the standard. The `//` pair can be used to mark the remainder of the line as comments. I find this creates C code that is much cleaner to read and understand. Everything on the line including and following the `//` mark is a comment. Line 3 is a blank line for readability of the code. Line 4 is the start of a procedure. This procedure takes the variables `a` and `b`, adds them together, and saves that result in the variable `c`. The variable `c` is defined on line 5 and located on the stack for `procedure1`. Lines 5 through 8 are indented for readability. The result of the addition is returned on line 8. Line 9 ends the definition of the procedure. The curly braces (lines 4 and 9) encapsulate the procedure and are required. The `int` and `void` on line 4 were not originally required in C, but you better use them or get an F in this class. It’s good to be

**Listing 1**—This is a simple example of C code (declaring variables and using them).

```
int a;      (1)
int b;      (2)
int c;      (3)
           (4)
a = 7;      (5)
b = 3;      (6)
c = a+b;    (7)
```

proactive and define everything. The compiler can use all of the information you can give it to help diagnose your problems.

Procedure1 knows about the existence of variables a and b because they were defined in the same code module and defined before Procedure1 was defined. This is referred to as scope of the variable. If we had another module and wanted to use the variables a and b, we would need to inform that other module about their existence.

```
extern int a; (1)
extern int b; (2)
```

Extern is a C keyword that does just that. Variables are not global. You as the designer need to establish which modules will have knowledge about which variables. At first, this seems a drag, but as your designs get more complicated, you'll be thankful.

Procedure1 adds two numbers and these numbers need to be in variables a and b. What if you wanted to add just any two variables? Look at procedure2: (see Listing 3). Line 1 defines procedure2 as a procedure that is passed two int variables, sums them, and returns the result as an int. Much cleaner and more understandable. Notice we didn't use the intermediate variable c.

Now we could use this new procedure2. Line 4 defines a variable representing the number of boys in the class. Line 5 defines a variable representing the number of girls in the class. And Line 6 defines a variable representing the number of students in the class. We next assign values into the variables (lines 7 and 8) and then call procedure2 (line 9). We pass procedure2 the variables to be summed and the procedure returns the result. All of this compiles without error because it's all in the same module, and line 9 knows about the procedure because it was defined in line 1.

A word about variable names. Uppercase and lowercase are acceptable. I use capitalization to make variable names easier to read. You can't start a variable or procedure with a digit. I refrain from using special characters such as \$%^&. Some of these are perhaps acceptable, but why take a chance?

**Listing 2**—This is an example of a procedure that sums two variables and returns the result.

```
int a; // define a (1)
int b; // define b (2)
(3)
int procedure1(void) { (4)
    int c; (5)
(6)
    c = a + b; // create the sum (7)
    return(c); // pass is back to the caller (8)
}; (9)
```

And you should check the specifications of your compiler just to see how compatible it is with the C language.

Now is also a good time to develop some style with variable names. The names used in the above example are very readable and give an indication as to what's going on. Another style goes something like this: NumberOfGirlsInClass. The type qualifier is either added as a prefix or suffix. This can be very useful in really large projects with many designers. One designer is responsible for creating the variable and others know immediately all about that variable.

A word of caution. It's common usage in C to use variables with all capital letters as constants. Here is a simple define:

```
#define MY_FIRST_CONSTANT 3 (1)
int UseAConstant; (2)
```

```
UseAConstant = MY_FIRST_CONSTANT (3)
```

Line 1 defines a constant equal to three. Line 2 defines a variable, and line 3 sets that variable into the constant. This is a great way to keep from hard coding constants into your code. The customer says I want you to blink the LED three times for an error. I would code the value 3 as a define so that it's not entangled in the code because customers and designers change their minds.

```
#define ERROR_BLINK_COUNT 10
```

It's now visible and easily changed.

## SYSTEM I/O

Let's look at input and output for our embedded system. On the type of machines we are going to cover, CPU

**Listing 3**—A perhaps better version of Listing 2. This procedure takes two parameters, performs a simple addition operation, and returns the result.

```
int procedure2(int a, int b) { (1)
    return(a+b); (2)
}; (3)

int NumberOfBoysInClass; (4)
int NumberOfGirlsInClass; (5)
int NumberOfStudents; (6)

NumberOfBoysInClass = 20; (7)
NumberOfGirlsInClass = 25; (8)

NumberOfStudents = procedure2(NumberOfBoysInClass,
    NumberOfGirlsInClass); (9)
```

**Listing 4**—This listing shows the use of the C #define statement and how to use it to define memory-mapped I/O.

```
#define PORT0_DIR 0x0190 // sets direction port0 (1)
#define PORT0_DATA 0x0192 // reads and sets port0 (2)
#define PORT1_DIR 0x0193 // sets direction port1 (3)
#define PORT1_DATA 0x0194 // reads and sets port1 (4)
```



**Listing 5**—This code fragment declares a variable as a pointer to an `int`, sets its value, and then uses that pointer to store a value.

```
int *x;    // defines x as a pointer to an int    (1)
x = 100;   // Stores the value 100 into x        (2)
*x = 7;    // Stores the value 7 into the       (3)
           // memory location contained in x (100) (4)
```

pins can typically be either inputs or outputs. The exact usage is defined by the setting of a bit in a pin direction register. Input is performed by reading the voltage level of the pin, while output is performed by writing the output bit either 1 or 0. Some popular processors such as the 80XXX family have a dedicated I/O instruction. Setting those aside for now, we'll be dealing with CPUs that have memory-mapped I/O.

The first thing we need to define is the address of the port direction register (see Listing 4). These are hypothetical values and not meant to be any particular CPU. The 0x0190 is hexadecimal notation (0x0190 = 400 decimal). Let's

say that we need to set the direction bit to a 1 for output and a 0 for input. (Your actual processor may vary.) Let's also say that the data register is positive true logic, and that the hardware designers put the switch input on bit 0 of Port0 and the LED output on port1 bit 1.

This is a good place for a pointer. I'll talk about pointers in more detail in a later installment. This is another C type. It is used to store a value used to reference memory. In memory location 100, I want to store the `int` 7 (see Listing 5). That's pointers and how they work. We'll expand on pointers in later articles. Pointers are great tools for speeding up code execution.

**Listing 6**—Using the memory-mapped I/O definitions in Listing 4, this listing shows a procedure that sets up the CPU pins as inputs and outputs. It then reads the input and based on the result performs different output operations.

```
Void OperateLED(void) { (1)
Char *p;                // define a pointer to an 8 bit entity (2)
p = P0_DIR;             // Set the pointer the memory          (3)
                        // loc of the Port0 control register (4)
*p = 0x00;              // set the direction as input        (5)
p = P1_DIR;             // Set the pointer the memory          (6)
                        // loc of the Port1 control register (7)
*p = 0x02;              // set the direction as input        (8)
p = P0_DATA;            // Now Set the pointer to memory      (9)
                        // location for the data register (10)
                        // (11)
if (*p & 0x01) { // Read and test the input (12)
    p = P1_DATA; // Now Set the pointer to memory (13)
                // location for the data register (14)
    *p = 0x02;  // If true set the output (15)
} (16)
else { (17)
    p = P1_DATA; // Now Set the pointer to memory (18)
                // location for the data register (19)
    *p = 0x00;  // Reset the output (20)
} (21)
}; // end of Void OperateLED(void) { (22)
                                     (22)
void main (void) { (24)
for(;;) { (25)
    OperateLED(); (26)
} (27)
} // end of void main (void) { (28)
                                     (29)
```

+++ NO ROYALTIES +++

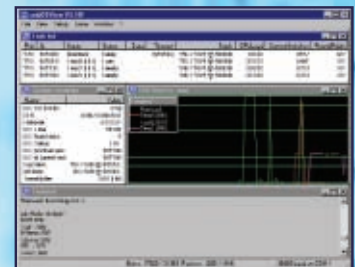
embedded software  
solutions

Eval versions  
available

**embOS**® (RTOS)

+++ 8/16/32 bits +++

Preemptive multitasking  
Zero interrupt latency  
Easy to use start project included  
Profiling support included  
Object/source code available



**emWin**® (GUI)

+++ 8/16/32 bits +++

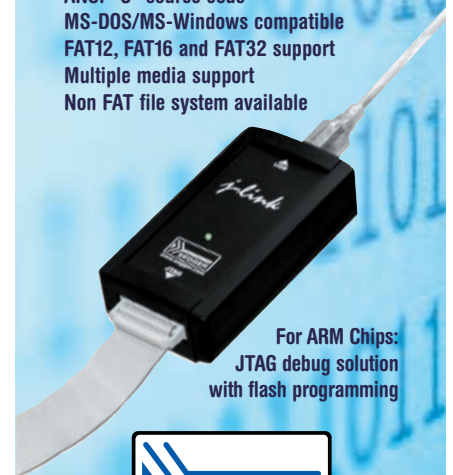
ANSI "C" source code, no C++ required  
Supports b/w, grayscale and color  
2D graphic library included  
Variety of fonts included  
PC simulation included  
Window Manager/Widgets (opt)



**emFile** (File system)

+++ 8/16/32 bits +++

ANSI "C" source code  
MS-DOS/MS-Windows compatible  
FAT12, FAT16 and FAT32 support  
Multiple media support  
Non FAT file system available



For ARM Chips:  
JTAG debug solution  
with flash programming



phone: 978-874-0299  
www.segger.com

Refer to Listing 6 for a complete design to meet the design requirements. This should compile, link, locate, and be ready to debug. Lines 1 through 22 define the OperateLED procedure. Lines 24 through 28 define the main procedure. Line 25 is a new C keyword. The for keyword is used to define a looping operation. Look up the definition of a for loop for the exact details, but take my word: this for(;;) will loop forever.

Look at line 12. It reads if (\*p & 0x01) {, which is the construct for an if-then-else statement. If the condition on line 12 is true, statements 13 through 15 will be executed. If the condition is false, then the statements 18 through 20 will be executed. I always include the {} braces. The C language will let you leave them out if there is only one statement to execute, but saving those keystrokes will cause you grief. Don't do it.

Exactly what is true and false should be your next question. False is defined as zero. True is defined and not false or nonzero. The & operator is the bitwise logical AND operation. Look up & (AND), | (OR), ~ (NOT), ^

(XOR) in your language references.

So, line 12 reads as follows. If the Port0 input pin bit 0 is a one (true), then execute lines 13 through 15. If it's a zero (false), then execute lines 18 through 20.

## THE REAL WORLD

Now for some real-world issues. What if the operation were inverted? That is, what if the LED was on when the switch wasn't pressed? What would you change? What if the hardware engineers really put the switch input on bit 1 of Port0 and the LED output on bit 0 of Port1. Oops, they're very sorry and all went home for the weekend!!! What if this code was too slow? How could you speed it up?

I'll be happy to discuss any questions on the *Circuit Cellar* bulletin board (<http://bbs.circuitcellar.com/phpBB2/>). ☐

*George Martin (gmartin@circuitcellar.com) began his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and co-founded a design and manufacturing firm (www.embedded-designer.com).*

*George's designs typically include servo-motion control, graphical input and output, data acquisition, and remote control systems. George is a charter member of the Ciarcia Design Works Team. He's currently working on a mobile communications system that announces highway information. George is a nationally ranked revolver shooter.*

## REFERENCE

- [1] Wikipedia, "What is C?" [http://en.wikipedia.org/wiki/C\\_programming\\_language](http://en.wikipedia.org/wiki/C_programming_language).

## RESOURCES

- M. Barr, *Programming Embedded Systems in C and C++*, O'Reilly, 1999.
- B. Kernighan and D. Ritchie, *The C Programming Language*, Prentice Hall, 1988.
- A. Koenig, *C Traps and Pitfalls*, AT&T Bell Laboratories, 1989.
- G. Perry, *The Absolute Beginner's Guide to C*, Sams Publications, 1994.

## Easy Embedded Linux

\$169

Qty 1



16MB FLASH / 32MB RAM

200Mhz Arm9 CPU

16 Digital I/O

Watchdog

10/100 Ethernet

Battery backed Clock/Calendar

Audio In/Out  
2 USB  
2 Serial Ports

*We brought you the world's easiest to use DOS controllers and now we've done it again with Linux. The **OmniFlash** controller comes preloaded with Linux and our development kit includes all the tools you need to get your project up and running fast.*

*Out-of-the-box kernel support for USB mass storage and 802.11b wireless, along with a fully integrated Clock/Calendar puts the **OmniFlash** ahead of the competition.*

Call **530-297-6073** Email [sales@jkmicro.com](mailto:sales@jkmicro.com)  
On the web at [www.jkmicro.com](http://www.jkmicro.com)

# JK microsystems

# APEC® 2007

February 25–March 1, 2007





Disneyland, Anaheim, CA

THE PREMIER  
GLOBAL EVENT IN  
POWER ELECTRONICS™

Visit the Apec 2007 web site  
for the latest information!

www.apec-conf.com

SPONSORED BY

# USBee DX

## Oscilloscope/Logic Analyzer

2 Analog and 16 Digital Signals  
Up to 24Mps, 100+ Million Samples  
Dual 8-bit ADC, +/-10V inputs

## Bus Decoding

Click and Drag Instant Decode of Bus Transactions  
I2C, SPI, ASYNC, CAN, USB Low and Full Speed  
I2S, SM Bus, 1-Wire, PS/2

## Digital Voltmeter

2-Channel, +/-10V, 8-bit ADC, Logging

## Data Logger

2 Analog and 16 Digital Signals Plus Timestamp

## Digital Signal Generator

16 Digital Outputs, Up to 24Mps  
Playback of Logic Analyzer Traces

## Pulse Width Modulator

16 User Controlled PWM Channels

## Frequency Counter

16 Channel Counter to 24MHz

## Frequency Generator

Generates Sets of Common Frequencies

## I2C Controller

Control I2C Devices Using Simple Text Scripts

## Remote Controller

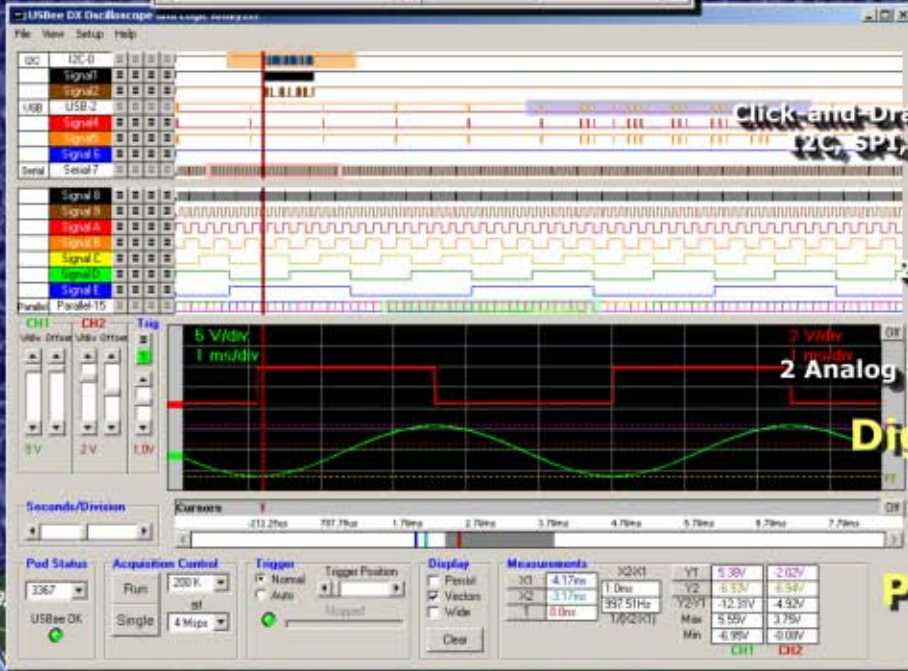
Easily Control Your Hardware Using Your PC

## Pulse Counter

16 Channel Pulse Counter With Gating Control

## plus the USBee Toolbuilder Source code and Library

Create Your Own Applications to Control the USBee DX



Actual Size

## Data Extractors

Optional Software Modules for the USBee AX-Pro and USBee DX

Continuous Real-Time Embedded Bus Data Streaming

Store Bus Data to Disk or Send to Your Custom Application

Capture and Process Entire Test Sequences

Parallel, Serial, I2C, USB, ASYNC, CAN, SMBus, SPI, I2S, 1-Wire

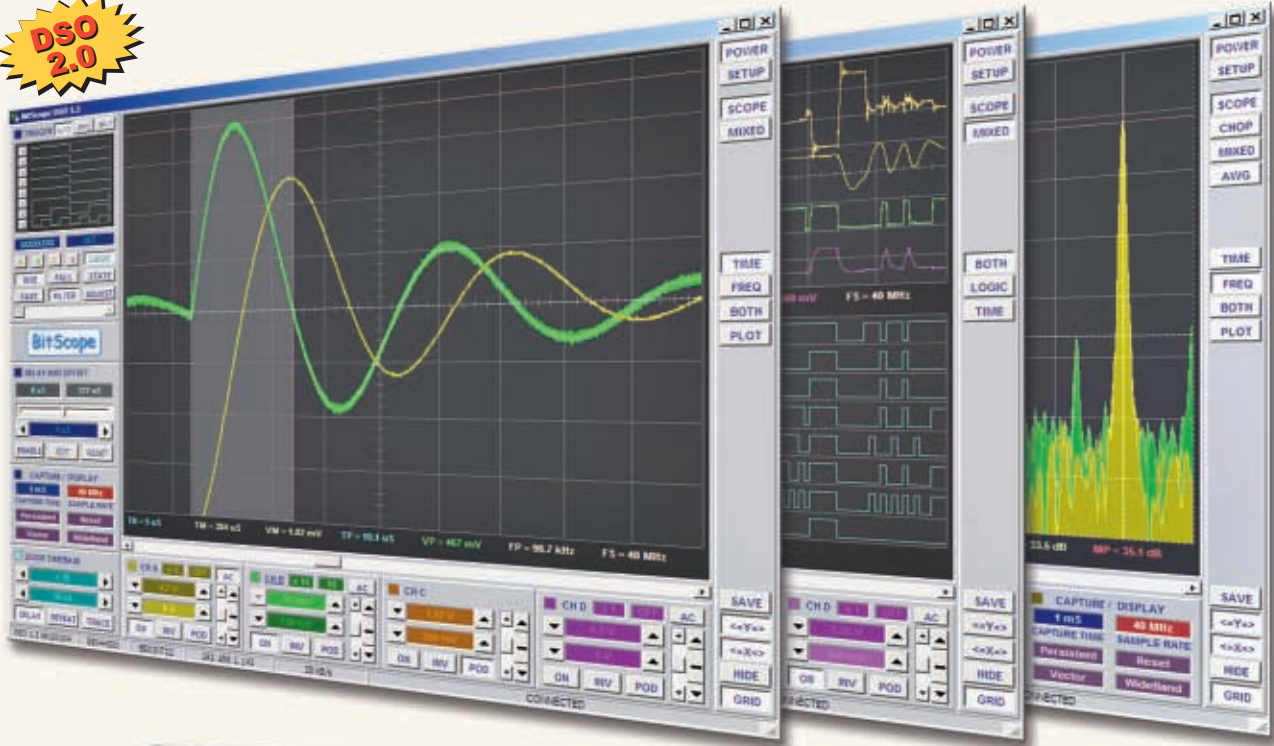
**USBee.com** (951) 693-3065  
support@usbee.com



# BitScope PC Oscilloscopes & Analyzers

## DSO Test Instrument Software for BitScope Mixed Signal Oscilloscopes

**DSO 2.0**



4 Channel BitScope



2 Channel BitScope



Pocket Analyzer

### Digital Storage Oscilloscope

- ✓ Up to 4 analog channels using industry standard probes or POD connected analog inputs.

### Mixed Signal Oscilloscope

- ✓ Capture and display up to 4 analog and 8 logic channels with sophisticated cross-triggers.

### Spectrum Analyzer

- ✓ Integrated real-time spectrum analyzer for each analog channel with concurrent waveform display.

### Logic Analyzer

- ✓ 8 logic, External Trigger and special purpose inputs to capture digital signals down to 25nS.

### Data Recorder

- ✓ Record anything DSO can capture. Supports live data replay and display export.

### Networking

- ✓ Flexible network connectivity supporting multi-scope operation, remote monitoring and data acquisition.

### Data Export

- ✓ Export data with DSO using portable CSV files or use libraries to build custom BitScope solutions.

## BitScope DSO Software for Windows and Linux

BitScope DSO is fast and intuitive multi-channel test and measurement software for your PC or notebook. Whether it's a digital scope, spectrum analyzer, mixed signal scope, logic analyzer, waveform generator or data recorder, BitScope DSO supports them all.

Capture deep buffer one-shots or display waveforms live just like an analog scope. Comprehensive test instrument integration means you can view the same data in different ways simultaneously at the click of a button.

DSO may even be used stand-alone to share data with colleagues, students or customers. Waveforms may be exported as portable image files or live captures replayed on other PCs as if a BitScope was locally connected.

BitScope DSO supports all current BitScope models, auto-configures when it connects and can manage multiple BitScopes concurrently. No manual setup is normally required. Data export is available for use with third party software tools and BitScope's networked data acquisition capabilities are fully supported.



[www.bitscope.com](http://www.bitscope.com)

# The Power of Flash

## Flash Memory Techniques for Your Toolbox

*You use flash memory to store code, but did you know that there are other uses for it? Mark describes several other ways to use flash memory in your embedded designs.*

**F**lash memory has become a pervasive element of embedded systems everywhere, so it's a safe bet that you are accustomed to using flash memory for storing code. But that's certainly not the only thing you can do with flash memory. In this article, I'll cover several other useful things embedded designs can do with flash memory. Get ready to add some useful new techniques to your toolbox.

### FLASH EVERYWHERE

Ah, the good old days before flash memory when read-only memory (ROM) was manufactured with your code in it—if, of course, you happened to have lots of money to spend on a custom chip manufacturing run. Your alternative was the programmable read-only memory (PROM), which came in two flavors: one-time programmable or UV erasable. Because every mistake was costly with program-once PROM, the popular choice was UV erasable. This was my introduction to the embedded project world. I remember programming 2732 EPROMs on weekends, hand-keying hex bytes into a suitcase-sized PROM programmer borrowed from a company with pockets deep enough to actually own such a thing, and always having a few spare chips basking under an unshielded UV lamp for the several hours it took to erase the last round of programming. That's a piece of the "good old days" that I certainly don't miss!

Flash memory chips were a giant step forward. The chip was erased electrically (rather than optically) in minutes or seconds instead of hours, and the entire process could be done in the chip pro-

grammer (or potentially right in the circuit where the memory was used). At first, in-system programming required some extra hardware to provide the typically higher erase and programming voltages. But as the in-circuit programming advantages for manufacturers became more apparent, flash memory chips came to run on the main supply voltage and didn't require special circuitry.

And just as flash memory chips came to supplant factory-programmed or UV ROMs, flash technology quickly found a home in microcontrollers. Previously, typical microcontrollers had been factory-programmed, connected to external ROM (squandering most of their I/O pins!), or occasionally manufactured with UV-erasable EPROM sections. With flash memory, a microcontroller needs no external memory bus and can be reprogrammed in-system (usually with very inexpensive tools), so it's no wonder that this is such a widely used technology. There is a wide array of flash memory-based microcontroller choices available for manufacturers and hobbyists alike because it's such a good technology for storing embedded code.

But storing embedded code isn't the only thing flash memory is good for. Many (although not all) flash memory solutions enable the embedded system to do its own flash memory programming on the fly. If you have a system with that ability, then there are several useful things you can do with it. You can use the flash memory to store non-volatile configuration data. You can use the flash memory for data logging. And you can create a system that's capable of updating its firmware in the field.

We'll look at each of these abilities.

### FLASH BASICS

A ROM chip typically looks like most of a static RAM chip. There's an address bus to specify the byte to read, a data bus to read it, a READ signal that says when to do this, and usually chip select signals to distinguish among multiple chips. A flash memory chip typically adds a WRITE signal, but it's *not* the same as the WRITE signal on a static RAM. In a static RAM chip, a write cycle updates the specified address with the specified data. In a flash memory chip, things are more complicated. A write to a flash memory chip is like access to some complex memory-mapped I/O device.

That's because flash memory is not randomly writable. First of all, changing a bit in flash memory requires more power and substantially more time than a read cycle. Secondly, bit programming is a one-way trip: you can choose to change a bit from its "erased" state (almost always one) to a programmed state (zero), but you can't go the other way. The best you can do is an erase, which takes longer still and clears out an entire block of memory called a sector, not just a selected bit or byte. And while we're listing the limitations of flash memory, there's one more to consider: there is a finite number of times you can change a bit before it is worn out and stops working.

So, when you program a flash ROM or a flash memory-based MCU, your programming tool is erasing the chip (or at least the necessary sectors), and then, byte-by-byte, programming bits as needed. Even a limited-reuse flash

memory part typically supports a minimum of a thousand reprogramming cycles (many support a million cycles), so you don't usually have to think about the number of programming cycles except for very well used parts (which may start showing noticeably slower erase times as a warning).

The process of erasing a sector of flash memory or programming a byte is accomplished by writing commands to the flash memory part. There is generally a set of write operations you perform to "unlock" the flash memory before it will accept any erase or programming commands. (This is a very good thing for in-system flash memory: you don't want a bug or a power glitch accidentally changing your code!) Then there is a command you send to begin an erase sequence, a different command you send to begin a program sequence, and some way to poll the device to know when your sequence has been completed.

The details vary widely between parts: the unlock sequence, which commands do what, what timings should be used, how to determine when an operation is finished, and so forth. Most flash memory parts have additional features, like the ability to "lock" sectors against reprogramming. For flash ROMs designed to the same or similar pin layouts, there is typically some agreement between vendors on how you can query the part to determine how you perform further operations. But in any case, if you are going to try to program flash memory from within your code, you need to read the component's datasheet for the details.

Don't forget that some microcontrollers with flash memory do not have the ability to reprogram the flash memory from within their code. For example, I've worked with an older 8051 derivative, the Atmel AT89S53,

which had 12 KB of flash memory but could be programmed only from outside while the chip is in reset. The techniques in this article cannot be used with such a processor. But then this article may suggest to you why you might want to select a microprocessor that can do these things. Refer to the NOR and NAND Flash sidebar for more information.

## PROGRAMMING FLASH

To program flash memory from code, you need a routine for each of the two essential operations: erasing a sector and programming a byte. The flash memory manufacturer's documentation will tell you the various commands and typically provide you with a flowchart of how these are accomplished. Read these carefully. Don't expect to find much in the way of code samples, though! The specifics of your code depend on the hardware you're using.

One complication in the code is that many flash memory parts specify certain timings, such as how much time must elapse between chip setup and programming, how long you should wait after a program command before you start polling for completion, or how long you should poll before determining that an error has occurred. As with any embedded code that needs to conform to timing specifications, the correct code depends on your processor and its clock speed.

A bigger and less obvious complication is that when you are programming a flash memory part, it's functioning as an I/O device instead of as a random-access memory. But if the flash memory you are programming is also your main code storage memory, as is usually the case, then you have a problem: while you're programming the flash memory, the processor won't be able to read code from it! So, the code to

erase or program the flash memory has to run from some other memory. Some systems or microcontrollers have more than one flash memory block, so you can run from one while programming the other. More often, you have to run your flash memory code from RAM. And if your interrupt handlers are in the flash memory being programmed, those have to be turned off too because the interrupt code can't work.

## CODE EXAMPLES

Let's look at some sample code for programming flash memory. My first example is for programming the Spanion Am29LV040 flash memory device (originally from Advanced Micro Devices). These flash memory chips offer 512 KB arranged in eight uniform sectors of 64 KB. They provide a straightforward programming model with no special timing issues. Table 1 shows some of the command definitions used by the flash memory chips. Figure 1 shows a flowchart for polling for the result of a program or erase operation. Both are taken from the chip's datasheet.

Listing 1 shows the C code for the program and erase operations. For simplicity, this code is from a 32-bit microprocessor environment (a Freescale Semiconductor ColdFire system), where the flash memory chip being programmed isn't used for main program code. The two routines are similar. Interrupts are disabled, and the command bytes described in Table 1 are written to the flash memory chip. Then the polling begins. While programming or erasing, bit 7 of the read back will show the opposite of what you are programming. When the erase or program has succeeded, bit 7 of the data will read back what we're looking for. Separately, bit 5 will show a zero while the chip is working. But if the

### NOR and NAND Flash

In this article, I describe the flash memory technology most often used in microcontrollers and discrete memory chips: NOR flash memory. But there is another type of flash memory. The newer NAND flash memory doesn't function as a system memory technology; it functions as a mass storage I/O device.

NAND flash memory erasing and programming is typically faster than NOR flash memory and tends to support more

rewrite cycles. But like a disk drive, it may include some "bad sectors" that have to be managed. This technology is used in flash memory sticks and portable media players, where sequential access to large data content is more important than rapid random read access of individual bytes.

NOR flash memory is still the more common type of flash memory used in microcontrollers because it allows direct execution of code. I focus on NOR flash memory in this article.

Command sequence	Cycles	Bus cycles											
		First		Second		Third		Fourth		Fifth		Sixth	
		Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data
Read	1	RA	RD										
Reset	1	XXX	F0										
Program	4	555	AA	2AA	55	555	A0	PA	PD				
Chip erase	6	555	AA	2AA	55	555	80	555	AA	2AA	55	555	10
Sector erase	6	555	AA	2AA	55	555	80	555	AA	2AA	55	SA	30
Erase suspend	1	XXX	B0										
Erase resume	1	XXX	30										

XXX = Don't care, RA = Read address, RD = Read data, PA = Program address, PD = Program data, SA = Sector address (any address within sector)

**Table 1**—Check out these selected command definitions for the Am29LV040 flash memory chip.

chip times out (the program or erase fails), bit 5 is set to one, telling you of a failure. So, the code is separately watching for that bit to indicate that the routine should not continue to loop forever. (In this error condition, you also have to put the chip back into regular Read mode using the F0 reset command. With ordinary success, the chip automatically switches back to Read mode once it completes.)

My second example is from an older embedded system using a Zilog Z180 microprocessor and either a 128-KB Am29F010 or 512-KB Am29F040 chip as the main code storage. These flash memory chips use the same commands and polling sequence as the Am29LV040 in the first example.

In a Z180 system, the 64 KB of logical address space is mapped onto a 1-MB physical memory space through the Z180's memory management unit. In my example system, the flash memory occupies the bottom 512 KB of physical memory and the bottom 32 KB of logical memory. The top 16 KB of logical memory (C000<sub>H</sub>–FFFF<sub>H</sub>) is set aside for banked memory and remapped to various locations in flash memory or RAM as needed. To erase or program some of the flash memory, the Z180 code would first map the appropriate section of the flash memory into the banked memory space.

Listing 2 is the Z180 assembly language code for the program and erase operations. This is doing the same kinds of work as we saw in Listing 1, but in assembly language. Listing 3 shows a snippet of C code for calling the program routine. First, the real program code is copied into RAM. The C code then calls

the RAM copy instead of the version in ROM. The assembly code was written with no absolute addresses so that it could be copied without having to do any special relocation.

The specifics of how you program and erase flash memory vary between systems. In any case, you need code for these two basic functions. With these in hand, you can start to put your flash memory to additional use.

## CONFIGURATION DATA

Embedded systems often need a provision for accepting some elements of data that can be changed (infrequently) in the field. This could be threshold or timing constants, a password, or just a start-up state choice. Because these choices are not likely to undergo a large number of changes over the life of the system,

flash memory is a reasonable method for handling the storage.

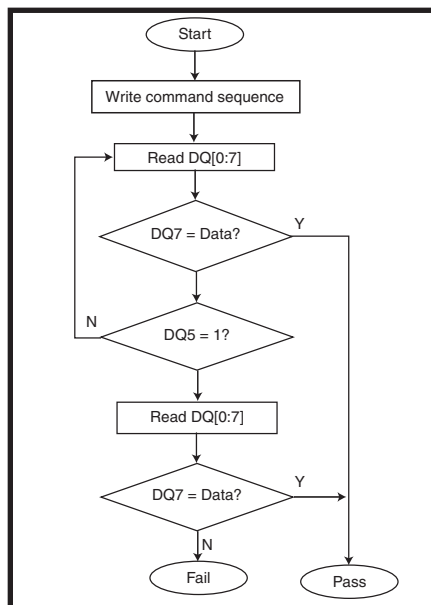
The quick and dirty solution is to dedicate a flash memory sector to each configuration item so that any change represents an erase-and-reprogram cycle. Of course, if you have more configuration choices than available sectors, this won't work. The next obvious approach is to store all of the configuration items

together in a structure in a sector and then erase and reprogram all the choices when any one changes. However, beyond causing more erase-and-program cycles than are really necessary, this approach is at risk if Murphy's law should strike. All of the settings will be lost if the system were to lose power between the erase and the completed reprogram.

A better approach for a system with multiple configuration values is a tokenized storage system that relies on a useful if not obvious fact of flash memory: you can change a value without erasing it, as long as the new value you want differs from the old value only by clearing additional bits (not setting any cleared bits). For example, you can always reprogram a byte with 0101010<sub>B</sub> to, say, 00010001<sub>B</sub>..., but not to 01110111<sub>B</sub>.

My approach to storing configuration data in flash memory involves defining a numeric token for each possible configuration value in the range 41<sub>H</sub> through 7E<sub>H</sub> and setting aside two sectors (or sets of sectors if it will take more than one sector to hold all of the choices) for the storage. One sector (or set) holds the working configuration values while the other is held in reserve. Within the working configuration sector, the first byte is an "in use" flag with a value of 00<sub>H</sub>. After that comes a list of configuration items, each consisting of a token value byte, a length byte, and the number of bytes of value information for the token. (If every choice has the same size of information, you don't need the length byte.)

To retrieve the value associated with a token, your code simply walks



**Figure 1**—Take a look at the data-polling flowchart for the Am29LV040 flash memory chip.

through the active sector, starting after the “in use” byte. It looks for the desired token and skips over the length for any token you don’t want. If you reach a token of FF<sub>H</sub>, the token you are looking for isn’t stored.

To set the value associated with a token, your code similarly walks through the active sector and checks tokens. If it finds the token, it first replaces the token with the same value (logical-ANDed with 3F<sub>H</sub>) and then replaces it again with a 00<sub>H</sub> token as a means of invalidating the value. When the code reaches the FF<sub>H</sub> byte, any old value has been removed and you are at the correct position to write the new value.

However, it is possible that writing the new value might run off the end of the sector because each “replacement” of a value consumes more of the sector without ever freeing any of it. So, if your new value doesn’t fit, it’s time to condense. What you do is program a pending value (say, 01<sub>H</sub>) into the first byte of the free sector. Then do a copy loop. For each token in the active sector that is a valid value (41<sub>H</sub> through 7E<sub>H</sub>), the data is copied to the pending sector, while any invalid token in the active sector is skipped. The result is that only the good values are now stored in the pending sector. After the copy is complete, rewrite the first byte of the pending sector to 00<sub>H</sub> and then erase the previous working sector. You are now ready to write the configuration value to the end of the newly active sector.

Writing the configuration value starts by programming the token, logical-ORed with 80<sub>H</sub>, and then the length and data. After this has been completed, go back and reprogram the token to the desired value.

If it seems like there are a few extra reprogramming cycles in all of this, you’re right. But they are there to protect against a power failure while any portion of the programming or erasing is still in progress. The device initialization code needs to check for any incomplete operations and clean up accordingly. If the system fails while copying from the active sector to the pending sector, there will be a sector at startup with the “pending” value in the first byte: erase that sector. If the system fails while adding a new value

**Listing 1**—The C/C++ code is for programming and erasing the Am29LV040 flash memory chip.

```
bool ProgramBytes(volatile BYTE * pMem, BYTE * pData, int cbData)
{
    int i;
    volatile BYTE * pROM = (BYTE *)((int)pMem & 0xFFF80000);
    BYTE n;
    for (i=0; i<cbData; i++)
    {
        // Start the unlock sequence...
        disable_interrupts();
        *(pROM+0x555) = 0xAA;
        *(pROM+0x2AA) = 0x55;
        *(pROM+0x555) = 0xA0;
        // Write the data to the ROM...
        pMem[i] = pData[i];
        enable_interrupts();
        // Wait for verification...
        while (1)
        {
            n = pMem[i];
            if ((n & 0x80) == (pData[i] & 0x80))
                break; // Good burn!
            if (n & 0x20)
            {
                if (pMem[i] == pData[i])
                    break; // Good burn!
                *pROM = 0xF0; // Reset
                return false; // Program failed
            }
        }
    }
    return true; // All bytes successful!
}

bool EraseFlashSector(volatile BYTE * pMem)
{
    volatile BYTE * pROM = (BYTE *)((int)pMem & 0xFFF80000);
    BYTE n;
    // Start the unlock sequence...
    disable_interrupts();
    *(pROM+0x555) = 0xAA;
    *(pROM+0x2AA) = 0x55;
    *(pROM+0x555) = 0x80;
    // Command the sector erase...
    *(pROM+0x555) = 0xAA;
    *(pROM+0x2AA) = 0x55;
    *pMem = 0x30;
    enable_interrupts();
    // Wait for verification...
    while (1)
    {
        n = pMem[i];
        if (n & 0x80)
            return true; // Good erase!
        if (n & 0x20)
        {
            if (pMem[i] == 0xFF)
                return true; // Good erase!
            *pROM = 0xF0; // Reset
            return false; // Erase failed
        }
    }
}
}
```

to the end of the data, the chain of values will end with an invalid token greater than 80<sub>H</sub>. Do the copy of good values to the alternate sector and stop as soon as you reach the invalid token (because you can’t necessarily trust the length byte that follows it). If the system happens to fail after creating a copy of the active sector but before erasing the other, then you have two sectors, both marked active, and they should contain the same data. Thus, simply erase one.

The logic behind the back-to-back

writes to invalidate a token is not obvious. But if you were simply to go from a token value to 00 in one program cycle and that program cycle were interrupted, only some of the bits might read as fully set to zero, in which case the token could read not as an invalid token but a wrong valid token. (C code for the configuration data logic is posted on the *Circuit Cellar* FTP site.)

## DATA LOGGING

An embedded system will often



### FlashDisk Module

Provides non-volatile, solid state data and program storage for embedded applications. Operating system, application software compatibility, and portability is ensured by the module's True IDE interface. No moving parts allow for more rugged and reliable performance than rotation hard drives.



**SimpleTech** [mouser.com/simpletech/a](http://mouser.com/simpletech/a)

### ConnectCore™ 9C

Powerful ARM9-based core module enables OEMs to design-in core processing functionality and networking capabilities with a single, high-performance solution. Delivers complete embedded network connectivity; additional bandwidth handles many sophisticated embedded applications.



**Digi** [mouser.com/digi/a](http://mouser.com/digi/a)

## Embedded Products for the Latest Technologies



Reduced time to market is critical for new product designs -- lost time means lost revenue. That's why engineers depend on Mouser to deliver a broad selection of embedded products fast!

And because these components have a solution designed in, engineers can utilize these plug-and-play modules from test through production - saving time and money.

Experience Mouser's time-to-market advantage! Our vast selection of the **NEWEST** products, **NEWEST** technologies, **new catalog every 90 days**, no minimums, and same-day shipping on most orders, gets you to market faster. We make it easy to do business with Mouser!



*The Newest Products  
For Your Newest Designs*

**mouser.com** (800) 346-6873

### BL2600 Wolf™

Ethernet-enabled single-board computer features a Rabbit 3000® microprocessor at 44.2 MHz, 10/100Base-T Ethernet connectivity, 512K Flash and SRAM, 12 analog channels, and 5 serial ports. I/O can be connected via IDC headers, friction-lock connectors, and mounting holes.



**Z-WORLD** [mouser.com/zworld/a](http://mouser.com/zworld/a)

### MOD5272 32-bit Processor Module

This module features a web-based control interface, full 32-bit architecture, full suite of TCP/IP protocols, and 10/100baseT RJ45 network interface. Adds network capabilities without taking up valuable design time.



**NetBurner** [mouser.com/netburner/a](http://mouser.com/netburner/a)  
Networking in 1 Day!

### 4000 High-Performance Microprocessor

Low EMI microprocessor for embedded control, communications, and Ethernet connectivity. Glueless architecture, 10Base-T Ethernet, C-friendly instruction set. Up to 60MHz, 8 independent DMA channels, supports 8 or 16-bit Flash and SRAM memories, 7 hardware breakpoints, 40+ I/O lines.



**RABBIT** [mouser.com/rabbitsemi/a](http://mouser.com/rabbitsemi/a)  
SEMICONDUCTORS

# PC/104 Single Board Computers

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 256 M (with ARM Tool Chain), USB WiFi

**200 MHz ARM9**  
Power as low as 1/4 Watt

- 5 boards, over 2000 configurations
  - Fanless, no heat sink
  - SDRAM - up to 128MB
  - 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
  - Linux, Real Time extension, NetBSD
- \$99**  
qty 100
- \$129**  
qty 1
- NEW!**  
▪ SD card option  
▪ VGA video option

Off-the-Shelf Solutions ready to design into  
your project using DOS development tools



TS-5600 Shown with  
optional flash modules,  
A/D, RS-485 and  
Merlin cellular modem

options include:  
RS-485 Half and Full Duplex, A/D Converter up to 8 Channels at 12 bits, DAC up to 2 Channels at 12 bits, Extended Temperature

**133 MHz 586**

- Power as low as 800mA
  - Fanless, no heat sink
  - SDRAM - up to 64MB
  - COM Ports - up to 4 ports
  - Ethernet Ports
  - DIO Channels - up to 40
  - PCMCIA II adaptor
  - Compact Flash adaptor
  - USB Ports (Except on TS-5300)
- \$229**  
qty 100
- \$259**  
qty 1

see our website for 33 MHz 386 configurations

- 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

## New Products and PC/104 Peripherals

### Tiny WiFi Controller boots Linux in 1.1 seconds



**\$249**  
qty 1

- 200 MHz ARM9
- Up to 128MB Flash
- Up to 128M SDRAM
- 802.11g WiFi
- SD Flash Card socket
- 1 external USB port
- 1 10/100 Ethernet
- 3 TTL serial ports

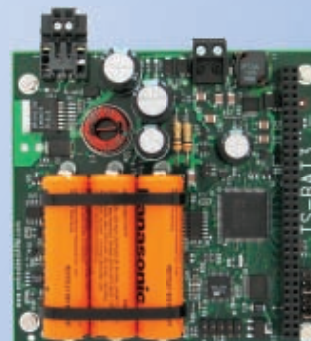
▪ Rugged aluminum enclosure  
measures 1.1" x 4.9" x 3.1"

### Intelligent Battery Back-up

**NEW!**

**\$119**  
qty 1

- Run your system for days  
with no external power source



<b>NEW!</b> ZigBee Wireless	low power wireless, simple serial interface, range up to 1 mile
Modems	33.6K baud, 56K baud, AT commands, caller ID, cellular using GSM and CDMA technologies
Non-volatile Memory	up to 2MB, 10 year lithium battery
Serial Ports	up to 4 serial ports with optional RS-485, opto-isolated available
12 bit A/D, DAC	8 channel 12-bit A/D converter, optional 2 channel 12-bit DAC, A/D jumpered for 0-2.5V, 0-10V or 0-20mA
CAN Bus Controller	Philips SJA1000, opto-isolated, up to 1 megabit/sec selectable termination resistor, Ocera Linux driver
64 Digital I/O	32 inputs, 32 outputs, 200 mA drive, optional 512 Kbyte or 1 MB battery-backed SRAM, stack up to four boards, RoHS compliant

see our website for more boards and option details



We use our stuff.

Visit our TS-7200 powered website at  
[www.embeddedARM.com](http://www.embeddedARM.com)

WIRELESS IS

# Stylish



CTIA WIRELESS 2007 is the world's largest wireless marketplace, drawing tens of thousands of attendees, featuring more than 1,000 exhibitors, 20% international participation from over 100 countries, and representing a \$500 billion global industry with 2.3 billion subscribers worldwide.



*What is wireless to you?*

FIND IT AT **CTIA WIRELESS 2007**

A Division of CTIA-The Wireless Association®

**MARCH 27-29, 2007** ORLANDO, FL  
ORANGE COUNTY CONVENTION CENTER

**WWW.CTIA.ORG/CTIAWIRELESS**

**CTIA**  
The Wireless Association®

**Listing 2**—The Z180 assembly code is for programming and erasing the Am29F010 or Am29F040 flash memory chip.

```

PUBLIC BurnByteBegin
PUBLIC BurnByte
PUBLIC EraseFlashBegin
PUBLIC EraseFlash
PUBLIC WorkRAMbuff
DSEG
BurnByte
EraseFlash
WorkRAMbuff
DEFS 256
CSEG
; BurnByteBegin
; This is the memory image of the BurnByte function, which is copied
; into RAM to run without causing ROM bus cycles. Note that this
; code contains no absolute memory references, since it runs at a
; different address than it occupies in ROM!
; Assembly calling convention:
; DE = mem address
; C = data byte
; return A = 1 if good, 0 if bad
; C calling convention:
; bool BurnByte(char * pMem, char nData)
; Before calling, switch the appropriate ROM sector into the bank
; space. The pointer should be the offset into THAT space.
BurnByteBegin
di
ld hl, 555h
ld (hl), 0AAh ; unlock step 1
ld hl, 2Aah
ld (hl), 55h ; unlock step 2
ld hl, 555h
ld (hl), 0A0h ; program command
ld a, c
ld (de), a ; write the byte
?burn1 in0 a, (WATCHDOG) ; keep alive
ld a, (de) ; get the byte back
ld b, a
xor c
and 128 ; does DQ7 show what we want?
jr z, ?burn2 ; yes: success
ld a, b
and 32 ; does DQ5 show timeout?
r z, ?burn1 ; no: continue
; at this point, we have a failure...
ld hl, 555h
ld (hl), 0F0h ; reset/read command
xor a
ei
ret ; return a failure
; success result...
?burn2 ld a, 1
ei
ret ; return success

; EraseFlashBegin
; This is the memory image of the EraseFlash function,
; which is copied into RAM to run without causing ROM
; bus cycles. Note that this code contains no absolute
; memory references, since it runs at a different
; address than it occupies in ROM!
; Assembly calling convention:
; return A = 1 if good, 0 if bad
; C calling convention:
; bool EraseFlash(void)
EraseFlashBegin
di
push bc
ld hl, 555h
ld (hl), 0AAh ; unlock step 1
ld hl, 2AAh
ld (hl), 55h ; ; unlock step 2
ld hl, 555h

```

(Continued)

need some provision for recording data in a manner that can survive power cycles. Your hardware can include some sort of battery-backed memory or an external EEPROM part, but you could simply use some flash memory sectors for the same purpose.

The advantage to logging data in flash memory is that no extra parts are needed, and your system may already have enough extra space in the flash memory to fit your storage needs. There are also some drawbacks. You need to consider the number of program/erase cycles you expect the system to experience. In addition, you need to make sure that the time it takes to erase a sector doesn't violate any timing requirements for interrupt servicing or polling.

In a typical flash memory data-logging application, your code views the flash memory space as a ring buffer (see Figure 2). In this buffer, log samples can never cross a sector boundary, at least one sector is always empty, and the first byte in each sector represents an "in use" flag. (Ideally, your log samples would be of uniform size and divide evenly into the sector size. In addition, they would start with some token identifier that could never have the value FF<sub>H</sub>.)

At start-up, your code would search the buffer space, sector by sector, looking for the first sector that is erased (starts with FF<sub>H</sub>), and then for the next sector that is in use (starts with any other byte value). The first sector in the ring after an erased section is the oldest record. The last sector before the erased section contains the newest record. (If the code doesn't find a used sector, the log is empty.)

When adding a log entry, your code looks for the next spot to write, advancing if necessary to the next sector. However, if this advance moves into the last empty sector, then the sector containing the oldest data needs to be erased. Once the erase is complete, you can proceed with logging into the previously empty sector.

Because the system can fail while performing the erase of the oldest sector, the start-up code might want to check to see if the next write to the log requires moving into the free sector. If so, it's possible that an erase was underway got interrupted, so the

oldest sector should be considered unreliable and simply erased.

## FIRMWARE UPDATES

When your embedded system's code is contained in flash memory, you may want to take advantage of the in-system reprogramming to allow your system code to be updated in the field. Many manufacturers take advantage of this ability to provide their customers with feature updates after the sale. (OK, so more often this is used to apply bug fixes after the sale. Does this benefit the customer or encourage the shipping of unfinished code? That's another topic.)

Of course, you don't have to be a manufacturer to value firmware updates. If your embedded system is located in some remote or hard-to access location, the ability to update firmware remotely can be valuable. Your one-time design might be located in the next county, at the bottom of a well, or out in space—none of which lend themselves to swapping a chip at need!

The trick to updating firmware isn't in the programming. I've already covered that. The two key questions to a firmware update are: Where does the new code come from, and how does the system work when its firmware is removed?

The new firmware can get into the system by a variety of means. For a consumer application, there may be some existing media interface to which an update can be attached: a memory stick interface, a USB port that can read a USB stick, or some sort of disk drive. For less accessible applications, there is often some sort of communications between the embedded device and a PC (e.g., wireless technology, a serial port, or Ethernet).

And then you get to the tricky part. In order to reprogram the firmware, you need to erase the old firmware. This means that there is a period of time when your system doesn't have its firmware. Sure, you can execute the reprogramming code from RAM, but what happens if your system loses power at this point? Unless you've designed very carefully, what you have then is a dead system. If you're like me, you've applied firmware updates to a PC BIOS or to some embedded peripheral and looked with alarm at the warning that tells you not to interrupt the update or

Listing 2—Continued.

```
        ld    (hl), 80h        ; erasure command
        ld    hl, 555h
        ld    (hl), 0AAh      ; confirm step 1
        ld    hl, 2Aah
        ld    (hl), 55h       ; confirm step 2
        ld    hl, 0C000h      ; HL = pointer into bank area
        ld    (hl), 30h       ; sector erase command
?erase1 in0 a, (WATCHDOG)    ; keep alive
        ld    a, (hl)
        ld    b, a
        and  128              ; does DQ7 show completion?
        jr   nz, ?erase2      ; yes: success
        ld    a, b
        and  32               ; does DQ5 show timeout?
        jr   z, ?erase1       ; no: continue
; failure result...
        ld    hl, 555h
        ld    (hl), 0F0h      ; reset/read command
        xor   a
        pop  bc
        ei
        ret                  ; return a failure
; success result...
?erase2 ld    a, 1
        pop  bc
        ei
        ret                  ; return success
```

turn off the power. And what, pray tell, happens if the power goes out? In many cases, shipping back to the manufacturer, that's what, because your product has lost its mind. And for your remotely located embedded system, that's just not an acceptable answer. A well-designed system has to start with the position that there must never be a time when the system is not recoverable.

One approach I've seen involves having enough code flash memory to store the firmware twice. That is, the old firmware is present and running while programming the new firmware into a different set of flash memory sectors. After the update has completed, then and only then is the old firmware erased. A little bit of start-up code has the job of deciding which firmware copy to trust in the event that both copies are present (because the programming or erase didn't finish). Otherwise, code runs from the trusted memory.

This approach is easiest when your system copies code from flash memory into system RAM at start-up (or when it has some memory mapping system that can make two different physical memory ranges occupy the same logical memory space). Otherwise, every firmware update has to have two binary images, one for each memory area. This approach also suffers if your

microprocessor has a lot of fixed memory addresses for interrupt vectors. Then you have to have them all in some fixed code (never erased) that reroutes these events to the trusted firmware set.

A different approach—which I've used many times—is to build a set of "boot code" (that is never erased) that ordinarily starts the main firmware but stands in for it in the event that the main firmware isn't present. This boot code contains enough capability to communicate with the host PC (or other means of accepting a new update) and program it into place. If the system restarts (incomplete), the boot code is running to let the user know the update didn't work and it should be performed again.

This boot code can be fairly complex. In one family of systems I've worked on, the mechanism for updates was over Ethernet, so the boot code needed to include a fully functional TCP/IP stack and a multithreading operating system just to make it possible to restart a botched firmware upload. Don't do anything this complicated unless you are prepared to thoroughly test the failure cases. This is, after all, the code you can't update later.

But either the dual code storage approach or the smart boot code approach gives you a margin of safety. It also enables you to program the flash

**Listing 3**—The Z180 C code is for executing the assembly language program and erase routines from RAM.

```
extern BYTE WorkRAMbuff[256], EraseFlashBegin[1], BurnByteBegin[1];
bool BurnByte(BYTE * pAddr, BYTE nData);
bool EraseFlash(void);
// ProgramBytes
// Burns the passed data into a specific memory address. The
// address must already be visible in the banked memory window.
bool ProgramBytes(BYTE * pMem, BYTE * pData, int cbData)
{
    memcpy(WorkRAMbuff, BurnByteBegin, sizeof(WorkRAMbuff));
    while (cbData)
    {
        if (!BurnByte(pMem++, *pData++))
            return false;
        cbData--;
    }
    return true;
}

// EraseFlashSector
// Erases a sector of Flash back to FFs. The sector must already
// be visible in the banked memory window.
bool EraseFlashSector(void)
{
    memcpy(WorkRAMbuff, EraseFlashBegin, sizeof(WorkRAMbuff));
    return EraseFlash(void);
}
```

memory as the firmware downloads (i.e., transfer a sector, program it, and on to the next one). The risky approach of just running an erase-and-program routine from RAM requires that the entire firmware was first loaded into more RAM, which means you have to have enough memory to hold the new firmware, the erase-and-program code, and anything else needed to keep the system afloat. Embedded systems often have substantially more flash memory than RAM, which leaves you no room for a temporary copy of the firmware anyway.

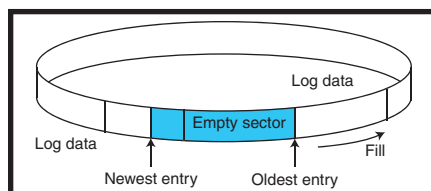
The start-up or boot code must address all the needs of your particular microcontroller for a working system. The reset code and any fixed interrupt vectors must all go first to the permanent code and only proceed to the main code when it can be used. Recognize that a system of fixed interrupt vectors is necessarily going to be slower to respond in such a system. (A much better solution is to use a processor family where the interrupt table is programmable so there is no performance penalty.)

Something else to consider is that you don't want to program the firmware incorrectly! You should have some validation that your data is being programmed into the correct addresses with no lost data, and with some checksum or CRC validation that the firmware data is trustworthy, both when transferring it into your system and when your

boot code is deciding whether to trust the firmware it sees. I typically make the first few bytes of my firmware image a "signature value," which the upload code programs only after it has programmed everything else. If the boot code doesn't see the correct signature, there's no point in even testing the remaining memory. The upload didn't complete, and you need to try it again.

If you are using a serial port for transfers, you might send your firmware data in either Motorola S-Record format or Intel hex format. Either of these is preferable to sending simple binary data. Most embedded tools can generate one of both of these formats, and the address information and checksums help you detect any lost data. Better still is to develop your own communications protocol, where the system accepts a block of data, programs it, and only then gives the host PC the go-ahead to send the next block.

Test the update code frequently. When your system is still on a bench, deliber-



**Figure 2**—The flash memory is arranged as a ring buffer for storing log entries. There is always one empty sector following the sector receiving the newest entries.

ately interrupt the process with loss of communications and loss of power at various points and make sure that you can always recover. This is your insurance policy against future changes to your code. Be sure the insurance works.

## FLASH POWER

Whether you are building code insurance with in-system reprogramming of your firmware, logging data, or storing configuration data, there is a lot that you can do with flash memory besides just holding your latest code. Hopefully, these examples will spark your imagination. How will your next project benefit from the power of flash memory? ☒

*Mark Bereit (www.markbereit.com) has been developing software and designing hardware since 1984. He spent six years as the founder and head of a small business offering contract hardware and software development for other businesses. For the past eight years, Mark has been the director of product development for IRIS Technologies, a manufacturer of video technology products.*

## PROJECT FILES

To download code, go to [ftp://ftp.circuitcellar.com/pub/Circuit\\_Cellar/2007/198](ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/198).

## RESOURCES

J. Makwana and D. Schroder, "A Non-volatile Memory Overview," 2004, <http://aplawrence.com/Makwana/nonvolmem.html>.

Spansion, "AM29F010B Data Sheet," 22336, 2006, [www.spansion.com/data\\_sheets/23479.pdf](http://www.spansion.com/data_sheets/23479.pdf).

———, "AM29LV040B Data Sheet," 21354, 2006, [www.spansion.com/datasheets/21354e4.pdf](http://www.spansion.com/datasheets/21354e4.pdf)

———, "AM29F040B Data Sheet," 21445, 2006.

J. Tyson, "How Flash Memory Works," <http://electronics.howstuffworks.com/flash-memory.htm>.

## SOURCES

**Am29LV040 Flash memory device**  
Spansion  
[www.spansion.com](http://www.spansion.com)



# Hot Chips 18

*The Hot Chips Conference officially came of age with its 18<sup>th</sup> birthday. Once again, Tom headed back to Stanford to check out the big chips on campus. The conference may be all grown up, but Hot Chips are always young at heart.*

Quite a heat wave last summer, and I'm not talking just about the weather. Once again, the Hot Chips conference, now in its eighteenth year, delivered the goods with dozens of presentations covering the latest and greatest.

Few designers work at the bleeding edge where these chips live, but that's OK. Hot Chips has never been about what you should design in today, but rather it has been about foretelling of the features and philosophy of the chips you'll design in tomorrow.

Indeed, the story isn't just about chips either. Despite the conference name, every now and then the organizers go beyond silicon to come up with something interesting. Let's start with a technology that predates silicon yet gives chips a run for their money in terms of delivering more for less.

## SPIN TO WIN

In my article on new-age memory chips a couple of months back ("Memory Lane Change," *Circuit Cellar* 196), I made the point that they are the Rodney Dangerfields of silicon, unable to get any respect and toiling in the shadows of their headline-grabbing microprocessor superiors. But even within the world of storage, there's a pecking order that has memory chips lording it over their merely mechanical counterparts, namely hard disk drives. Don't hold your breath waiting for a "Hot Disk" conference. But unless you're Houdini, don't hold your breath waiting for the oft-predicted demise of

disk drives either.

Yes, motors spinning rusty metal platters seem like an artifact of the industrial revolution compared to fancy-schmantzy silicon. And yes, some rotating media has passed on to that great bit bucket in the sky (e.g., vinyl records and the floppy disk). But the hard disk folks aren't going down without a fight.

Let's put it in perspective. In "Memory Lane Change," I marveled at the fact the latest and greatest memory chips are roughly 1 billion times better than the originals, all things considered (i.e., speed, power, size, and price). Well, it's now been 50 years since the invention of the hard disk by IBM, so now is a good time to look back and take stock of the progress.

That pioneering Random Access Method for Accounting and Control (RAMAC) unit was as big as a refrigerator and weighed a ton (see Photo 1a). Price-wise you were looking at thousands of dollars, not to purchase a unit outright but merely to lease it for a month. Packed inside RAMAC were 50 pizza-sized platters that totaled a whopping 5 MB of storage and delivered data at a leisurely 10 Kbps.

Flash forward to the conference and the latest and greatest in disk drives courtesy of Toshiba.<sup>[1]</sup> They're shipping 4 GB versions of their 0.85" disk drives (see Photo 1b). Better yet, by taking advantage of perpendicular recording techniques—which inject the bits deeper into the platter, thereby increasing a real density—they recently

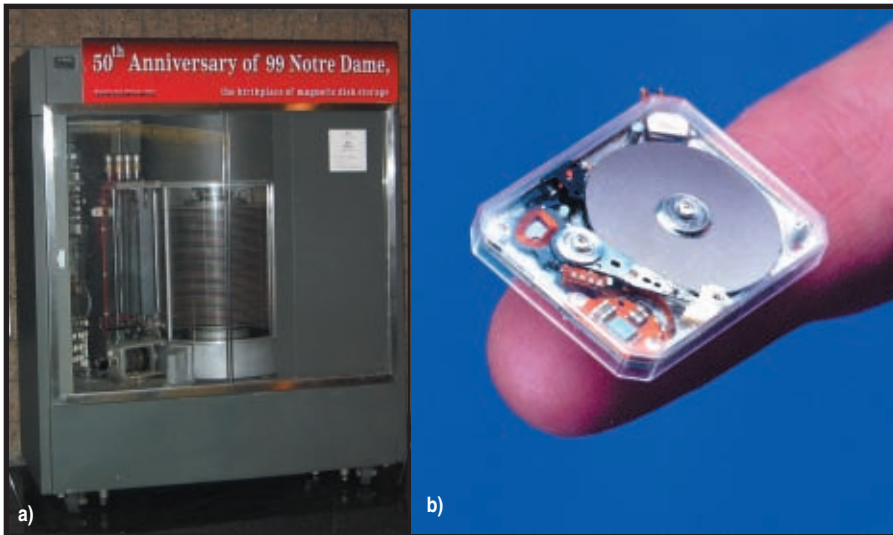
demonstrated a 10-GB prototype.

Do the math comparing RAMAC with the new Toshiba unit and you can see that disk drive advances have far outpaced those of their memory chip brethren. The capacity (10 GB versus 5 MB) and transfer rate (12.5 Mbps versus 10 Kbps) gains alone provide a million-fold-plus gain. And while today's memory chip is about the same size as the original, you could probably fit 100,000 of the Toshiba disks in the area occupied by RAMAC. Then you could throw in at least another factor of a 1,000 or so reflecting the price difference, not even considering inflation. And don't forget power consumption, which surely must add a bunch more zeros. I'm having trouble keeping track, so what the heck, let's just call it a trillion-fold improvement.

Indeed, the trend might see disks not only holding their own, but even replacing memory chips in certain applications. For example, even as MP3 players and cell phones rely on flash memory at the low end (e.g., the iPod Nano), the growing desire for mobile video and other bloatware features pushes demand back into the hard disk camp.

Ah, but what about power consumption and durability? Surely the mechanics of a disk drive are no match for the efficiency and robustness of a chip, right? Historically, that's been true, but the gap is shrinking along with the disk drives themselves. Chips have Moore's law on their side, but ever tinier disk drives





**Photo 1**—Do you know the way to San Jose? You should, because it was home to the IBM research lab that developed RAMAC (a), the first hard disk drive. From those humble beginnings, this Toshiba 0.85" drive (b) demonstrates how far we've come—and the party isn't over yet.

have Newton's law on their side.  $F = M \times A$  means a less massive disk requires less force (i.e., power consumption) to do what you want (e.g., spin the platter and move the head) and can tolerate more shocking accelerations when something bad happens.

Ironically, it's a memory chip that can make all the difference in terms of hard disk power consumption and robustness. For an MP3 player, an entire song that might play for 5 minutes can be loaded into a built-in buffer RAM with less than 1 s of hard disk access. That means the disk is idle (consuming little power) with the head moved to a safe position more than 99% of the time.

The digital revolution owes much to Rey Johnson who led the IBM team in San Jose that invented RAMAC 50 years ago. He may not have a "law" named after him like Gordon Moore, but his invention has and continues to deliver the goods.

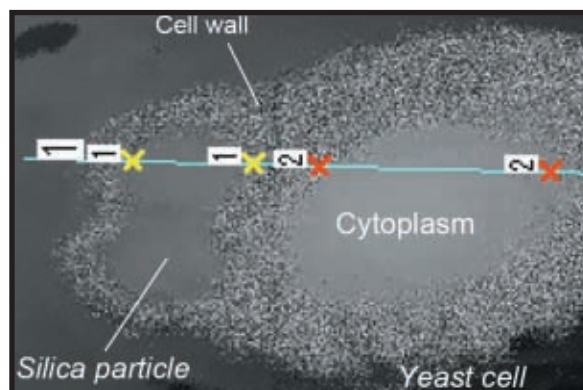
## FANTASTIC VOYAGE

Maybe a MEMS disk drive is just around the corner. In the meantime, the automotive market continues to lay the foundation for MEMS move into the mainstream. Considering new features like active suspension, stability control, and tire pressure monitoring, along with the

ever-growing collection of airbags, each new vehicle is a veritable MEMS showroom.

Beyond accelerometers and gyros, some of the more esoteric MEMS concepts are starting to make the move from lab to fab. For example, the aptly named SiTime is now in production with MEMS oscillators that overcome the accuracy and aging limitations of earlier efforts. I'm not sure it's time to sell your quartz futures, but there's no question that SiTime's tiny "chip" oscillators have the potential to replace our little "tin can" friends. That's especially true for size-constrained applications, which boil down to pretty much every hand-held gadget these days.

Remember that *Fantastic Voyage* (20<sup>th</sup> Century Fox, 1966) movie from



**Photo 2**—Brave new world indeed. This photo shows a Toshiba "physical antibiotic" experiment in which silica particles were electronically targeted to breach the walls of a yeast cell.

way back when? Besides Raquel Welch doing battle with the antibodies, you may also recall that the gist of the story was to put a group of scientists in a submarine, shrink 'em down to size, and inject them into an ill patient where they engaged in hand-to-hand combat with what ailed him.

Sounds "fantastic" all right, at least until you check out the latest in real-world sci-fi from the Toshiba Advanced Electron Device Laboratory. They're fusing medical science and engineering to come up with a MEMS-based "physical antibiotic."<sup>[2]</sup> Although you won't find them on the shelf down at your pharmacy anytime soon, Toshiba has demonstrated success in breaching yeast cell walls with silica nanoparticles using MEMS-vibrator induced heating (see Photo 2). Can a mini-me Roto-Rooter that'll ream our arteries of all that fast-food detritus be far behind?

## SAY WHAT?

Voice recognition is one of those "tomorrow's technologies of tomorrow" that never quite seems ready for prime time. Its true progress has been made by juggling the trade-offs between vocabulary size, speaker dependence, and recognition speed. Notably successful and practical examples that come to mind include hands-free dialing for cell phones ("phone home") and automated phone directories.

But as for the holy grail of human-like voice recognition, even the mightiest computers have a tough go of it. Some of you have no doubt seen the video floating around on the 'Net of a Microsoft marketer living the nightmare of a live demonstration of Vista voice recognition run amok, with laughably bad results. Just Google "Vista Voice Demo" or some such to find a copy. Maybe they should call it "Voice Wreckognition."

The problem isn't so hard. In fact, according to researchers at Carnegie Mellon University, the problem is soft, as in the bloaty software behind the curtain of current voice recogni-

Recognizer	Word error rate (%)	Clock (GHz)	Speedup over real time
SPHINX 3.3 (fast decoder)	7.32	1	0.74x
SPHINX 4 (single CPU)	6.97	1	0.82x
SPHINX 4 (dual CPU)	6.97	1	1.05x
SPHINX 3.0 (single CPU)	6.707	2.8	0.59x
<b>Hardware model</b>	<b>6.725</b>	<b>0.125</b>	<b>1.67x</b>

**Table 1**—Compared to a software speech recognizer (SPHINX) running on big iron, the Carnegie Mellon hardware approach proves a little silicon can go a long way. Hardwiring the basic recognition algorithms delivers competitive performance while slashing cost, power consumption, and clock rate.

tion schemes.<sup>[3]</sup> For speaker-independent, large vocabulary, continuous speech recognition, they've come up with a 1-10-100-1000 rule of thumb that goes something like this: "To achieve 1x (i.e., real-time) recognition with a 10% or less error rate requires 100 MB of software on a computer consuming 100 W with a 1,000-MHz CPU."

Today's PC should be able to handle that. Indeed, if you Google the subject a little further, you'll learn that the aforementioned Vista demo faux pas is said to be attributable to a bug (improper gain setting), and you'll also find video of more successful demos. But even though a PC may be theoret-

ically up to the task, presumably it has something better to do. After all, the CPU in a PC could also push the pixels around on the screen, but that's left for a graphics coprocessor. PC aside, you can see how the 1-10-100-1000 rule that's an inconvenience on the desktop is a showstopper for something like a cell phone.

Furthering the challenge, interest is emerging in the concept of "faster-than-real-time" recognition. The example used in the CMU presentation is saying "Hasta, la vista, baby!" to search for the movie where Arnold says that famous line. Or how about humming a few notes to find a song? Of course, to the degree faster than

real-time search is a goal, it directly ups the ante in terms of processing power.

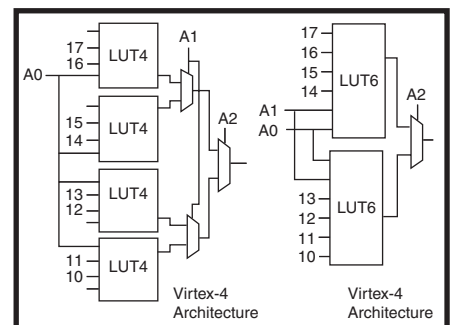
As an aside, the latest take on the subject of voice recognition exploits "lip-reading," supplementing the audio input with video of the speaker.<sup>[4]</sup> Indeed, in favorable (i.e., "talking head") conditions, video-only recognizers by themselves can do a pretty good job. I wouldn't write the eulogy for QWERTY just yet, but maybe there's a combination of audio and video input backed by dedicated hardware that can take voice recognition to the next level.

Got a hard problem? That's why they call it "hardware." When the going gets tough, the tough design a chip, which is just what the CMU researchers are doing with their "In Silico Vox" project.

They started with bit-by-bit analysis of the way software recognizers work. Basically, it's a three-step process comprised of acoustic front-end, feature extraction, and word search, the latter complicated by the need for language context and awareness to discriminate between, for example, *bye*, *buy*, and *by*.

Next, they focused on portions of the existing algorithms amenable, or modifiable to be so, to execution by dedicated hardware. The resulting hardware requirements are quite different from a general-purpose CPU because features like floating point and giant caches turn out to be of little use for the specific task at hand (er, mouth).

Understanding what's going on under the hood, the CMU team has



**Figure 1**—As this example of an 8:1 mux demonstrates, the new six-entry LUT in the Xilinx Virtex-5 can boost performance by reducing the logic depth and signal fan-out compared to the older four-entry LUT.

## Only 4 Steps...

...are required to generate efficient, reliable applications with the  $\mu$ Vision IDE and development tools from Keil.

### Step 1. Select Microcontroller and Specify Target Hardware

Use the Keil Device Database ([www.keil.com/dd](http://www.keil.com/dd)) to find the optimum microcontroller for your application.

In  $\mu$ Vision, select the microcontroller to pre-configure tools and obtain CPU startup code.

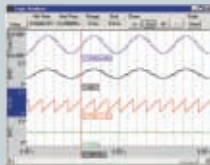
### Step 2. Configure the Device and Create Application Code

The  $\mu$ Vision Configuration Wizard helps you tailor startup code to match your target hardware and application requirements.

Extensive program examples and project templates help you jump-start your designs.

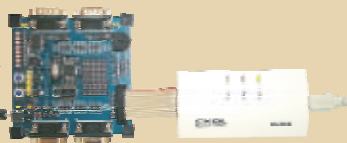
### Step 3. Verify Program Execution with Device Simulation

High-speed simulation enables testing before hardware is available and helps you with features like instruction trace, code coverage, and logic analysis.

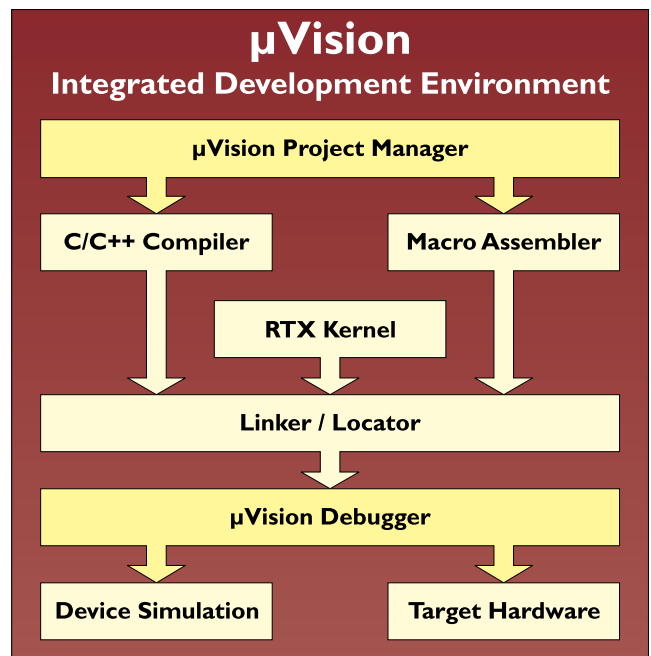


### Step 4. Download to Flash and Test Application

Once your application is runs in simulation, use the Keil ULINK USB-JTAG Adapter for Flash programming and final application testing.



**Keil Microcontroller Development Tools** help you create embedded applications quickly and accurately. Keil tools are easy to learn and use, yet powerful enough for the most demanding microcontroller projects.



Components of Keil Microcontroller Development Kits

Keil makes C compilers, macro assemblers, real-time kernels, debuggers, simulators, evaluation boards, and emulators.

Over 1,200 MCU devices are supported for:

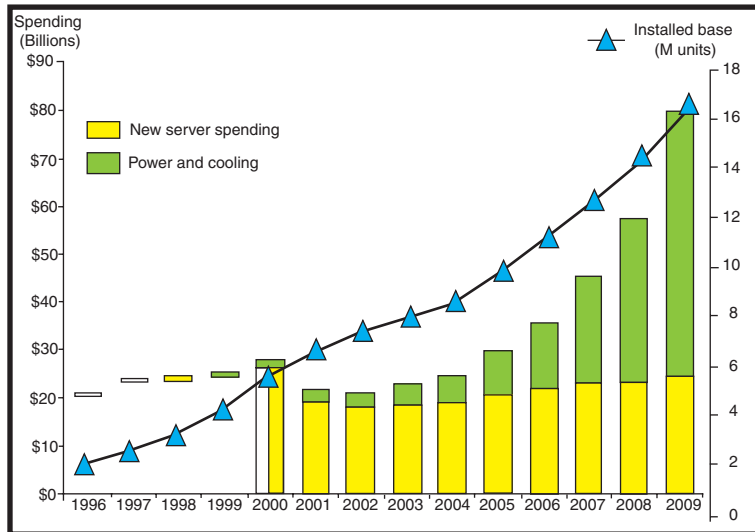
- **8-bit** - 8051 and extended 8051 variants
- **16-bit** - C166, XC166, and ST10
- **32-bit** - ARM7, ARM9, and Cortex-M3

Download an evaluation version from [www.keil.com/demo](http://www.keil.com/demo)

come up with a design for an ASIC that simulates favorably compared to current state-of-the-art software recognizers running on high-performance PCs (see Table 1). Better yet, they've even prototyped a version, albeit slightly less capable (simplified vocabulary, not quite real time), that runs in an FPGA.

Speaking of FPGAs, Xilinx gave a presentation on their latest-and-greatest Virtex-5 FPGA.<sup>[5]</sup> For performance-at-any-price applications such as prototyping an ASIC, this puppy is a designer's dream come true.

Moore's law's stride may be getting shorter, but it's still got legs. With the move from 90 to 65 nm, Xilinx can cram 1 billion transistors on their latest rocket-science chips, even as they predict 5 billion transistors by 2010.



**Figure 2**—The problem with hot chips isn't just that they're too hot to handle. It's getting to the point where power and cooling costs exceed the costs of the computers themselves.<sup>[5]</sup> (Source: J. Humphreys and J. Scaramella, "The Impact of Power and Cooling on Data Center Infrastructure," 201722, 2006.)

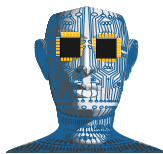
You've come a long way, baby!

Like watching your kids grow, you don't notice the day-to-day changes that turn the little rug rat into the strapping lad who wants to borrow the car. Looking under the hood, it's amazing to see how fancy today's

FPGA is compared to the original.

Actually, the familiar LUT-based logic cell is relatively recognizable, although having grown to six inputs from the originals four (see Figure 1). The wider LUTs enable the depth of logic to be reduced, thereby increasing speed. In addition, now the LUTs are broken into two camps: regular and so-called "M-LUTs." The latter features some local RAM and shift registers.

Moving up the ladder with functional specialization is what really sets the modern FPGA apart. Xilinx has gone as far as to split their product line into four Application-Specific Modular Block (ASMBL) camps: logic, serial I/O, DSP, and PowerPC. Under the hood, there are



# DESIGNCON 2007

Connecting the World of Electronic Design

## FREE Exhibition Pass A \$75 Value

Conference Jan. 29 – Feb. 1, 2007 / Santa Clara Convention Center

Exhibition Jan. 30 – 31, 2007 / Santa Clara, California, USA

Bring this pass to the Santa Clara Convention Center to gain **FREE** access to the following and be entered to **win one of many fabulous high-tech prizes!**

- 125+ Exhibitors
- Plenary and Technical Panels
- Technology Pavilions
- Keynote Addresses
- TecPreviews
- Networking Receptions



International Engineering Consortium  
www.iec.org

[www.designcon.com/2007](http://www.designcon.com/2007)

	Ambric 45-bric chip	Ambric 70-bric estimate	TI C641x DSP	Xilinx Vertex-4 LX100-LX200
Process	130 nm	90 nm	90 nm	90 nm
Megahertz	333 MHz	Est. 450 MHz.	1,000 MHz	500 MHz nominal
Published DSP benchmarks	10–25× throughput, one-third the code	Est. 20–50× throughput, one-third the code	1×	—
Multiply-Accum./Sec. (16 x 16 to 32 bit)	60 GMACS	Est. 125 GMACS	4 GMACS	48 GMACS

**Table 2**—Novel multicore architectures, such as the TeraOps chip from Ambric, offer the promise of shaking things up. And it's not just the idea of putting multiple CPUs on a single die, but also novel programming techniques that make them easy to use effectively. Ambric uses a structural object/message programming language said to significantly reduce development time and cost.

all manner of block RAMs (including dedicated FIFO and ECC logic), DSP accelerators, 500-plus-MHz clock synthesizers, and gigahertz-class pin drivers.

Xilinx put the V5 architecture to the test with a myriad of benchmarks, one example being the latest version of their Microblaze softcore processor. In essentially the same silicon area, V5 was able to both expand the MIPS per megahertz (by stretching the pipeline from three to five stages) and boost the clock rate by more than 10% (180 to 201 MHz) for a total per-

formance improvement of nearly 40%.

Not that the journey onward and upward isn't a bit rocky. The show-stopper of leakage current looms while 1-V operation leaves little margin for error. And wiggling the pins faster (2 V/ns) and harder (50 mA/ns) creates all kinds of signal integrity and noise challenges for the board and box designer. As one slide in their presentation sums up: "Complex chip, complex package, complex board"—to which they forgot to add "complex tools."

Oh well, nobody said it would be

easy. Actually that's not quite true. There's usually some guru that predicts it will be easy soon, but it never is.

## RAMPING UP

If you've followed my yearly columns on the Hot Chips conference, you know there's been a sea change in high-end processor design. Everybody now accepts the fact that simply pumping more steroids into a muscle-bound "SuperDuper" processor is a dead end. Besides diminishing bang-per-buck from architectural bloat, the approach is doomed to go down in flames, literally, as these "hot chips" get too hot to handle (see Figure 2). Although it won't be easy, the only apparent solution, brute force it may be, is to combine multiple simpler and lower-power CPU cores on a single chip, the so-called Chip Multi-Processor (CMP).

More on that in a moment, but first I should emphasize this doesn't mean single-CPU chips are dead by any

**A new era of secure RF**

The ultimate in security for RKE and remote control applications. The HS Series transfers the status of up to eight buttons via a highly encrypted transmission.

CipherLinx™ security technology  
Never sends the same data twice  
Secure data authentication  
Up to 8 inputs  
Never loses sync  
80-bit key  
ISE evaluated  
Easy user setup

**LICAL-ENC-H5001 ENCODER IC**

**LICAL-DEC-H5001 DECODER IC**

**CIPHERLINX TECHNOLOGIES**

"In short, the CIPHERLINX™ protocol in the HS Series is well-designed and is an excellent choice for applications requiring a secure unidirectional link."

**FUTURLEC** (Independent Security Evaluators)

**800-736-6677**  
159 Ort Lane  
Merlin, OR 97532

Visit [www.CipherLinx.com](http://www.CipherLinx.com) for more detailed information.

**Save Up To 60% On Electronic Components**

**Great New ET-AVR Stamp**

- Includes ATmega128 Microcontroller
- Up to 53 I/O Points
- 8-Channel 10-bit A/D
- Direct In-Circuit Programming
- Ideal as a Removable Controller

**ONLY \$19.90**

**Exciting New ET-ARM Stamp**

- Includes LPC2119 Microcontroller
- High-Speed Operation
- Heaps of I/O plus CAN, UART, I2C
- In-Circuit Programming
- Supporting Board Also Available

**ONLY \$24.90**

**Save Heaps on Components**

We carry a wide range of Integrated Circuits, Microcontrollers, Capacitors, LED's and LCD's.

All at very competitive prices.

**ONLY \$5.90**

We are your one-stop shop for Microcontroller Boards, PCB Manufacture and Electronic Components

**www.futurlec.com**

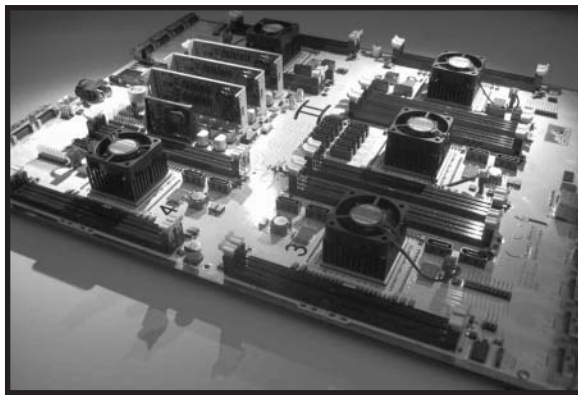
means. The vast majority of simple blue-collar applications are still serviceable by a single CPU now and forever. Nevertheless, whether a single- or multi-core, less power is always better.

The most popular approach to reducing power is clock gating, which allows power management hardware and/or software to dynamically turn off the clock to unused circuits. A related embellishment is dynamic frequency and voltage scaling, whereby the clock rate and supply voltage are adjusted in real time to the bare minimum required to meet the instantaneous performance demand.

The PWRficient family of processors from PA Semi is a good example of a design, where in their words, "Power is the first-order design principle."<sup>[6]</sup> Features include more than 15,000 (!) clock gates and independent dynamic voltage and frequency scaling for each core in their dual-core chip. Special low-power modes allow varying the trade-off between wakeup time and the amount of architectural state that is saved. They even go as far as to customize the  $V_{DD}$  spec for each individual part. The result is a very high-performance 2-GHz, 64-bit PowerPC that consumes a fraction of the power you might expect: just 7 W maximum (4 W typical) for the processor core, including 128 KB of L1 cache.

A more esoteric approach to cutting power consumption is found in the "clockless" asynchronous ARM996HS from Handshake Solutions.<sup>[7]</sup> The async approach can be considered "perfect clock gating" in the sense that the only circuits that consume power are those actually doing something.

The good news is that an asynchronous design always delivers the maximum possible performance for a given set of environmental conditions (temperature, voltage level, and process variation). The bad news is that the environment does indeed change, which means performance can vary. The ARM966HS addresses this concern with the ability to synchronize internal operations to an external clock. In essence, this acts like a



**Photo 3**—Actually making a CPU chip is something few can afford, all the more so for starving students. The RAMP project relies on a bunch of FPGAs and DRAMs packed into a BEE box to provide a common, and relatively inexpensive, proving ground for architectural prototyping and experimentation.

brake that limits performance to a lesser, but fixed and predictable, level.

The results for the ARM966HS are decently impressive. Compared to the synchronous equivalent (i.e., the ARM968E), die size and transistor count are only slightly larger, but the async design consumes only one-third of the power for a given level of performance.

OK, now back to the CMPs. First, let me dispense with the 'x86 PC-side of the story. One comment I overheard expressed fears of a possible "core race" in which the number of cores a PC has itself becomes the goal, not whether they are used effectively. To which I say, be afraid, be very afraid. The fact is, the PC market is already jumping (and will continue to do so) all over multi-core as a keep-up-with-the-Joneses selling feature.

The more interesting action on the CMP front is with embedded chips because they are less constrained by need for compatibility with an existing software base or, for that matter, conventional programming techniques. Let's consider a sampling of CMPs taken from the conference aptly demonstrating the broad scope of architectural alternatives.

The NXP (founded by Philips) PNX8535 "HDTV-on-a-chip" combines a MIPS core with the company's TriMedia A/V accelerator.<sup>[8]</sup> The NXP Nexperia Mobile Multimedia Processor similarly combines a TriMedia with an ARM926.<sup>[9]</sup> The Renesas SH-MobileG1 combines their own SH core with two ARM cores that handle real-time, baseband, and

application tasks respectively.<sup>[10]</sup>

The Sun Niagara-2 I-way server farm chip features eight SPARC cores.<sup>[11]</sup> The UC Davis Asynchronous Array of Simple Processors (AsAP) has 36 16-bit processors in the first prototype, and it's said to be scalable to hundreds and even thousands of cores.<sup>[12]</sup>

The Ambric TeraOps chip is kind of like an FPGA, except the "logic cell" is a 32-bit processor (see Table 2). The first chip has 360 processors, and the next will have 560.<sup>[13]</sup> And finally,

the Connex media processor features 1,024 linearly connected small-and-simple 16-bit processors.<sup>[14]</sup>

Just keep in mind all this CMP stuff is relatively uncharted territory. An architectural brainstorm, like talk, is cheap—but chips aren't. The problem is how to validate a back-of-the-napkin architectural idea without having to cough up the big bucks to actually make a chip.

Enter the Research Accelerator for Multiple Processors (RAMP) project led by David Patterson of UC Berkeley and a stellar cast of luminaries (<http://ramp.eecs.berkeley.edu/index.php>). As an aside, I noticed one of the participants is Jan Gray, whom you may remember as the author of the excellent "RISC in an FPGA" article series back in 2000. Check out his RAMP ruminations in "Mapping CMPs to Xilinx FPGAs."

The idea is pretty simple. Cram a bunch of FPGAs and DRAMs into a box serving as a test bed for CMP prototyping. Yes, at an estimated cost of \$100,000 for a Berkeley Emulation Engine 2 (BEE2) box, it isn't cheap, but it's a heck of a lot less costly and time consuming than spinning a custom chip (see Photo 3). Perhaps most importantly, actually running code on silicon (albeit performance limited to perhaps 10% that of a real chip) delivers results that are fundamentally more credible than simulation or other guesstimates.

## STAY COOL

Hot Chips serves as the checkpoint that enables everyone to calibrate their compass for the journey ahead.

We can see that the hot chips of tomorrow will be the ones that can be cool at the same time. In the meantime, keep that fire extinguisher handy.

The emergence of multicore as the leading design paradigm will shake things up a bit. It will be exciting to see the novel hardware and software schemes that emerge as architects engage in hand-to-hand combat with some really hard problems.

FPGAs continue to march onward and upward. While the rocket-science Virtex-5 and \$100,000 RAMP boxes are bleeding edge, the implications for the mainstream market are clear: FPGA price and performance are headed your way.

With all the excitement, let's not forget the people. I'd like to say thanks to the organizers of the Hot Chips conference, all of whom are volunteers (and many were so from the very beginning), for once again putting on a classy show. Hot Chips was 18, and I liked it. ☺

*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.*

## REFERENCES

- [1] A. Takeo, et al, "Ultra Small HDD for Mobile Applications," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).
- [2] K. Suzuki, et al, "Micro Manipulator Array for Nano-Bioelectronics Era," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).
- [3] R. Rutenbar, et al, "In Silico Vox: Toward Speech Recognition in Silicon," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).
- [4] A. Eisenberg, "What's Next: Beyond Voice Recognition to a Computer that Reads Lips," *New York Times*, September 11, 2003.
- [5] S. Douglass, "Virtex5: The Next Generation 65nm FPGA," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).
- [6] T. Yeh, "The Low-Power, High-Performance Architecture of the PWRficient Processor Family," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

[7] A. Bink, "ARM966HS: The First Licensable, Clockless, 32-bit Processor Core," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

[8] B. Pronk, "Highly Integrated Nexperia PNX8535 Hybrid Television Processor," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

[9] S. Mutz, "Heterogeneous Multiprocessing for Efficient Multi-Standard High-Definition Video Decoding," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

[10] M. Ito, "SH-MobileG1: A Single-Chip Application and Dual-mode Baseband Processor," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

[11] G. Grohoski, "Niagra2: A Highly-Threaded Server-on-a-Chip," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

[12] B. Baas, "Hardware and Applications of AsAP: An Asynchronous Array of Simple Processors," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

[13] L. Anderson, "TeraOPS Hardware: A New Massively-Parallel, MIMD Computing Fabric IC," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

[14] G. Stefan, "The CA1024: A Fully Programmable System-On-Chip for Cost-Effective HDTV Media Processing," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

[15] B. Meyerson, "Collaborative Innovation: A New Lever in Information Technology Development," Hot Chips 18, [www.hotchips.org](http://www.hotchips.org).

## RESOURCES

J. Gray, "Mapping CMPs to Xilinx FPGAs," Microsoft Corp., <http://ramp.eecs.berkeley.edu/Publications/Mapping%20CMPs%20to%20Xilinx%20FPGAs.ppt>.

Hot Chips Conference, [www.hotchips.org](http://www.hotchips.org).

MEMS oscillators, SiTime, [www.sitime.com](http://www.sitime.com).

RAMP Project, <http://ramp.eecs.berkeley.edu/>.

**PIC-SERVO**  
MOTION CONTROL

MOTION CONTROLLERS FOR  
BRUSH, BRUSHLESS AND  
STEPPER MOTORS.

- controller chips
- controller boards

[www.picservo.com](http://www.picservo.com)  
JEFFREY KERR, LLC

**Advertising Calendar**

March, Issue #200  
Robotics  
Space Close: Jan. 12

**BONUS DISTRIBUTIONS:**  
CTIA Wireless &  
Trinity College Robotics Contest

April, Issue #201  
Embedded Programming  
Space Close: Feb. 12

**BONUS DISTRIBUTIONS:**  
Embedded Systems Conference West

Call: Shannon Barraclough  
(860) 872-3064  
[Shannon@circuitcellar.com](mailto:Shannon@circuitcellar.com)

**bob**  
basic overlay board

**Introducing:**  
Vector graphics  
Custom Font Capability  
SPI or RS-232 Interface

**Meet bob-4**  
Video display generator

[www.decadenet.com](http://www.decadenet.com)


**DECADE ENGINEERING**  
503-743-3194 Turner, OR, USA

# 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 sheet. **ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2" x 3" FORMAT.** Call for current rate and deadline information. Send your disk and digital submission sheet to: IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066 or e-mail [adcopy@circuitcellar.com](mailto:adcopy@circuitcellar.com). For more information call Shannon Barraclough at (860) 872-3064.

The Suppliers Directory at [www.circuitcellar.com/suppliers\\_dir/](http://www.circuitcellar.com/suppliers_dir/) is your guide to a variety of engineering products and services.



**phyCORE<sup>®</sup> OEMable Single Board Computers**

**XScale:** PXA270, PXA255

**ARM:** LPC3180 (ARM9); LPC22xx, LPC229x, AT91 (ARM7)

**PowerPC:** MPC5554, MPC5200B, MPC565, MPC555

**ColdFire:** MCF5485      **x86:** Elan SC520

**C166/XC16x/ST10/8051**      **CAN**

**Blackfin:** BF537

**Faster-to-Market:** Save time by integrating a PHYTEC Single Board Computer Module into your target circuitry.

**Make - or - Buy:** Why make your own when you can buy PHYTEC off-shelf solutions, cost-effective to 1000s units/year?

**Integrated Support Services:** Let PHYTEC assist you in the design of your end product: from tools and RTOSes to production. Our hardware is bundled with leading compilers (Keil, IAR, CodeWarrior), RTOSes (WinCE, Linux) and debuggers.

**Immediate Support:** Talk to PHYTEC technical staff with every call. No waiting for answers.

**Your OEM solution:** With 20 years design, production, and integration experience, PHYTEC is your OEM partner.

PHYTEC America, LLC ■ 203 Parfitt Way SW, G100 ■ Bainbridge Island, WA 98110 USA  
[www.phytec.com](http://www.phytec.com) ■ (800) 278-9913 ■ [www.phycore.com](http://www.phycore.com)

## USB

Add USB to your next project—it's easier than you might think!

- USB-FIFO up to 8 mbps
- USB-UART up to 3 mbps
- USB/Microcontroller boards pre-programmed with firmware
- 2.4GHz ZigBee™ & 802.15.4
- RFID Reader/Writer

**Absolutely NO driver software development required!**

[www.dlpdesign.com](http://www.dlpdesign.com)



## C Programmable Controllers & ICs



- Onboard Tiny C interpreter for fast program development
- Multiple program storage
- Free IDE includes text editor, compiler, simulator, serial downloader and test terminal

Full documentation on website

[www.ozitronics.com](http://www.ozitronics.com)

## USB-ICP & LCD-Demo Kit



**USB-ICP** Using a standard USB port, USB-ICP supports In-System Programming (both ISP and ICP mode) for Philips 80C51, LPC9xx, and LPC2xxx ARM7 microcontroller families. In-System Programming uses a two-wire serial interface to program and erase ISP enabled microcontroller devices without removing them from the system. This device is powered over the USB connection, so no external power supply is required! **\$69.00.**



**LCD-Demo Kit** Small footprint "pocket" LCD Demo board with 8-character alphanumeric LCD. Uses a highly integrated Flash based 80C51 device. A single CR2025 3V coin cell battery powers the entire board and the unit is User Re-Programmable via the ICP connector. **\$49.00.**

For XA Development Kits and I2C, MDIO and SPI tools, please visit our website.



**Future Designs, Inc.**  
 2702 Triana Blvd  
 Huntsville, AL 35805  
 (256) 883-1240  
 Fax (256) 883-1241

[www.teamfdi.com](http://www.teamfdi.com)  
 VISA/MC/Amex

## ELECTRONICS MANUFACTURING SERVICES

### PRO-TECH ELECTRONICS CANADA



**IN BUSINESS SINCE 1988**

- BOARD DESIGN / PCB LAYOUT
- PARTS PROCUREMENT
- SMT ASSEMBLY (BGA / FINE PITCH)
- CUSTOM WIRING HARNESSSES
- PROTOTYPES AND PRODUCTION
- WORLDWIDE CUSTOMER BASE

Tel: (905) 760-2185 Fax: (905) 760-2186  
 E-mail: [sales@protechelectronics.com](mailto:sales@protechelectronics.com)

[www.protechelectronics.com](http://www.protechelectronics.com)

## USB Host Stack

- USB 2.0
- EHCI, OHCI, UHCI, ISP116x, ISP1362, ISP176x, ARM, ColdFire
- Hub, Mass Storage, Modem, Mouse, Keyboard, Printer, Serial Converter
- Low Cost, Royalty-Free
- Full Source Code
- Standalone or RTOS
- Device and OTG Available



**Micro Digital Inc**  
 RTOS Innovators

[www.smxrtos.com/usb](http://www.smxrtos.com/usb)



[www.can232.com](http://www.can232.com)

**Only \$108 \$99**

**CAN232 Features:**  
 Free sample programs  
 9-15VDC supply via CAN  
 Timestamp in mS  
 Small size 2.7" by 1.2"  
 100% Bandwidth up to 125Kbit  
 Both 11 & 29 bit ID support  
 32 Message Receive FIFO  
 Works up to 1Mbit CAN  
 Simple ASCII protocol  
 Supports RTR Frames  
 Max 230Kbaud RS232  
 Firmware upgradable  
 No drivers needed  
 OS independent  
 CE Approved

**CANUSB Features:**  
 Free ActiveX component  
 PC, MAC & Linux support  
 Both 11 & 29 bit ID support  
 Simple CAN logger included  
 Free Threaded Windows DLL  
 Firmware upgradable via USB  
 Sample programs in C, C++, VB, Delphi, C#, PureBasic etc.  
 No need for external power  
 Works up to 1Mbit CAN  
 Supports RTR Frames  
 USB 2.0 Full Speed  
 Free USB drivers  
 CE Approved

**Only \$154 \$129**

[www.canusb.com](http://www.canusb.com)

**ANSI C software** development tools for:

**NEW!**  
Global Optimizer

**Version 7 Tools**  
Full-featured 45-day demos on our website

**hardware kits:**  
 Elektronikladen - HC08 & HCS12 / 12 ref. design & starter kits  
 CANDIP - AVR/CAN on a chip!  
 Ethernet - Complete AVR TCP/IP solution

**high powered development, garage powered prices!**

**www.imagecraft.com**  
 info@imagecraft.com  
 706 Colorado Ave. Palo Alto, CA 94303  
 (650) 493-9326 • FAX (650) 493-9329

VISA / MC / AMEX  
Check / Money Order

...say 'NO' to JURASSIC PRICES!

**Add a color touch interface to your embedded product!**

**High level RS232 interface**  
**Low cost interface**  
**Easy to program**  
**In stock!**

Add color graphics to any 8/16 bit embedded system.  
 Easy, fast and flexible. Up and running in hours!

**REACH TECHNOLOGY INC.**  
[www.reachtech.com](http://www.reachtech.com) • 510-770-1417

842 Boggs Avenue • Fremont, CA 94539

**Solve complex signal acquisition problems...**

- positioning & control
- environmental
- acceleration
- transients
- pressure
- vibration
- sonar
- GPS

- Linux Driver
- Guaranteed in stock
- Customization available
- 16-bit analog inputs and outputs
- Million sample FIFO eliminates interrupts
- Wide analog input and output ranges
- -40°C to +85°C Standard

**www.stx104.com**  
**Apex Embedded Systems**  
 sales@stx104.com • 608-256-0767 x24

**Get The Whole Picture**

Camera Interface Application Kit for remote monitoring, security and event-capture via the internet.

Kit includes:

- RCM3365
- Serial Camera
- Servos
- Dynamic C®
- Dev. Tools

**Buy Now \$499**

**RABBIT** Semiconductor  
[www.rabbitappkits.com](http://www.rabbitappkits.com)

**ATMEL AVR Stacking System**

- ATMEL AVR AT90CAN128 Processor
- Versatile Expansion via stacking I/O cards
- Software Libraries Included
- Low Cost
- In Stock
- CAN Network with optional software library
- Real Time Clock
- Expanded Memory

**PIONEER AUTOMATION**

Create a flexible, expandable, embedded control system with our ATMEL AVR stacking control system.

**Call to order: 608-873-9800**  
[www.pioneercontroller.com](http://www.pioneercontroller.com)

**I<sup>2</sup>C/SMBus**

- Bus Monitors
- Protocol Analyzers
- Host Adapters
- Multiplexers
- Battery Applications
- Software Tools

**MCC**  
 Micro Computer Control

I<sup>2</sup>C is a trademark of Philips Corporation

[www.mcc-us.com](http://www.mcc-us.com)

Start designing with  
Freescale's  
advanced S12X family!

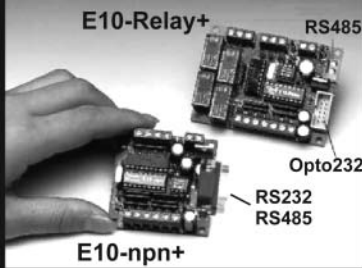


**freescale**  
Alliance Member

Toll-free: 1-877-963-8996  
[www.technologicalarts.com](http://www.technologicalarts.com)

## The \$69 PLC

Work as Stand-Alone Ladder Logic PLC.  
Or as Smart Remote I/Os of PC/ PLCs.  
RS485 allows 256 units to be networked.



Incredibly Easy to Program!  
Our software is used by many  
colleges for teaching PLCs!

**TRI**  
Triangle  
Research

Get Free Ladder Logic Simulator:  
[www.tri-plc.com/cci.htm](http://www.tri-plc.com/cci.htm)

Tel: 1-877-874-7527 - PLC specialist since 1993

**CIRCUIT  
CELLAR®**

back issues available as

*Searchable Archives  
on CD-ROM*

NOW SHIPPING:

CD-ROM #10 2005 Issues 174-185

CD-ROM #9 2004 Issues 162-173

CD-ROM #8 2003 Issues 150-161

Order Online:

[www.circuitcellar.com](http://www.circuitcellar.com)  
or call 860.875.2199

## Electronic parts for less



Low shipping.

FREE shipping on  
qualifying orders.

Digikey pooling  
to save you S/H.

ICD2 clone kit \$36

PIC12F683-I/P \$1.17

PIC16F54-I/P \$0.78

**See website  
for details.**

## DIPmicro Electronics

Fax: (866) 603-7109 [sales@dipmicro.com](mailto:sales@dipmicro.com)

[www.dipmicro.com](http://www.dipmicro.com)

## CONTROLLER AREA NETWORK CAN to RS232 CAN to USB

**Flash CAN232** - The original RS232 to CAN  
adapter. Supports both ASCII and binary  
commands. \$99

**CANPIC CAN232** - Enhanced CAN to RS232  
adapter just \$89

**NEW! CANUSB** - CAN to USB adapter \$99

**Lowest! TrashCAN** - Bare bones CAN to  
USB adapter as low as \$69.95.

**I2C232** - I2C to RS232 adapter only \$89.

**I2CUSB** - I2C to USB adapter only \$89.

## Need a Software Guy?

Over 25 years experience from high volume  
automotive components to one-of-a-kind test  
boxes. Let EMICROS do your next embedded  
software project.

## EMBEDDED MICRO SOFTWARE

email: [info@emicros.com](mailto:info@emicros.com)

[www.EMICROS.com](http://www.EMICROS.com)

sponsored by [ThePokerBug.com](http://ThePokerBug.com)

## Full Speed CAN USB Adapter

✓ Simple configuration & use

✓ 10% Discount for first  
orders Code: GRIDCC20



+1 630-245-1445

Naperville, Illinois USA

[www.c-a-n.com](http://www.c-a-n.com)

**gridconnect™**

## SpectraPLUS 5.0

*Audio Spectrum Analysis*

### Features

Sound Card based I/O  
FFT sizes to 1048576pts, 1/96 Octave  
Up to 24 bit, 200kHz sampling rates  
3-D Surface and Spectrogram  
Digital Filtering, Signal Generation  
THD, IMD, SNR, Transfer Functions  
DDE, Macros, Data Logging,  
Vibration Analysis, Acoustic Tools

**FREE 30 day trial!**

[www.spectraplus.com](http://www.spectraplus.com)

**PHS**

Pioneer Hill Software  
360 697-3472 voice  
[pioneer@telebyte.com](mailto:pioneer@telebyte.com)



USB to CAN Interface  
USB 2.0 Hi-Speed 480Mbps!

Other companies may claim USB 2.0 but  
if it isn't Hi-Speed it is only 12Mbps.

2006 marks our 10th year of  
selling CAN interfaces in over 30  
countries!

**\$249**

USD.



*Zanthic Technologies Inc.*

403-526-8318

[www.zanthic.com](http://www.zanthic.com)

**ALL  
ELECTRONICS**

C O R P O R A T I O N

Electronic and Electro-mechanical  
Devices, Parts and Supplies.

Wall Transformers, Alarms, Fuses,  
Relays, Opto Electronics, Knobs,  
Video Accessories, Sirens, Solder  
Accessories, Motors, Heat Sinks,  
Terminal Strips, L.E.D.S., Displays,  
Fans, Solar Cells, Buzzers,  
Batteries, Magnets, Cameras,  
Panel Meters, Switches, Speakers,  
Peltier Devices, and much more....

[www.allelectronics.com](http://www.allelectronics.com)

Free 96 page catalog

1-800-826-5432

# MYLYDIA, INC.

Layout Gerber,  
Prototype Making

**QUICK TURN**

PCB & Turnkey  
@  
The Best Prices

Tel: 800-695-9342

Sales@mylydia.com

WWW.MYLYDIA.COM

## CHINA PCB SUPPLIER

- DIRECT SALE FROM CHINA
- PROTOTYPE TO PRODUCTION

instant online quote  
shopping cart ordering system  
China competitive prices  
free Electrically test



web: <http://www.pcbcart.com>  
email: [sales@pcbcart.com](mailto:sales@pcbcart.com)  
Tel: +86-571-87012818  
Addr: No. 2 Haining Road,  
Hangzhou, P/R China

WWW.PCBCART.COM

## PC/104 PERIPHERALS

SCIDYNE Offers a Full Line of Innovative Modules for PC/104 Applications



**ADIO-104** *This Module Does it all!*

- 16 Analog Inputs
- 8 Analog Outputs
- 24 Digital I/O lines
- 5V Power
- Pulse Accumulator
- Open-Drain Outputs



**DIO96-104**

- 96 Bi-Directional Digital Channels
- Input, Output and Strobed I/O functions
- Uses Familiar 82C55A Chips

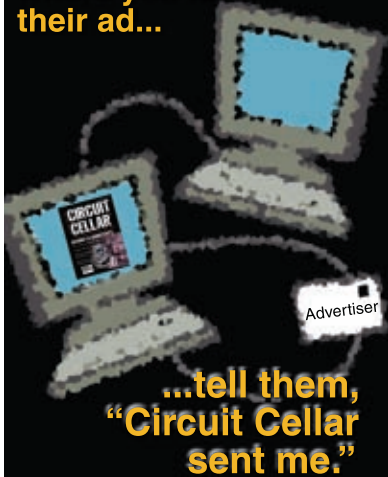
**XIO-RO8**



Add eight High-Power relay outputs to any digital port.

Call or Visit us on the web  
**SCIDYNE**  
1-877-724-3963  
[www.scidyne.com](http://www.scidyne.com)

When contacting our  
advertisers, tell them  
where you found  
their ad...



Advertiser

...tell them,  
"Circuit Cellar  
sent me."

## Instant LCD



inst.LCD-002 - 5189

- Versatile Programmable Module
  - Serial/PC/USB/Parallel Interface
  - AVR/BASIC Stamp/VB Compatible
    - Onboard Flash Bitmap Memory
    - Downloadable TTF Fonts
- 2.7" or 5.6" TOUCH Color TFT LCD
  - 240x160 or 320x240 resolutions
    - Transflective w/ LED Frontlight (2.7")
    - Transmissive w/ 350 nit backlight (5.6")
    - 512 colors or 65,535 colors

EARTH.LCD.COM We Make LCDs Work.™

**PCBFABEXPRESS**  
High Quality PCBs @ Low Impact Prices

Get  
**5 PCBs for  
\$13 each in  
5 days**

2 layer, 20 sq. in., FR-4, 0.062" thick  
FREE Tooling, FREE Soldermask,  
FREE Silkscreen

Visit website to see all offers  
on 2 to 8 layer PCBs

**ORDER ONLINE**  
408-522-1500  
[www.PCBfabExpress.com](http://www.PCBfabExpress.com)

Also see our fantastic  
PCB Assembly prices online

MEASUREMENT SCIENCE  
2007 CONFERENCE

APPLY METROLOGY...  
RULE the WORLD

January 22-26, 2007  
Long Beach Convention Center  
Long Beach, California

GREAT New Venue  
EXCELLENT training opportunities  
MORE than 100 Exhibit Booths (Wed-Fri)  
NETWORK with experts from  
government, industry, academia

Register today!

(866) 672-6327

[www.msc-conf.com](http://www.msc-conf.com)

## 50™ 586-based Industrial Controller

High-performance Ethernet/TCP, 24-bit  
ADC, DAC, HV I/O, and CF Interface.

OEM \$199



- 151x82 mm. DIN rail mounting
- AMD SC520, program in C/C++
- 4 RS232/485, ADC, DAC, Solenoid Drivers, OPTO
- CompactFlash and FAT16 file system support
- Hardware TCP/IP stack for 100M Base-T Ethernet

50+ Low Cost Controllers with ADC, DAC, solenoid drivers, relays, PC-104, CF, LCD, DSP motion control, 10 UARTs, 300 I/Os. Custom board design. Save time and money.

**TERN**  
INC.

1724 Picasso Ave., Suite A, Davis, CA 95618 USA

Tel: 530-758-0180 • Fax: 530-758-0181

[www.tern.com](http://www.tern.com) • [sales@tern.com](mailto:sales@tern.com)



# ValueCAN

The High Value  
Tool For  
Controller  
Area  
Network

- USB to CAN
- Simple software analyzer included
- DLL with examples for custom applications
- PC isolated from CAN
- 100% bandwidth at 500Kb

Only

## \$295



www.intrepidcs.com

8/16-bit EEPROM | Ser. EEPROM | Flash EPROM | GAL / PALCE | Most MCU's | Low Voltage to 1.3V. | DIL dev. w/o Adapter.

Conitec's last generation Galep-4 employs ASIC universal pin technology for each pin of 40 pin ZIF-socket. **9000+device library** / lifetime free updates. Programs 8/16 bit EPROM'S, EEPROM'S, 0-Pwr RAM, FLASH, Serial EEPROM's, GAL, PALCE,

microcontrollers such as 87/89xxx, PIC, AVR, ST62, etc. Low voltage devices down to 1.3V. No adapter required for DIL devices. 8 Hrs. operation on battery (AC charger included). Runs **WIN 98,NT,ME, 2000, XP** with Hex/Fuse Editor.

Remote control from other apps, (e.g. VisualBasic). Substitutes high priced universal programmers e.g. ALL-11 (HILO) or LAB-TOOL-48 (ADVANTECH) **Provides virtually matching performance at 1/3 to 1/5 the price.** Info, orders, softwr : **619-462-0515**



# 9000

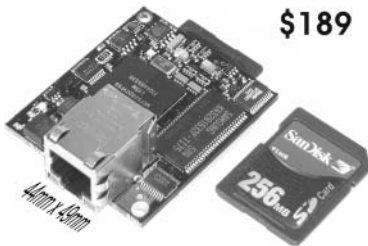
DEVICES



A PALM-SIZED PROGRAMMER HANDLES OVER 9000 DEVICES ?

SMALL PACKAGE. BIG FEATURES.

SALES, SUPPORT / EMAIL: CONTACT@CONITEC.NET / WWW.CONITEC.NET TEL: 619-462-0515. FAX: 619-462-0519



\$189

## Tiny Linux controller

16MB fast SDRAM	10/100 Ethernet
64MHz Coldfire MCU	3 serial ports (RS232)
4.5 MB flash memory	1 CAN port
SD card socket (to 2GB)	LCD/KPD port

Eclipse/CDT dev. env. Free serial debugger

55 I/O pins & capabilities to burn...

www.steroidmicros.com

## PROTOTYPE CIRCUIT BOARDS

**\$85<sup>U</sup>** for two 5" x 6"  
**\$85<sup>S</sup>** 2-layer boards

Shipped **NEXT BUSINESS DAY**  
if data is received by  
**1:00 pm**  
**EASTERN**

www.apcircuits.com

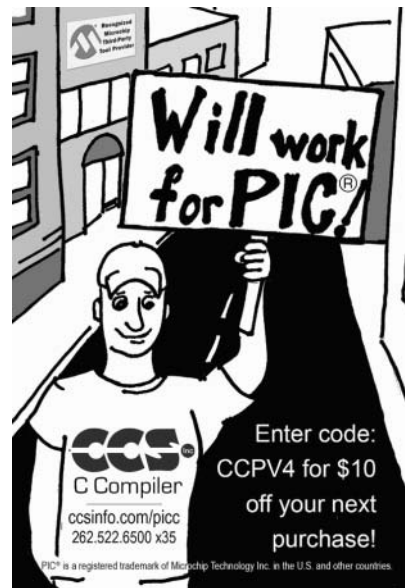


**AP Circuits**

(403) 250-3406



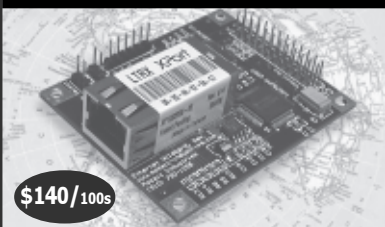
staff@apcircuits.com



Enter code:  
CCPV4 for \$10  
off your next  
purchase!

PIC® is a registered trademark of Microchip Technology Inc. in the U.S. and other countries.

## EtherSmart Wildcard™ Network - Enables Your Product



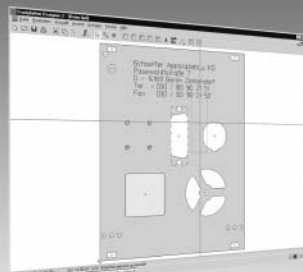
\$140/100s

- Standard RJ45 jack hosts 10/100Mbit Ethernet
- HTTP, SMTP, TCP, DHCP, ICMP, and ARP Protocols
- Email program-controlled messages to a specified LAN IP address
- Establish a TCP/IP connection to exchange binary or ASCII data
- Serve software-controlled dynamic content to your web browser

Mosaic Industries Inc.  
tel: 510-790-1255 fax: 510-790-0925  
www.mosaic-industries.com

## Front Panels?

Download the free Front Panel Designer  
to design your front panels in minutes



Order your front panels online  
and receive them just in time

www.frontpanelexpress.com

Unrivaled in price and  
quality for small orders

**Serial • Ethernet  
Web Server**

**Simple, DTE, DCE  
No SW Changes**

**\$99.<sup>95</sup>  
Qty 1**



+1 630-245-1445  
Naperville, Illinois USA  
[www.gridconnect.com](http://www.gridconnect.com)

**gridconnect.**

**FAT 12/16/32  
FILE SYSTEM**

- DOS/Windows Compatible
- USB Flash Disk
- USB Floppy & Hard Disk
- SD/MMC
- CompactFlash, ATA/IDE
- DiskOnChip
- NAND & NOR Flash
- 10KB RAM / 25KB ROM typical
- Low Cost, No Royalty
- Full Source Code

[www.smxrtos.com](http://www.smxrtos.com)

**Micro Digital Inc**  
RTOS Innovators

800.366.2491 sales@smxrtos.com



**Weather  
Instruments**  
for PCs

**AAG**  
electrónica

[www.aagelectronica.com](http://www.aagelectronica.com)



**High Quality Enclosures  
The Most Competitive Price**

- \* Sheet Metal Cabinets
- \* Aluminium Extruded Enclosures
- \* 19inch Standard Cabinets
- \* Plastic Injection
- \* Precision Aluminium Die Casting
- \* Highly Custom Made
- \* Fast Prototyping
- \* From 1pc to 10,000,000 pcs

[WWW.EzPCB.COM](http://WWW.EzPCB.COM)

**Flashlite  
186**

- 186 processor @ 33 MHz
- DOS w/ Flash File system
- 44 Digital I/O lines w/ CPLD
- Console / Debug Serial Port
- 7-34V DC or 5V DC power • 2 Serial Ports
- Accepts 8MB DiskOnChip • 2 16-bit Timers
- 512K DRAM & 512K Flash • Watchdog Timer
- Expansion options with Peripheral Boards

**\$69  
QTY 1**

**\$99  
Development  
System**

**Development kit includes:**

- Flashlite 186 controller
- Borland C/C++ ver 4.52
- FREE Email Tech Support
- Serial Driver library
- AC Adapter and cable

Call 530-297-6073 Email sales@jkmicro.com  
On the web at [www.jkmicro.com](http://www.jkmicro.com)

**JK microsystems**

**CUSTOM MEMBRANE  
KEYBOARDS / SWITCHES**



- 1 TO 2 WEEKS TURNAROUND
- VERY COMPETITIVE PRICING
- Ex.: (5) 4-switch keyboards for \$395.00
- PCB backed switches
- Custom metal backplates/assemblies
- Electronic assemblies/graphic overlays
- Electronic file transfer capabilities

**Picofab Inc.**

4780B Blvd. Henri-Bourassa  
Charlesbourg, Quebec, Canada G1H 3A7  
Tel: (418) 622-5298 • Fax: (418) 622-9996  
Email: sales@picofab.net



**ANYONE**  
Can Now Easily  
Hand Solder Surface-  
Mount Components!  
**Even A 10  
Year Old!**

[www.schmartboard.com](http://www.schmartboard.com)

**PRINTED CIRCUIT BOARDS**

**QUALITY PRODUCT**  
**FAST DELIVERY**  
**COMPETITIVE PRICING**

- Aluminum Backed PCB
- Single & Double sided
- SMOBC/RoHS
- LPI mask
- Through hole or SMT
- Nickel & Gold Plating
- Routing or scoring
- Electrical Testing
- Artwork or CAD data
- Fast Quotes
- Flex Circuits

**PROTOTYPE  
through  
PRODUCTION**

**SPECIAL OFFER:**  
**10 pcs (3days) 1 or 2 layers \$249**  
**10 pcs (5days) 4 layers \$695**  
(up to 30sq. in. ea.) includes tooling, artwork, L.P.I. mask & legend

**PULSAR, INC**

9901 W. Pacific Ave. Franklin Park, IL 60131 • Phone 847.233.0012  
Fax 847.233.0013 • [www.pulsar-inc.com](http://www.pulsar-inc.com) • sales@pulsar-inc.com



# Full Speed It writes your USB Code!

**NEW! HIDmaker FS for Full Speed FLASH PIC18F4550**

Creates complete PC and Peripheral programs that talk to each other over USB. Ready to compile and run!

- Large data Reports
- 64,000 bytes/sec per Interface
- Easily creates devices with multiple Interfaces, even multiple Identities!
- Automatically does MULTITASKING
- Makes standard or special USB HID devices

**NEW!** "Developers Guide for USB HID Peripherals" shows you how to make devices for special requirements.



DOWNLOAD the HIDmaker FS Test Drive today!

[www.TraceSystemsInc.com](http://www.TraceSystemsInc.com)

301-262-0300

Both PC and Peripheral programs understand your data items (even odd sized ones), and give you convenient variables to handle them.

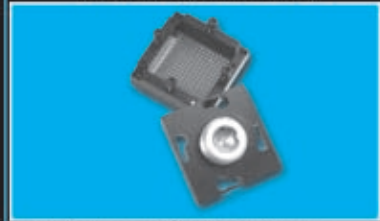
**PIC18F Compilers:** PICBASIC Pro, MPASM, C18, Hi-Tech C.

**PIC16C Compilers:** PICBASIC Pro, MPASM, Hi-Tech C, CCS C.

**PC Compilers:** Delphi, C++ Builder, Visual Basic 6.

HIDmaker FS Combo: Only \$599.95

## GHz Bandwidth Sockets for DSP's in BGA



Sockets for dozens of DSP chips from TI, Analog Devices, and other manufacturers.

- Pitch - 0.3mm to 1.27mm - BGA, MLF (QFN)
- Bandwidth to 10+ GHz
- Smallest Socket Footprint in Industry
- Optional 500,000 Insertions
- Heatsinking to 100 watts

We design custom sockets to your requirements including heat sink option.



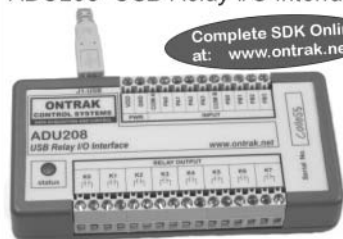
**Ironwood Electronics**

1-800-404-0204

[www.ironwoodelectronics.com](http://www.ironwoodelectronics.com)

## USB Data Acquisition

ADU208 -USB Relay I/O Interface



Complete SDK Online at: [www.ontrak.net](http://www.ontrak.net)

### FEATURES

- 8, 5-AMP relay outputs
- 8, ISOLATED digital inputs
- Port Powered, Aux 5VDC output \$189.00 QTY 1

Other Models:

- ADU200- 4 Channel Version with RS232 \$139.00
- ADR218- Solid-State Version 8 Channel \$225.00
- ADU100- 3 CH, 16-Bit ISOLATED Analog Inputs, PGA, 4 digital I/O, RS232 and 5 AMP Relay Output \$199.00

ONTRAK CONTROL SYSTEMS INC.

PH: (705) 671-2652 Fax: (705) 671-6127

[www.ontrak.net](http://www.ontrak.net)

## JStamp!

FAST 32-bit native-execution Java  
Executes 3 million Java byte codes/sec  
compact 1 x 2 inch DIP40 package  
Real J2ME CLDC Java (threads, floats)  
2048 KB Flash & 512 KB SRAM

[www.jstamp.com](http://www.jstamp.com)

**SYSTRONIX**  
Salt Lake City, Utah, USA

Java is a TM of Sun Microsystems, Inc. JStamp is a TM of Systronix, Inc.

## NEW RS232 to TCP/IP

- TCP/com<sup>tm</sup> v2.0, RS232 to TCP/IP software. Plus TCP/IP to RS232.
- WinWedge<sup>tm</sup>, RS232 or TCP/IP data direct into any Windows app. - Excel, Access, etc.

**TALtech**

Free 30 day evals at  
[www.taltech.com](http://www.taltech.com)

Order online at:  
[www.melabs.com](http://www.melabs.com)

*microEngineering Labs, Inc.*

Development Tools for PIC<sup>®</sup> Microcontrollers

Phone: (719) 520-5323

Fax: (719) 520-1867

Box 60039

Colorado Springs, CO 80960

### USB Programmer for PIC<sup>®</sup> MCUs

**\$119.95**  
(with accessories)

RoHS Compliant

Programs PIC MCUs including low-voltage (3.3V) devices

Includes:

Programmer, USB Cable, ZIF Programming Adapter for 8 to 40-pin DIP, Software for Windows 98/Me/NT/2K/XP



EPIC<sup>™</sup>

**Parallel Port Programmer starting at \$59.95**

**Serial Port Programmer starting at \$79.95**

### LAB-X Experimenter Boards



Pre-Assembled Boards Available for 8, 14, 18, 28, and 40-pin PIC<sup>®</sup> MCUs  
2-line, 20-char LCD Module  
9-pin Serial Port  
Sample Programs  
Full Schematic Diagram

**Pricing from \$79.95 to \$349.95**

### PICPROTO<sup>™</sup> Prototyping Boards



Double-Sided with Plate-Thru Holes  
Circuitry for Power Supply and Clock  
Large Prototype Area  
Boards Available for Most PIC<sup>®</sup> MCUs  
Documentation and Schematic

**Pricing from \$8.95 to \$19.95**

### BASIC Compilers for PICmicro<sup>®</sup>



Easy-To-Use BASIC Commands  
Windows 9x/Me/2K/XP Interface

**PICBASIC<sup>™</sup> Compiler \$99.95**

BASIC Stamp 1 Compatible  
Supports most 14-bit Core PICs  
Built-In Serial Comm Commands

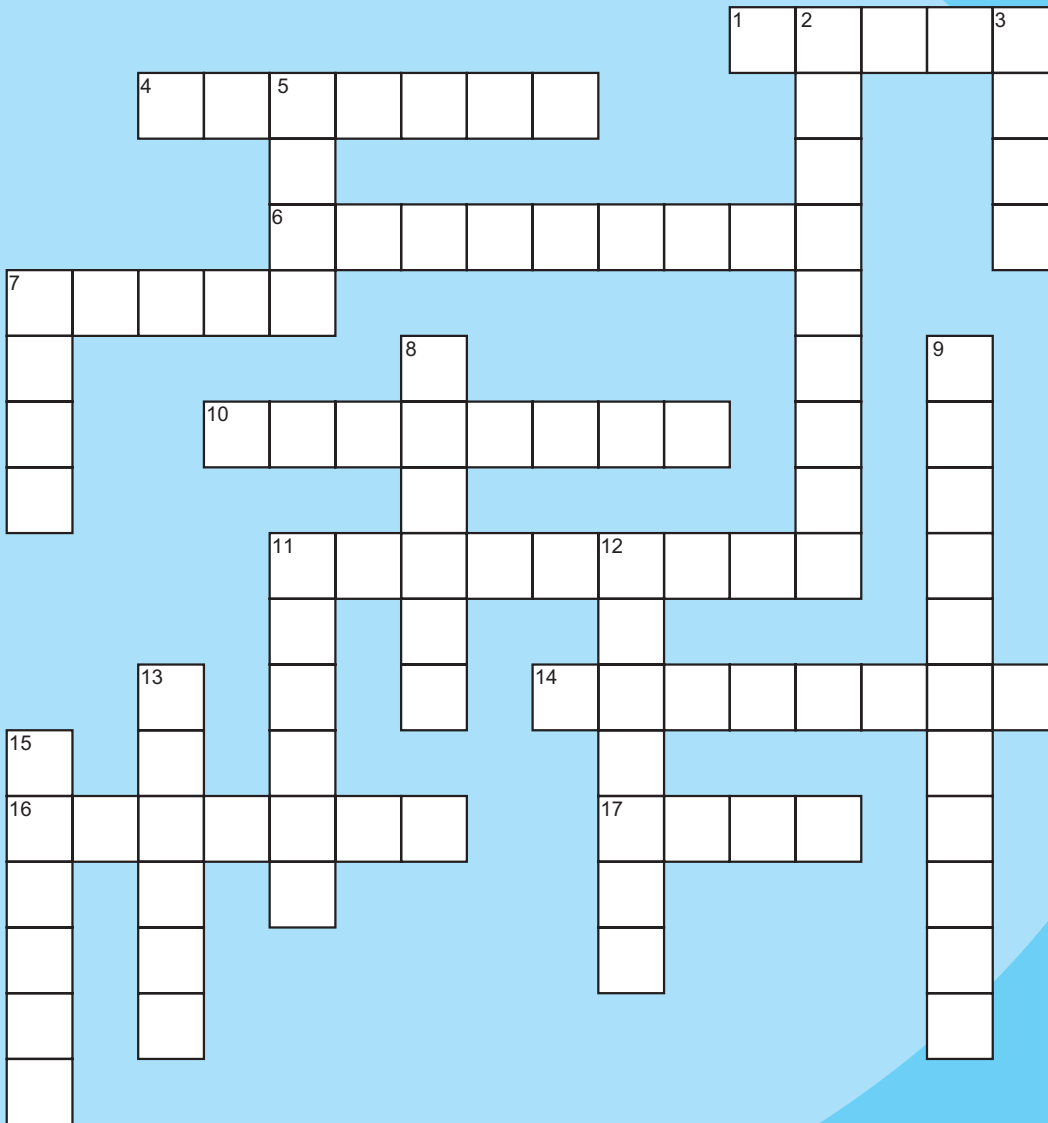
**PICBASIC PRO<sup>™</sup> Compiler \$249.95**

Supports All PICmicro<sup>®</sup> MCU Families  
Direct Access to Internal Registers  
Supports In-Line Assembly Language  
Interrupts in PICBASIC and Assembly  
Built-In USB, I2C, RS-232 and More  
Source Level Debugging

See our full range of products, including books, accessories, and components at:

[www.melabs.com](http://www.melabs.com)

# CROSSWORD



## Across

1. A bad solution to a software or hardware problem
4. To adjust the spacing between letters
6. Free software
7. An exact replica
10. Having a positive and negative pole
11. Regards, Managing Editor, Circuit Cellar, [www.circuitcellar.com](http://www.circuitcellar.com)
14. FF
16. Quicksilver; planet; Roman god (mythology)
17. Text message: "Beam me up, Scottie."

## Down

2. Abbreviation: LC
3. Current
5. @>—;
7. The portion of a spreadsheet that holds data and is identified by its row and column
8. The white space between text and the edge of a document
9. A person online who serves as an intermediary between a vendor and a consumer
11. Ink-less pen
12. Button holder: Save, Cut, Copy, Paste ...
13. A series of objects (e.g., characters)
15. The "A" in VA

The answers are available at  
[www.circuitcellar.com/crossword](http://www.circuitcellar.com/crossword).

# INDEX OF ADVERTISERS

The Index of Advertisers with links to their web sites is located at [www.circuitcellar.com](http://www.circuitcellar.com) under the current issue.

Page		Page		Page		Page	
91	AAG Electronica, LLC	86	FDI-Future Designs, Inc.	89	MSC Conference	5	Renesas
90	AP Circuits	90	Front Panel Express, LLC	34	MVS	91	Schmartboard
64	APEC	83	Futurlec	9	Maxstream	89	Scidyne Corp.
88	All Electronics Corp.	89	General Circuits, Inc.	86, 91	Micro Digital, Inc.	63	SEGGER Microcontroller Sys. LLC
87	Apex Embedded Systems	88, 91	Grid Connect	92	microEngineering Labs, Inc.	10	Sierra Proto Express
8	Arcom Control Systems	95	HI-TECH Software LLC	90	Mosaic Industries, Inc.	92	Systronix
14	Arcturus Networks	55	IPC (Printed Circuits Expo)	71	Mouser Electronics	92	TAL Technologies
7, 33	Atmel	87	IMAGEcraft	89	Mylydia, Inc.	C3	Tech Tools
66	Bitscope Designs	90	Intec Automation, Inc.	C2	NetBurner	72, 73	Technologic Systems
74	CTIA Wireless	90	Intrepid Control Systems	42	Nurve Networks LLC	88	Technological Arts
65	CWAV	25	Intronix Test Instruments, Inc.	92	Ontrak Control Systems	45	TEK Industries, Inc.
23	CadSoft Computer, Inc.	92	Ironwood Electronics	86	Oztronics	89	Tern, Inc.
90	Conitec	64, 91	JK microsystems, Inc.	89	PCB Fab Express	9	Tianma Microelectronics
90	Custom Computer Services, Inc.	11	Jameco	15	PCB-Pool	32	Tibbo Technology, Inc.
1	Cypress MicroSystems	85	Jeffrey Kerr, LLC	C4	Parallax, Inc.	92	Trace Systems, Inc.
88	DIPmicro Electronics	18, 81	Keil Software	86	Phytec America LLC	16	Tri-M Systems, Inc.
86	DLP Design	42	LabJack Corp.	91	Picofab Inc.	88	Triangle Reasearch Int'l, Inc.
85	Decade Engineering	42	Lakeview Research	87	Pioneer Automation, Inc.	88	Zanthic Technologies, Inc.
82	DesignCon West	22	Lantronix	88	Pioneer Hill Software		
45	EMAC, Inc.	87	Lawicel AB	39	Pololu Corp.		
89	Earth Computer Technologies	80	Lemos International	86	Pro-Tech Electronics Canada		
10	Elprotronic	2	Link Instruments	91	Pulsar, Inc.		
88	eMicros	83	Linx Technologies	3, 18	Rabbit Semiconductor		
19	ExpressPCB	17	Luminary Micro	87	Rabbit Semiconductor		
91	ezPCB	87	MCC	87	Reach Technology, Inc.		

## Preview of February Issue 199 Theme: Wireless Communications

**Modular Wireless Tracking System**

**RFID Security System**

**DSP-Based Vehicle Monitoring**

**Wireless Firmware Update**

**Universal Modbus Simulator (Part 1): Theory and Preparation**

**Nixie Tube Propeller Clock**

**Build a Reflow Oven Controller**

**ABOVE THE GROUND PLANE Battery Capacity: Discharge**

**APPLIED PCs Embedded Capacitive Touch Applications**

**FROM THE BENCH Electric Movement and Control**

**SILICON UPDATE Traveling Man**

## ATTENTION ADVERTISERS

### March Issue 200 Deadlines

Space Close: Jan. 12  
Material Close: Jan. 22

### Theme: Robotics

**BONUS DISTRIBUTION:**  
**CTIA Wireless,**  
**Trinity College Robotics Contest**

Call Shannon Barraclough  
now to reserve your space!

**860.872.3064**

e-mail: [shannon@circuitcellar.com](mailto:shannon@circuitcellar.com)



# We make chips think

HI-TECH Software delivers the industry's most reliable embedded software development tools and compilers for over 12 different 8-bit, 16-bit, 32-bit, and DSP chip architectures!



As one of the top five compiler vendors with the largest variety of supported chip architectures, HI-TECH Software's product range is renowned for delivering cutting-edge technology and robust results for development teams worldwide.

With over two decades of industry experience, our long-term relationships with leading chip manufacturers ensure that our products are tightly attuned to new technological releases.

Whichever processor family you are targeting, HI-TECH Software's C compilers can help you write better code and bring it to market faster.

HI-TECH PICC™ Enterprise Edition

HI-TECH PICC™

HI-TECH PICC-18™

HI-TECH dsPICC™

HI-TECH C® for ARM®

HI-TECH C® for 8051

HI-TECH C® for MSP430

HI-TECH C® for HOLTEK MCU

HI-TECH C® for ARCLite™

HI-TECH C® for XA

HI-TECH C® for Z80

HI-TECH C® for H8/300

HI-TECH C® for 68HC11

To see how our compilers can improve your productivity, download a demo now at [www.htsoft.com/downloads/demos.php](http://www.htsoft.com/downloads/demos.php).

HI-TECH C® is a registered trademark of HI-TECH Software. HI-TECH PICC™, HI-TECH PICC-18™ and HI-TECH dsPICC™ are licensed exclusively to HI-TECH Software by Microchip Technology Inc. All other trademarks and registered trademarks are the property of their respective owners.



# PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

## For Want of a Paper Trail

**F**or all the computing power available today, it's ironic that the ability to archive information for the long term was accomplished better at the advent of the printing age hundreds of years ago than it is today. I'm discovering, and perhaps you are too, that there are some real downsides to our fast-moving technology.

Recently, I was digging through my old article archives and I had an interesting revelation. The real Circuit Cellar at my house is the ultimate archive (some affectionately call it the museum). It's truly not as messy as I humorously alluded, but it is definitely full of history since the hardware on the shelves spans 40 years of technology. The last time I was digging through some of the piles I even found a couple 8008 processors along with some Raytheon CK722 transistors from the late '60s. More importantly, I actually discovered an original printed 8008 processor manual in the pile.

As for the old articles, they were all in file folders that contained the printed manuscripts, 35-mm picture slides, *BYTE* author proofs, and the original Word Star files on 5" or 8" floppy disks. Interestingly, I also found the folder for the first issue of *Circuit Cellar* back in 1987, and it was a similar story. The folder contained a few floppy disks (I haven't a clue which word processor it was), some slides, and a pile of printed article proofs. The irony of all of this is that while the history of my endeavors are always tuned to presenting the latest technology, today I have to actually use the oldest and least sophisticated technology if I want to see what I said back then. I have no computer or software system in the Circuit Cellar that can read these old disks.

In fact, if I hadn't already transferred things to newer-generation storage, apparently I wouldn't be able to electronically view any original medium much more than about 10 years old. If it weren't for the printed magazines and developed photographs in these folders, it would all be useless landfill.

Digital archiving is a complex process and a significant problem. It's one thing to save the physical disks, tapes, and drives that hold your data, but you also need to make sure those media are compatible with the hardware and the software of the future in order to recover them. Unlike raw language on a simple piece of paper that requires only raw intelligence to decipher, digital stuff is always encoded and formatted such that both media-specific hardware and unique decoding software are necessary to render it in a form that we can see. If the hardware to read it or the software to decrypt it becomes obsolete, we're in a whole lot of trouble.

Media obsolescence isn't a new phenomenon. It just seems that the changes from an analog to a digital world have made extending the life of an archive into an exceedingly more complex problem. Throughout the past, preserving information for posterity was simply a matter of physical storage—stashing photographs and printed documents in a secure place. As we have evolved to a digital life, where incompatible coding from generation to generation seems to be the norm rather than the exception, long-term non-obsolescent data retention requires us to keep massaging the old data into each new technology. Unless someone invents the ultimate hardware/software emulator, people better be shrink-wrapping laser disk players, old PCs, CD players, and whatever for future use if they aren't converting files and restoring them periodically. That, or throw it all away when the old hardware deep-sixes itself.

The problem is even greater in business and government. Historically, these entities have created vast paper trails memorializing everything from orders for paper clips to specification documents on every item in the inventory and even the menus served at board meetings. In the 1980s, computers replaced typing pools and file clerks. Carbon copies were gradually replaced by perishable e-mails, cryptic PowerPoint slides, and transient web sites that could be deleted instantly. Worse yet, think about schematic programs and CAD systems. Are you still using the same programs you used 10 years ago, or are the files from those older programs compatible with the ones you use today? It's one thing to update files from old piles of PC floppies to a DVD. It's quite another to make sure that you can even print the schematic of a design you did 10 years ago with today's software.

I don't have a good answer for this dilemma other than to say that we may be a generation with no history to share with future generations if we don't solve it. Our history will simply evaporate. Apparently, we don't care about the hundred laser disks we discard when purchasing a new networked HDTV system because we either purchase the same moves again on DVD or "subscribe" to a commercial audio/video provider who maintains the real audio/movie/TV archive and simply transmits these materials in a format compatible with our latest hardware iteration. And, I suppose we don't care about finding a 10-year-old schematic because the hardware is already obsolete.

If I sold CD-ROMs for a living, I'd be pointing out the availability of gold-plated CD-ROMs that guarantee a 25-year retention rather than describing all this doom and gloom. Of course, no one can guarantee that we'll even have a CD-ROM reader on any computer 25 years from now any more than he could have predicted 8-GB USB flash drives half the size of a car key 10 years ago. It's pretty much safe to say that storage technology, and the necessity to keep moving archives to updated media, will remain dynamic. The bad news is that unless we eventually have some nonproprietary formats that everyone uses or can emulate all past data coding formats, we're still destined to lose most of it to obsolete software.

In the meantime, here at *Circuit Cellar*, we'll just keep plugging along with one foot in each world. For the people who think that reading *Circuit Cellar* on their cell phone display is fun, we'll continue to have an electronic version. For the rest of us who like the idea that someone might look at a printed copy of *Circuit Cellar* 200 years from now and know exactly who we are, we'll stick with some paper and ink, too.

steve.ciarcia@circuitcellar.com

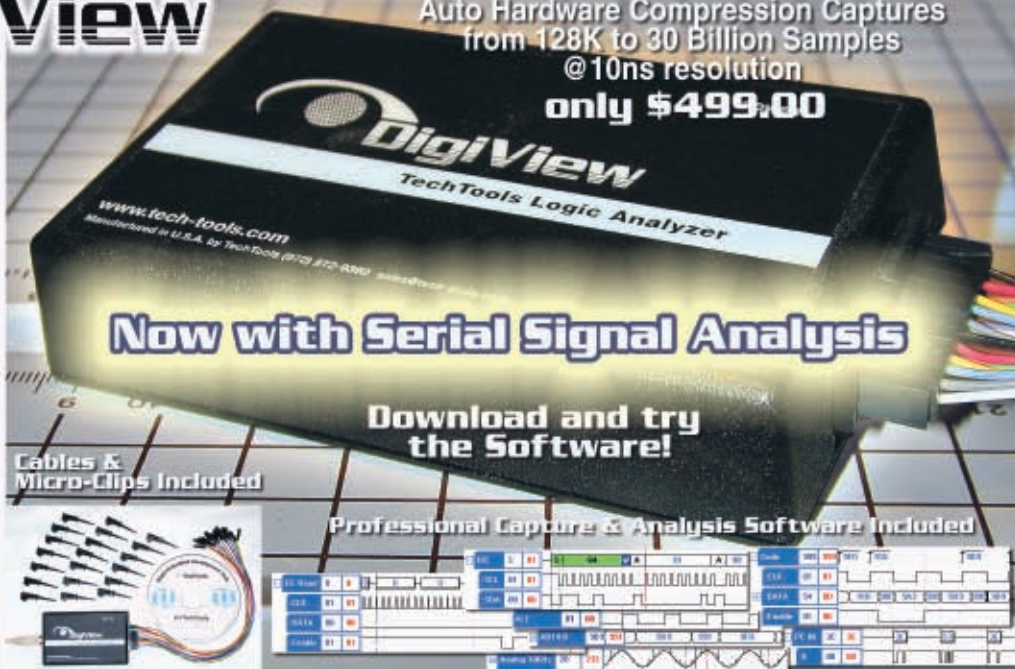


# 100 MHz, 18 Channel, Portable Logic Analyzer

Auto Hardware Compression Captures  
from 128K to 30 Billion Samples  
@ 10ns resolution

**only \$499.00**

- Automatic Real-time Hardware Compression can save up to 5 minutes of data @ 10ns.
- Display I2C, Synchronous (SPI), Asynchronous (RS-232), State, Boolean, Bus and Analog Data.
- Edge and Pattern Triggers.
- Pattern Searches with Match, Duration & Animation.
- Specialized Sequential Searches for Serial and State Mode Signals.
- Drag & Snap Markers.
- Dual Waveform View with signal edge snapping.
- Multi-Signal Data Tables with Link Groups.
- Specialized Exports and List Views for Serial Signals.
- Print or Save Images with custom comments.
- USB Powered for Portability.
- USB 1.1 & 2.0 compatible.



**Now with Serial Signal Analysis**

**Download and try the Software!**

Cables & Micro-Clips Included

Professional Capture & Analysis Software Included

## EconoROM III™



**EPROM & FLASH Emulation at its best!**  
**from \$179.00**

## FlexROM III™



[www.tech-tools.com](http://www.tech-tools.com)

(972) 272-9392 • [sales@tech-tools.com](mailto:sales@tech-tools.com)

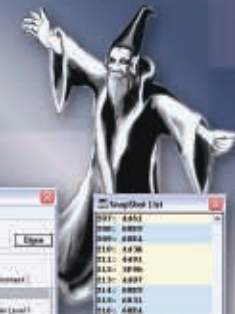
Copyright © 2006 TechTools • DigiView, FlexROM, EconoROM and QuickWriter are trademarks of TechTools • PICmicro is a registered trademark of Microchip Technology Inc.

## PICmicro® MCU Programmer

Multi-Function In-Circuit & Gang Operation

**only \$199.00**

- Supports 10, 12, 14, 16 & 18F Series PICmicro MCUs.
- Edit Code, EE Data, IDs and Configuration Fuses.
- Auto or Manual Serialization in Code or EE Data.
- Firmware auto synchronizes to Software version.
- Control Files protect Option Settings.
- Accepts Command-Line Parameters.
- Tests and Monitors Voltages.
- Control Signal for Self-Powered Circuits.
- Single and 4 Gang Adapters available.
- Includes Cables, Software & U.S. Power Supply.



# Remember when programming was fun?



**Tired of what conventional chips have to offer? Ready to explore the possibilities without rules?** With eight 32-bit processors and real simultaneous multi-processing, the Propeller™ chip will take you back to when programming was seriously fun, instead of just serious.

Propeller Chip Specifications	
Power Requirements	500 $\mu$ A @ 3.3V @ 3333 volts DC
External Clock Speed	DC to 80 MHz (4 MHz to 8 MHz with clock PLL running)
Internal RC Oscillator	12 MHz or 20 kHz
System Clock Speed	DC to 80 MHz
COGs	8
Performance	20 MIPS per COG @ 80 MHz
Global RAM/ROM	32 KB RAM / 32 KB ROM
Processor RAM	512 x 32 per COG
I/O Pins	32
Current Source/Sink per I/O	30 mA



The **PropSTICK USB** (#32210; \$79.95) from Parallax is a complete Propeller prototyping system, closely mimicking the pinout of the DIP Propeller chip. Great for breadboarding or socketed use.

Find the PropSTICK USB and many other Propeller chips, accessories, programming kits, objects, and free downloads at our website.



**PARALLAX**  
www.parallax.com