#176 March 2005

www.circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

EMBEDDED PROGRAMMING

gEDA Overview

Portable FAT Library

TDD Solution for Writing Software

Reinforced Keyless Entry System



03>
7 25274 75349 9
$4.95 U.S. ($5.95 Canada)

NetBurner Mod 5270

Do the Math!



95 MHz

+95 MIPs

+32 bits

$59 *Priced per 1000*

NetBurner

Networking in 1 day!

1-800-695-6828 • www.netburner.com

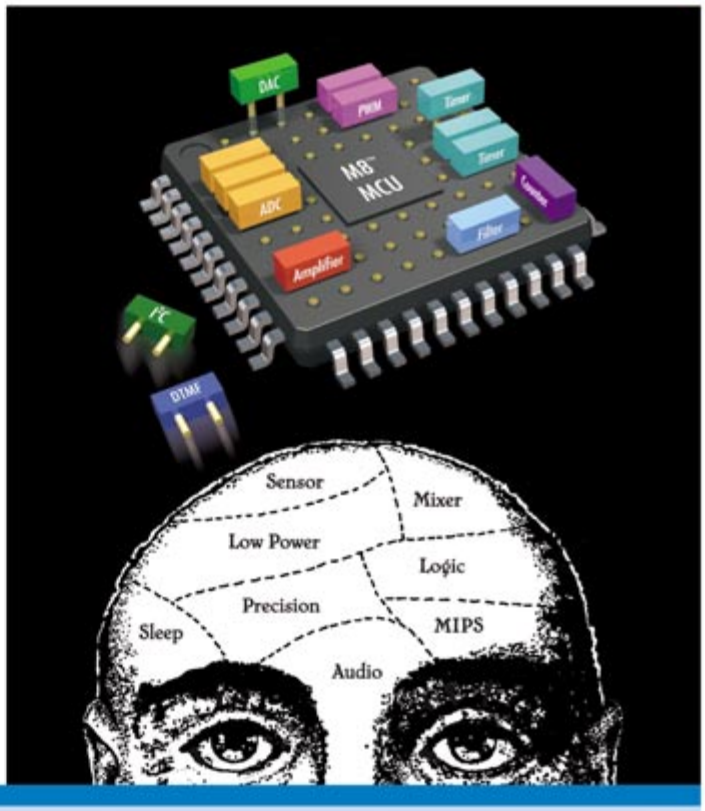
30% OFF
All PSoC Items in Cypress Online Store
www.cypress.com/ad/psoc-cea1

PSoC™ Mixed-Signal Array.
It ll change the way you think about embedded design.

Now with instrumentation-quality programmable analog.

Powerful new programmable analog and digital blocks with memory and MCU for less than $2. Winner of the EDN Innovation of the Year award, our PSoC™ Programmable System-on-Chip™ device is changing the face of embedded design.

- Replace 1,000s of fixed-function devices with a couple of keystrokes
- Dynamically reconfigure a PSoC device, changing functionality on the fly in any application
- Select from hundreds of predefined blocks in our mixed-signal library
- Already designed into 1,000s of applications; check out our online app note library



Reduce board size and BOM up to 80%

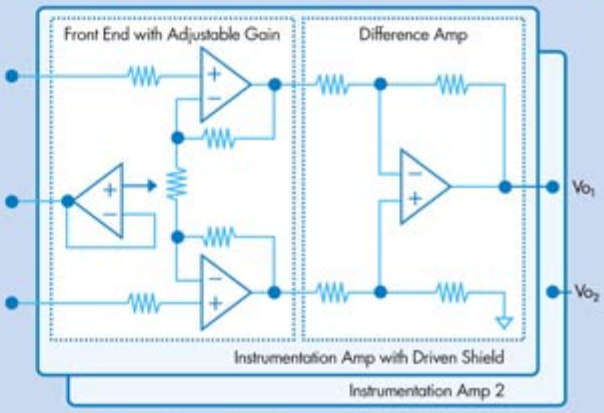


PSoC Mixed-Signal Arrays with M8 Microcontroller

	CY8C27X	CY8C24X	CY8C22X
CEA ANALOG BLOCKS	12	6	3
DIGITAL BLOCKS	8	4	4
HI REL SONOS FLASH	16K	4K	2K
SRAM	256	256	256
COST	low as $1.99	low as 99¢	low as 69¢

CYPRESS ENHANCED ANALOG™ (CEA™)

• Rail-to-rail analog • Instrumentation amps • Lower voltage offset
• Lower input leakage currents • Programmable gains • Better stability



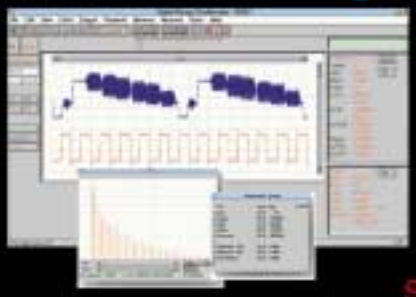
One of 1,000s of examples of programmable analog blocks.

Check out our free online training and our 4-hour applications support:
www.cypress.com/ad/psoc-cea1

CYPRESS MICROSYSTEMS
A Subsidiary of Cypress Semiconductor

©Cypress Microsystems, Inc. 2003. PSoC, Programmable System-on-Chip, Cypress Enhanced Analog and CEA are trademarks of Cypress Microsystems, Inc. All other trademarks are the property of their respective owners.

Logic Analyzers



NEW!

New LA-5000 Series

More memory and USB 2.0 support



- 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)	**$1700** USB 2.0/Parallel
LA5280 (200MHz, 80CH)	**$2350** USB 2.0/Parallel
LA5540 (500MHz, 40CH)	**$2500** USB 2.0/Parallel
LA5580 (500MHz, 80CH)	**$3500** USB 2.0/Parallel
LA55160 (500MHz, 160CH)	**$7500** USB 2.0/Parallel

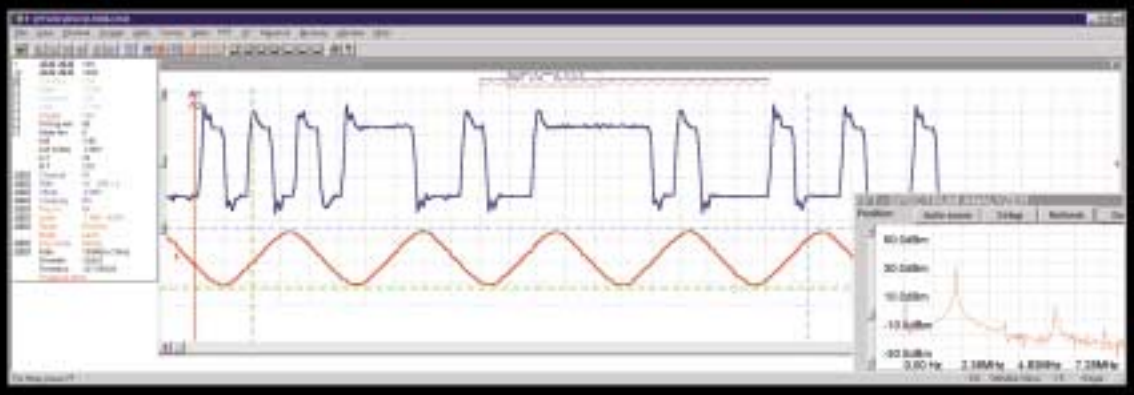
All prices include Pods and Software

Small and portable LA-2124

- 24 Channel Logic Analyzer
- 100MSa/S max sample rate
- Variable Threshold Voltage
- Large 128k Buffer
- Small, Lightweight and Portable
- Only 4 oz and 4.75" x 2.75" x 1"
- Parallel Port Interface to PC
- Trigger Out
- Windows Software

LA2124-128K (100MSa/s, 24CH) Clips, Wires, Interface Cable, AC Adapter and Software **$800**

Digital Oscilloscopes



- 2 Channel Digital Oscilloscope
- **100 MSa/s** max single shot rate
- 32K samples per channel
- Advanced Triggering
- Only 9 oz and 6.3" x 3.75" x 1.25"
- Small, Lightweight, and Portable
- **USB or Parallel Port** interface
- Advanced Math
- FFT Spectrum Analyzer optional



DSO-2102S	$525
DSO-2102M	$650
DSO-2102S(USB)	$600
DSO-2102M(USB)	$725

All prices include **Oscilloscope**, Probes, Interface Cable, Power Adapter and Software

Link Instruments (973) 808-8990
17A Daniel Road East • Fairfield, NJ 07004 • Fax (973) 808-8786
www.Linkins4.com

Seetron Serial LCDs

...almost too easy!

LCDs have a well-deserved reputation for being a pain in the neck to interface. Their non-standard parallel busses and weird timing are enough to turn your hair gray—if you don't pull it out first. Our LCDs have an easy-to-program serial interface (up to 9600 bps) that makes them a snap to use, particularly with friendly micros like BASIC Stamps®.

• *Easy hookup and mounting.* We don't stop at simplifying display interfacing and programming. For easy, quick connections, we offer a variety of ready-made wiring harnesses. Need mounting hardware? We've got that, too.

• *In stock right now.* Hate to wait? Our products are in stock and ready to ship the day you place your order.

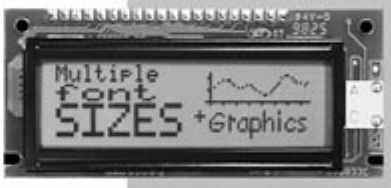
• *Online satisfaction.* Our customers love our web site—it's fast, uncluttered, and jammed with tech data on our products. The online store runs weekly sales with savings up to 40%. Checkout is a breeze with a simple one-page form.



BPI-216N
2x16 serial LCD
ultra-low power
45.00



ILM-216L
2x16 serial LCD
w/ switch inputs,
config EEPROM
49.00



SGX-120L
120x32-pixel
serial graphics LCD
99.00



TRM-425L
4x20 serial LCD
w/ keypad input,
95-screen memory
119.00

www.seetron.com

Many more models available at seetron.com

Scott Edwards Electronics, Inc.
1939 S. Frontage Rd. #F, Sierra Vista, AZ 85635
phone 520-459-4802 • fax 520-459-0623
web www.seetron.com • email sales@seetron.com

TASK MANAGER

Stay Under Budget

So much of what engineers are able to do hinges on cost—the cost of hardware, software, design tools, testing equipment, and so on. When you consider building a new application, sometimes the potential cost can inhibit you from even starting. We're featuring a number of articles this month for those of you who are budget-conscious (that's everyone, right?).

A trio of engineers came to us recently with their particular problem: How do you turn your circuits into PCBs and chips if you can't afford electronic design automation (EDA) tools? Some people benefit from their employers' deep pockets and get to work with expensive commercial EDA tools. Many others simply do without. Stuart Brorson, Ales Hvezda, and Dan McMahill sought a solution for professional-quality work that anyone could afford. All three are part of an international effort started by Ales to establish an array of open-source EDA tool suites. In their article this month, Stuart, Ales, and Dan discuss the gEDA (GPL license EDA) tool suite for Linux (page 12).

Another great project we're featuring this month is a portable FAT library designed by Ivan Sham, William Hue, and Pete Rizun (page 18). They wanted to be able to write files with a microcontroller that could be read on a PC with a standard low-cost card reader. Additionally, they wanted to be able to read files written by a PC from a flash memory card. Putting their heads together, they built a Texas Instruments MSP430 MCU-based portable FAT library that's notable for its convenience.

Cost-effectiveness was also a factor for David Brown when he designed his award-winning entry for the Atmel AVR 2004 Design Contest (page 52). The AT90S2313 microcontroller-based Stealth Keyless Entry System is a reinforced security system that eliminates the traditional 10-button keypad. Over time the buttons on a keypad get worn down and dirty, which makes it easier for an intruder to determine your entry code. David designed a clever scheme that uses Morse code taps on a regular doorbell instead of the keypad to control entry. To ensure the system is user-friendly, David designed it so that users have to memorize only a few Morse code letters. The Stealth Keyless Entry System is a suitable alternative to keypad use in homes and offices.

In his column this month, Jeff Bachiochi discusses how to incorporate an analog joystick in your applications (page 72). This project is designed around Microchip PIC microcontrollers and the E-Switch JS1100AQ joystick navigation switch. Jeff points out design choices that will help you keep your project cost within a reasonable budget.

Fred Eady also guides you through design decisions that could make or break your budget (page 66). This month, Fred reviews Freescale's MC9S12NE64 development kit, and shows you that a single-IC 10/100 Ethernet solution is possible.

Cost is almost always a factor when starting a new project. What we've tried to do is provide you with some naturally low-cost projects plus sensible alternatives to expensive hardware and software so that you can keep building *and* stay under budget.

jennifer.huber@circuitcellar.com



CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

EDITORIAL DIRECTOR/FOUNDER
Steve Ciarcia

MANAGING EDITOR
Jennifer Huber

TECHNICAL EDITOR
C.J. Abate

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Ingo Cyliax
Fred Eady
George Martin
George Novacek
Jeff Bachiochi

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Steve Bedford
Ken Davidson
David Tweed

CHIEF FINANCIAL OFFICER
Jeannette Ciarcia

CUSTOMER SERVICE
Elaine Johnston

CONTROLLER
Jeff Yanco

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNER
Mary Turek

STAFF ENGINEER
John Gorsky

QUIZ COORDINATOR
David Tweed

ADVERTISING

PUBLISHER
Dan Rodrigues E-mail: dan@circuitcellar.com

ASSOCIATE PUBLISHER/DIRECTOR OF SALES
Sean Donnelly Fax: (860) 871-0411
(860) 872-3064 E-mail: sean@circuitcellar.com
Cell phone: (860) 930-4326

ADVERTISING COORDINATOR
Valerie Luster Fax: (860) 871-0411
(860) 875-2199 E-mail: val.luster@circuitcellar.com

ADVERTISING ASSISTANT
Deborah Lavoie Fax: (860) 871-0411
(860) 875-2199 E-mail: debbie.lavoie@circuitcellar.com

Cover photograph Chris Rakoczy—Rakoczy Photography
www.rakoczyphoto.com

PRINTED IN THE UNITED STATES

CONTACTING CIRCUIT CELLAR

SUBSCRIPTIONS:
INFORMATION: www.circuitcellar.com or subscribe@circuitcellar.com
To Subscribe: (800) 269-6301, www.circuitcellar.com/subscribe.htm, or subscribe@circuitcellar.com
PROBLEMS: subscribe@circuitcellar.com

GENERAL INFORMATION:
TELEPHONE: (860) 875-2199 Fax: (860) 871-0411
INTERNET: info@circuitcellar.com, editor@circuitcellar.com, or www.circuitcellar.com
EDITORIAL OFFICES: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066
NEW PRODUCTS: New Products, Circuit Cellar, 4 Park St., Vernon, CT 06066
newproducts@circuitcellar.com

AUTHOR CONTACT:
E-MAIL: Author addresses (when available) are included at the end of each article

For information on authorized reprints of articles, contact Jeannette Ciarcia (860) 875-2199 or e-mail jciarcia@circuitcellar.com.

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) and Circuit Cellar Online are published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (860) 875-2751. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $21.95, Canada/Mexico $31.95, all other countries $49.95. Two-year (24 issues) subscription rate USA and possessions $39.95, Canada/Mexico $55, 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 © 2004 by Circuit Cellar Incorporated. All rights reserved. Circuit Cellar and Circuit Cellar INK are registered trademarks of Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.



Delivering the latest with the greatest.

Efficient and innovative embedded solutions are critical, so is an efficient distributor. As the catalog distributor focused on the needs of engineers, Mouser is an all-in-one package. We not only deliver the latest technologies, but the great service and support you need.

SEMICONDUCTORS | PASSIVES | INTERCONNECTS | POWER | ELECTROMECHANICAL | TEST, TOOLS & SUPPLIES



Supplier Authorized Distributor

New Products,
New Suppliers,
New Technologies,
New Catalog Every 90 Days!

MOUSER ELECTRONICS
a tti company

(800) 346-6873
www.mouser.com

Mouser® and Mouser Electronics® are trademarks of Mouser Electronics, Inc.

March 2005: Embedded Programming

FEATURES

12 **gEDA Design Suite for Linux**
Stuart Brorson, Ales Hvezda, & Dan McMahill

18 **Portable FAT Library for MCU Applications**
Ivan Sham, William Hue, & Pete Rizun

28 **Embedded Security Design (Part 1)**
Product Enclosure
Joe Grand

34 **Practical Application for TDD (Part 1)**
Write Software in a Test-Driven Development Environment
Mike Smith, Moreno Bariffi, Warren Flaman, Adam Geras, Lily Huang, Andrew Kwan, Alan Martin, & James Miller

42 **Zeroing in on ZigBee (Part 2)**
Chipsets and Source Code
Pete Cross

52 **Stealth Keyless Entry System**
David Brown
Atmel AVR 2004 Design Contest Winner

60 **Control an FPGA with I²C**
Roberto Ferrabone



Test the Merits of TDD (p. 34)


Beef Up Your Security System (p. 52)

COLUMNS

66 **APPLIED PCs**
Single-IC 10/100 Ethernet Solution
Fred Eady

72 **FROM THE BENCH**
Joystick Technology
Jeff Bachiochi

78 **SILICON UPDATE**
'51 Favorites
Tom Cantrell


Freescale MC9S12NE64-Based Ethernet Solution (p. 66)


Make Room for an Analog Joystick (p. 72)

DEPARTMENTS

4 **TASK MANAGER**
Stay Under Budget
Jennifer Huber

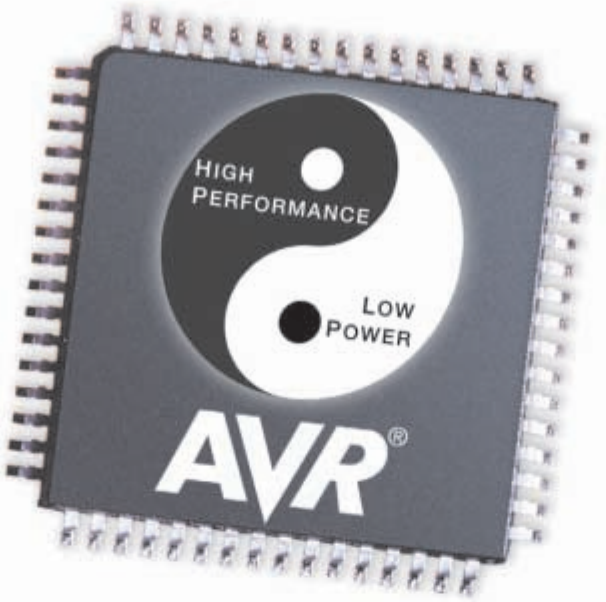
8 **NEW PRODUCT NEWS**
edited by *John Gorsky*

10 **TEST YOUR EQ**
edited by *David Tweed*

94 **INDEX OF ADVERTISERS**
April Preview

96 **PRIORITY INTERRUPT**
Anniversaries
Steve Ciarcia

Come see us at ESC SF
March 8–10, 2005
Booth# 1314



AVR microcontrollers
The perfect combination of performance and power consumption



Specifically designed for the latest battery operated portable applications, Atmel's new Low Power AVR microcontrollers provide six low-power operation modes. These devices work down to 1.8 volts providing maximum battery life. In low power modes such as standby, AVR turns off all internal peripherals and the core. Standby mode leaves only the external crystal running to deliver start-up times as low as 6 clock cycles.

AVR is a RISC CPU that executes instructions in a single clock cycle, with a rich CISC-like instruction set and 32 working registers. AVR microcontrollers have a very high code density and extremely fast execution speeds up to 24 MIPS. The Flash program memory and on-chip EEPROM will help you slash months off your development efforts – save thousands of dollars in project costs, and save YOUR energy as well.



AVR comes in a full selection of package and performance options, and is supported by a full compliment of integrated tools to make your development process fast, simple and successful. Atmel dedicated staff as well as certified technical consultants are available to help and support your projects.

See for yourself: Atmel's Butterfly evaluation kit demonstrates AVR's low power capabilities, and also serves as a development kit for the mega169 microcontroller.

So get your project started off right. Learn more about AVR at www.atmel.com/ad/lowpoweravr, and **register to win a free butterfly evaluation kit.**

See why AVR offers the perfect combination.

Check out AVR today at: www.atmel.com/ad/lowpoweravr

Low Power AVR Typical Power Consumption				
Operating Voltage	Power Down	Real Time Counter	Active 32 kHz	Active 1 MHz
1.8 Volts	Less than 100 nA	4.5 µA	15 µA	240 µA
3 Volts	Less than 250 nA	7 µA	26 µA	420 µA

ATMEL®
Everywhere You Are℠

© 2004 Atmel Corporation. Atmel®, AVR® and the Atmel logo are registered trademarks. "Everywhere You Are℠" is a service mark of Atmel Corporation.

NEW PRODUCT NEWS

Edited by John Gorsky

FLASH MEMORY PROGRAMMING SOFTWARE FOR ColdFire BOARDS

Crossware's new **flash memory programming software** can program Freescale ColdFire boards using the FireFly USB background debug mode (BDM) interface. The FireFly can program flash memory more than eight times faster than via a parallel port interface.

The flash memory software, which is available for free to FireFly interface users, simultaneously controls up to 127 USB ports, each of which are connected to a separate ColdFire board. A single click of the mouse programs the selected software file into any or all of the connected boards. The software can automatically generate a configuration script, which is required so the target board can be correctly set up for flash memory programming, by examining the program that will be downloaded.



Multiple program formats are supported including Motorola S records, IEEE 695, and ELF. The software can be used with programs generated by any ColdFire tool chain, as well as with those generated with the Crossware ColdFire development suite.

FireFly features a USB controller; it doesn't require an external crystal. The miniature debugger plugs directly into the target board, thus minimizing signal delays and ensuring compatibility with the new generation of ColdFire chips running at up to 200 MHz.

The FireFly USB BDM interface is driven by a new version of Crossware's source-level debugging software, which integrates seamlessly with the rest of its ColdFire development suite. The user interface is unchanged, so the USB debugger can replace an existing parallel port interface. In addition, it can be used on platforms such as the newer generation of notebook PCs that don't have parallel ports.

The FireFly USB BDM interface costs **$656**.

Crossware Products
www.crossware.com

AVR LCD MICROCONTROLLERS

The **mega329/649/3290/6490** flash memory microcontrollers are designed to address the requirements of portable and battery-operated applications that need to interface with an LCD. These devices combine a code-efficient and powerful microcontroller with a low-power flash memory technology. The mega329 and mega649 have an integrated 100-segment LCD controller. They are housed in a 64-pin package. The mega3290 and mega6490 have a 160-segment LCD controller. They are available in a 100-pin package.



Members of this family of microcontrollers consume only 300 µA at an operating frequency of 1 MHz. They have a throughput of 2 MIPS per milliwatt. In Power-Down mode, all peripherals, with the exception of external interrupts, are turned off and power consumption drops to 100 nA. These parts are extremely responsive to external events in both Active and Power-Down modes. Interrupts on up to 18 inputs can wake the MCUs from Power-Down mode in less than 1 µs.

The mega329 and 3290 cost **$4.50** and **$4.95**, respectively, in 10,000-piece quantities. The mega649 and mega6490 will be available in early 2005 and cost **$5.25** and **$5.75**, respectively. Evaluation and debug tools are available.

Atmel Corp.
www.atmel.com

ISOLATED ADC WITH PGA AND RELAY OUTPUT

The **ADU100** features three isolated 16-bit analog inputs, four digital I/O lines, one 5-A relay contact output, four 16-bit event counters, and an auxiliary RS-232 port. Two of the analog input lines can be programmed in 20 mV to ±2.5 V ranges via an internal PGA. The third analog input is a high-level input with programmable ranges of 0 to 5, ±5, 0 to 10, and ±10 VDC. Two reference outputs are available (2.5 and 5 V) to allow for the direct connection of strain gauge type sensors.



This low-cost data acquisition interface is easy to use with Visual Basic and Visual C++ via standard HID drivers included with Windows 98, 2000, and XP. A mini-driver (DLL) allows for control using standard ASCII commands using read and write commands. The ADU100 USB data acquisition interface also features an auxiliary RS-232 port for connecting legacy RS-232 devices. The ADU100 includes a 10' USB cable. It's available as a PCB or in an enclosure with several mounting options. The variety of I/O make it well suited for integration into a variety of instruments.

The ADU100 costs **$199**. Three additional USB boards are currently available. Four more will be available by the end of 2005.

Ontrak Control Systems, Inc.
www.ontrak.net



Innovation and creativity could win you a share of $20,000*

The Renesas M16C Design Contest 2005 is the perfect opportunity to work with the Renesas M16C family. Your innovation and creativity could win you a share of $20,000 in cash prizes!*

The Renesas M16C™ MCU platform is the industry's broadest microcontroller product line, offering seamless code, peripheral and pin compatibility from small 20-pin to large 144-pin devices, all within a single common software development environment. The complete line of highly optimized MCUs delivers practical solutions for your applications.

For the official rules and entry form, go to www.circuitcellar.com/renesas2005m16c.

Deadline: June 30, 2005

Everywhere you imagine RENESAS

M16C PLATFORM™

DESIGN CONTEST 2005

*See the web site for complete details.

RENESAS CIRCUIT CELLAR®

CIRCUIT CELLAR

Test Your EQ

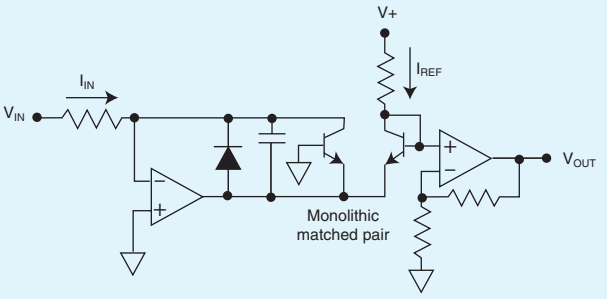
Edited by David Tweed

Problem 1—What was the capacity of the first generation of 8″ floppy disk drives?

Problem 2—What is the structural difference between NAND flash EEPROM and NOR flash EEPROM?

Problem 3—What is the difference in application between NAND flash EEPROM and NOR flash EEPROM?

Problem 4—What is the function of the circuit below?



Contributed by David Tweed

What's your EQ?—The answers are posted at www.circuitcellar.com/eq.htm

You may contact the quizmasters at eq@circuitcellar.com

It gets no easier than this!

SPLat is an innovative new set of solutions that's changing how OEMs world-wide think about embedded controls. With SPLat you can program a quick and dirty sequencer literally in minutes, yet the power is there to make sophisticated state machine based multi-tasking systems with elegant user menus and floating point arithmetic. Our hardware comes with real-world I/O interfaces, ready to wire in, straight out of the box. We have a wide range of special function add-ons for hard to implement tasks like AC current measurement and DC motor control. Best of all, you can do a proof of concept using off-the-shelf product and later migrate all your knowledge and 98% of your code to a fully customized SPLat controller for quantities down to a few hundred. Check out our website today!



Example: SPLat MS12OEM216

- 12 digital I/Os *plus*
- 2x16 LCD, blue backlight
- 5 push buttons
- 4 programmable LEDs
- Expandable

$99 Qty 1

SPLat CONTROLS

Ph. (781) 729-3005

Use the exact URL below for a special offer on your first purchase

www.splatco.com/cc

P&E's Stand-Alone Programmer...



...No Longer Stands Alone

CYCLONE MAX

P&E's Cyclone MAX automated programmer and debug interface is a cost-effective, high-speed, all-in-one solution for your in-circuit programming needs. It is ideal for both development and production, and its ability to function either stand-alone or with a PC gives you incredible flexibility. The Cyclone MAX works with a variety of high-end devices:

ColdFire MCF5xxx | PowerPC MPC5xx/8xx, MPC55xx | ARM MAC71xx

www.pemicro.com

For HC08/HCS08/HC12/HCS12 use P&E's Cyclone PRO!

Need Device Connectivity To The Pervasive Internet?*

Save Money, Save Time , Use The PowerCore



"Based on experience, I can tell you that Rabbit stuff works before you can get it all out of the box."

–Fred Eady, *Circuit Cellar*

The Core of your Embedded System

The PowerCore FLEX is a complete Rabbit microprocessor system with optional features such as a power supply, Ethernet, and a low-cost rugged A/D system. It plugs into a motherboard you design. You pay only for the options you need, and we quickly manufacture it. Development Kits include a powerful software development platform and extensive libraries.

From $29 qty. 5000 **Pay Only For What You Need**

Network and Internet Support

Wi-Fi

Your embedded device can network and serve web pages. Hardware and software supports Ethernet, Wi-Fi, and cellular telephone data networks. TCP/IP and most associated protocols are included in the Development Kit. SSL secure server software and other premium software modules are available at a nominal cost. With the implementation of optional software modules such as RabbitWeb, SSL, PPP, and AES, you can securely and conveniently communicate with your embedded device from anywhere in the world.

Configurable Features

- Microprocessor Speed
- On-Board Power Supply
- SRAM / Flash
- Serial Data Flash
- Ethernet Option
- A/D Converter System
- Battery-Backed SRAM

Build Your Own

Development Kit Includes



- A loaded PowerCore FLEX with all options or non-Ethernet low-end version
- Prototyping board with development area.
- Documentation on CD.
- Dynamic C integrated development environment.
- Serial cable, power supply, and more.

Standard Features

- 39 General Purpose I/O
- 5 Serial Ports
- Real-Time Clock
- Watchdog / Supervisor

* *Pervasive Internet: Embedded devices providing internet connectivity for commercial and industrial applications. Learn more at* ***www.powercoreflex.com***

Buy A Development Kit!

Purchase a complete development kit, PowerCore, and prototyping board with sample circuits.

www.PowerCoreFLEX.com

KITS FROM $129 Limited Time Only

RABBIT Semiconductor

2932 Spafford Street, Davis, CA 95616
Tel **530.757.8400**

gEDA Design Suite for Linux

Wouldn't it be nice to have professional-quality EDA tools on hand the next time you start designing a circuit board? Look no further than the gEDA design suite for Linux. Read on to learn about the free, open-source tools at your disposal.

If you design circuit boards and chips, you're probably familiar with various software tools for circuit design. Such tools fall under the rubric of electronic design automation (EDA). Many companies offer EDA tools that allow you to draw a schematic representation of your design, perform simulations of the design, create netlists of various sorts, perform design rule checking, and lay out your circuit for fabrication as a PCB or chip.

Fluency in using a suite of design tools is an essential part of an engineer's job nowadays. If you're lucky enough to work for a large, well-funded company, you've probably had the luxury of using a high-quality commercial EDA suite costing between $1,000 and $100,000. Smaller companies and individual consultants must either tighten their belts a few notches to buy the expensive software or forgo the advantages of using electronic design tools altogether. This is not an easy choice.

If you're currently facing the dilemma presented by high tool costs and increasing design complexity, don't despair. Linux—and the open-source software movement in general—has come to your rescue!

Engineers and programmers working in loose confederation around the world have succeeded in creating an astounding variety of open-source EDA tools suitable for doing circuit design at a professional level. Commercial-grade (or nearly commercial-grade) open-source tool suites exist for numerous applications, including schematic capture, analog and digital simulation, PCB layout, VLSI layout, and Gerber viewing. Although the quality of open-source EDA tools can vary, the mainstream tools have achieved a high level of usability because they are written by practicing engineers using open-source methodologies. Best of all (at least for your CFO), the tools are free to download and use without any royalties, license fees, or usage limitations. This means that you can try the tools without hassle, and then use them as often as you'd like (and install them on numerous machines) without having to pay user fees or put up with nagware.

gEDA PROJECT

The gEDA project offers perhaps the largest and most mature open-source design suite available for board design. The gEDA project is an umbrella under which many different tools, produced by many different individuals, are gathered into one interoperable design suite.

GPLed electronic design automation, or gEDA, is a name that alludes to the tool suite's license policy (the GPL) as well as its purpose. gEDA is an EDA suite suitable for end-to-end PCB design, from schematic capture to layout.

The GPL license policy ensures that the underlying source code is distributed with the rest of the suite. By distributing the source code, the tool's inner workings remain open and transparent to end users. In practice, you can download the suite from the project web site and compile the source code with one of several automated installation methods. Thereafter, you may not notice or care to notice that the source code is available. However, access to the source means that a power user can always inspect, modify, customize, and improve the code. Easy customization is an important feature of the gEDA suite.

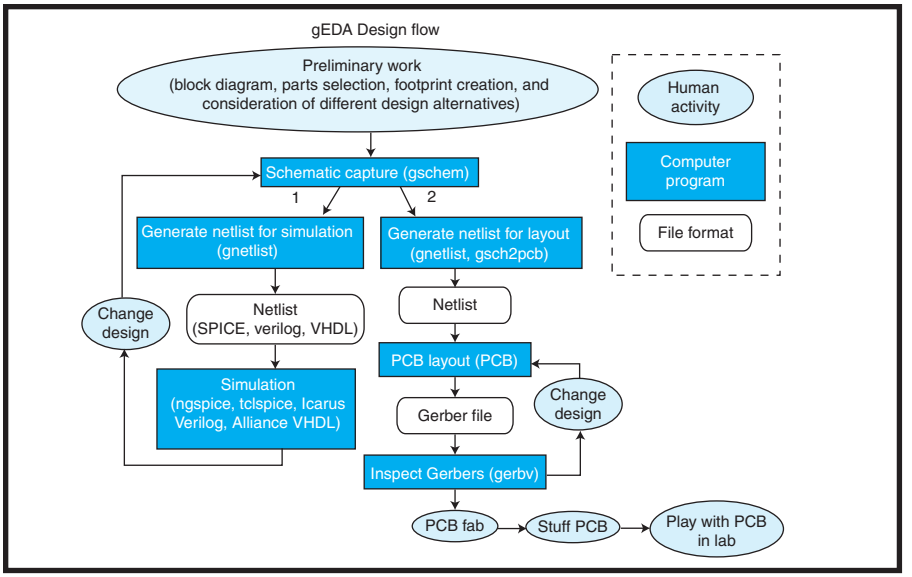


Figure 1—*Take a look at the gEDA design flow. This figure diagrams the process of creating a circuit board as the design is carried through schematic capture, netlisting, simulation, PCB layout, fabrication, and testing in the lab.*

Ales Hvezda started the gEDA project in the summer of 1997. According to Ales, he created gEDA to "scratch the proverbial itch," a reason cited as the genesis of many free software projects. What Ales meant was that he wanted to design circuits using his Linux box, but couldn't because the EDA tools didn't exist. Therefore, he was compelled to write gEDA and then distribute it in the spirit of cooperative collaboration. As a result, the gEDA tools run best on Linux machines. Nevertheless, considerable effort has been made to ensure that they also will run on other Unix variants. Some will run on Windows as well.

The first publicly available version of gEDA was released on April 1, 1998. This software release included only a bare-bones schematic capture program called gschem. From this modest beginning, the gEDA package quickly grew to include a netlister, a symbol library, a symbol checker, and a slew of utilities. As time passed, additional developers contributed to the project and these core tools.

The project also has attracted independent developers working on their own tools who've wanted to contribute to the larger gEDA project. Spurred on by gEDA's success at attracting free EDA projects, Ales renamed the core components of the suite "gschem and friends," or "gaf," leaving gEDA as the moniker for the larger suite of tools. During the last five years, the gEDA suite has grown to the point where it's now worthy of serious consideration if you're looking for a low- to mid-level EDA suite.

gEDA TOOLS

What can you do with the gEDA suite? Perhaps it's most commonly used for designing PCBs with layer counts up to four or six layers, even though gEDA tools also exist for other design tasks.

To better understand gEDA's abilities and uses, take a look at the design flow shown in Figure 1. Several components are frequently used for board design: "geda" (the project manager), "gschem" (the schematic capture program), "gnetlist" (the suite's netlister), "gsch2pcb" (a special forward annotation tool for "PCB"), "PCB" (the board layout program), and "ngspice," which is used for SPICE simulation. With the exception of PCB and ngspice, these tools are all part of gEDA/gaf.

Various other tools exist under the gEDA umbrella, including other simulators (gnucap) as well as programs for digital logic design (Icarus Verilog and GTKWave). Figure 1 shows many of the available tools in the context of gEDA's design flow. You may run the tools from either the Linux command line as stand-alone applications or use the gEDA project manager to maintain your files and run the programs.

GSCHEM

At the front end, the major tool is gschem, which is the schematic capture program that provides all the features you'd expect from a schematic capture program. Use it to input components (e.g., capacitors, resistors, and chips) and connect them with nets and buses. The program is aware of electrical connectivity. Furthermore, it handles attribute attachment, and provides support for hierarchical designs. Photo 1 shows gschem in action.

gschem's easy-to-use GUI allows you to manipulate your design with either pull-down menus or hotkeys. The program is unique because every function has a hotkey equivalent. After you learn the program, you can drive it with one hand on your mouse and the other on your keyboard. This operating style makes working with gschem quick and easy. The key bindings are also configurable, so you can create your own settings if you don't like the default settings.

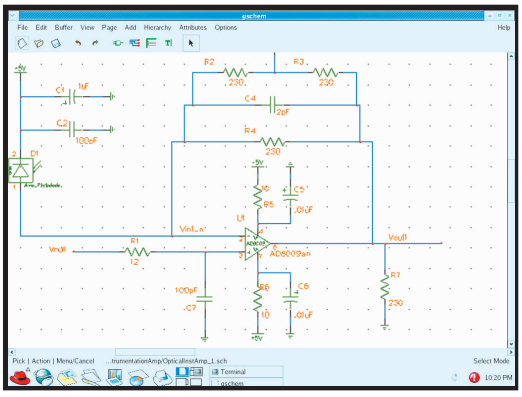


Photo 1—*This is a typical circuit schematic drawn using gschem, which is gEDA's schematic capture tool. gschem is also used to create custom component symbols that may be used in your designs.*

gschem is used for more than capturing schematics; it's also used to create custom component symbols. Its library currently contains approximately 1,400 components, but you typically have to create a few new components for your designs (like you have to do with commercial schematic capture packages).

Creating a component involves drawing the graphical elements, adding the appropriate pins, and adding the necessary attributes. The newly created symbol is then stored as an ASCII file in one of the directories where gschem looks for components. The component symbol is available for use in the schematic as soon as it's placed in the directory.

An extremely important feature of gschem (and the rest of the gEDA tool suite) is that all design files are maintained as ASCII text files. The file format is straightforward, cleanly architected, and well documented. This feature stands in contrast to the way design files are handled by many commercial EDA packages, which often try to achieve vendor lock-in by trapping designs in cryptic binary files and databases.

Cleartext ASCII files are preferable to proprietary binary files for several reasons. ASCII text files are easily created and handled using a scripting language like TCL, Perl, or Python. This enables you to use scripts to automate routine design tasks like creating complex symbols or to perform automated DRC checking. More importantly, you can open your design and inspect it if a strange bug appears. Who hasn't experienced the all-too-common bug where the tool thinks the design looks one way but displays something different? With gEDA you can use a standard text editor to inspect your design file and fix the problem. If your design is held in a proprietary binary file, you're out of luck!

GNETLIST

Following a schematic capture, the gEDA design flow calls for netlisting using gnetlist, which is a command-line program that reads in your schematic and symbol files and writes out a netlist

from your design. The netlist is a textual representation of your circuit. Follow-on programs like a layout program or a circuit simulator use it.

Gnetlist is divided into two parts. The first is a front-end written in C language that reads in your design and creates an internal representation of its connectivity. The second part is a back-end written in Scheme language that converts the internal representation into a text netlist suitable for output. The advantage of the divided architecture is that you can easily write new back-ends to handle new netlist formats. You also can modify and improve existing netlisters without recompiling gnetlist. Depending on the complexities of the target netlist format, you can write a new netlist back-end in a matter of hours.

Only four output netlist formats were supported in the first few gEDA releases. But thanks to great community interest and involvement, more than 20 different netlist types are currently supported. Several common commercial layout netlists are supported (PADS, Allegro, and Protel-II) as well as the gEDA project's PCB. Also, netlists used in simulation may be generated, including SPICE, Verilog, VHDL, and RFCascade. (Refer to the "SPICE Your Design" sidebar for more information.) Because programming a new back-end is relatively simple, new netlisters for other netlist formats are introduced on a regular basis.

PCB

After a layout netlist is created, it's on to the PCB layout. Although gnetlist supports numerous commercial layout netlist formats, the gEDA project has an official layout tool called PCB. Interestingly, PCB has been around much longer than the rest of the gEDA suite. It was originally written about 20 years ago for the Atari computer! It was ported to commercial UNIX approximately 10 years later. Most recently, it was adopted by the gEDA community and put on the SourceForge web site to facilitate further development. Although it wasn't originally written by the same authors as gschem and gnetlist, PCB has become so popular with gEDA users that special tools (e.g., gsch2pcb) have been written to facilitate the process of exporting and annotating gschem schematics to PCB.

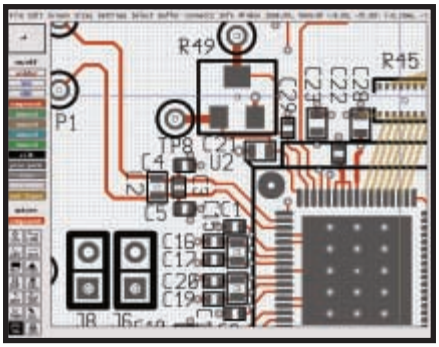


Photo 2—*Here's PCB in action. The tool palette is on the left. The layout under design is visible in the main window. A large TQFP is being routed on the lower right.*

Reflecting its longevity, PCB's GUI still uses the Athena widget set. Furthermore, its behavior is modal, which means the user interface behaves differently depending on the program's mode. This kind of interface behavior may be unfamiliar if you're used to modern GUIs, but it will feel natural after you learn it and put in some practice.

Photo 2 shows PCB in action. PCB provides all features expected from a layout tool. It currently contains nearly 2,000 elements in its footprint library, which is continually expanded and verified through contributions from the large user/developer community. Standard through-hole and surface-mount packages such as DIP, SIP, SO, QFN, and QFP are well represented. PCB supports both manual and autorouting. In addition, it includes a trace optimizer that can optimize the autorouted layout. As with any good layout tool, you can manually route critical analog traces and let the autorouter handle the digital section of a board.

After the layout is finished, PCB tells you that all the connections have been made and no shorts exist. PCB can also perform several design rule checks. For output, it can generate postscript for hard copy check plots, as well as Gerber output for board fabrication. Prior to ordering boards, you can preview the Gerber files with gerbv, which is an open-source Gerber viewer that's part of the gEDA project.

UTILITIES

Besides the core tools used for PCB creation, the gEDA suite incorporates a number of handy utilities that assist with various aspects of the design process. Let's take a closer look at a few of them.

Gattrib is a spreadsheet-based attribute editor. It allows you to add, modify, and delete component attributes over an entire design without having to click on all the components on each page of the schematic. Gsymcheck is a command-line tool for checking a symbol for syntactic validity. Garchive is a command-line tool for creating portable, configuration-independent project archives. Sarlacc, olib, and convert_sym are file translators from various commercial EDA tools into gEDA. Finally, note that tragesym and gmk_sym are command-line tools for automating the creation of complex symbols.

New gEDA utilities continue to appear as the popularity of the suite increases. Furthermore, the source code is available, so you can readily reuse pieces of it and build on the work of others. This lowers the barriers for those of you who feel the need to "scratch the itch" and write a new utility. You don't need to reinvent the wheel if another program does something similar to what you want. Just borrow the code and modify it to suit your needs!

COMMUNITY SUPPORT

gEDA's development and extension is a vital, ongoing process. The focal point of the user community is the "geda-user" mailing list, which is accessible through the main gEDA web site (www.geda.seul.org). Questions, as well as requests for support and advice, are welcome. Moreover, if you'd like to contribute a patch or a new utility, post an announcement on the mailing list and your contribution will be appreciated.

To get started with gEDA, refer to the how-to guides posted on the web site. There you can learn how to create gschem symbols, perform SPICE simulations, create PCBs, and carry out a large number of tasks using the gEDA suite.

COMMERCIAL ACCEPTANCE

Although gEDA was originally targeted at students and hobbyists, the

SPICE Your Design

An important step in the process of designing real analog circuits involves using SPICE simulation to check your design. SPICE is an open-source program with a history and pedigree longer than Linux's. Originally written at Berkeley in the 1970s and 1980s, several versions of Berkeley SPICE have been available on the Internet for years, usually as raw source code.

A recent highlight of gEDA's development was the introduction of an advanced SPICE back-end for gnetlist called spice-sdb, which allows you to easily incorporate vendor SPICE models into your schematic and output netlist. Coupled with the availability of open-source SPICE simulators, the gschem-to-gnetlist flow provides a viable method of designing and simulating complicated analog circuits without having to spend thousands of dollars on commercial SPICE simulators.

Many people have worked on SPICE, so the different versions on the 'Net vary in bugginess and features. Therefore, you should be wary of plain SPICE packages that you can find with on-line search engines (unless you like to hack, of course). If you want to get going quickly with a working version of SPICE, stick with one of the packages typically used with gEDA's other tools.

ngspice is the first SPICE allied with gEDA. University of Rome researchers originally developed it as a project to rewrite SPICE as a GPLed application, as well as to provide modern device models missing from the mainstream SPICE applications on the 'Net. Under the pressures of reality, the researchers' grandiose goals quickly became more modest: refactor the preexisting SPICE 3f5 code and incorporate a GNU-standard build system into the code base so that it's easy to install on Linux machines. Along the way, lots of bugs were fixed, and the XSpice extensions were integrated into the code to enable simulations of circuits incorporating vendor models with POLY constructs.

The ngspice project has been successful. The current version, ngspice-15, is the most advanced, bug-free, open-source SPICE simulator available for Linux.

tclspice is an outstanding example of the advantage offered by the open-source software development process. It started with the ngspice code, but extended ngspice by exporting the SPICE command set to the scripting language TCL. With tclspice, you can write TCL programs that automate the process of using SPICE. For example, you can write a program that runs a loop, changes circuit values, and then computes a merit function evaluating your circuit's performance for a variety of different circuit values. This facility is particularly useful if you want to vary several parameters at once.

You can also use tclspice to generate parametric plots of your circuit. Finally, you can write custom circuit optimizers that vary the parameters you chose and perform any optimization algorithm that you can code in TCL.

A third option for analog simulation is the GNU circuit analysis program (gnucap), which was written by Al Davis, who is currently a professor of electrical engineering at Kettering University. gnucap is an outgrowth of Al's doctoral thesis in which he investigated ways to combine event-driven digital simulation with standard continuous-time analog simulation.

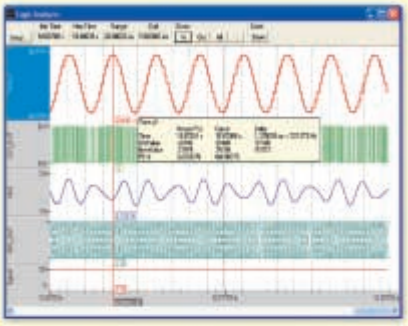
gnucap differs from ngspice and tclspice in several ways. First, it isn't really SPICE at all. The simulation engine is different. gnucap uses an algorithm combining event-driven digital simulation with continuous-time analog simulation. Also, several new component types are present in gnucap. Native digital devices are the most important ones. There are also commands and facilities for behavioral modeling.

gnucap's extended command set is a super-set of SPICE's set. On the other hand, a couple of SPICE devices (e.g., JFETs) are missing, at least in gnucap's current version (0.34). Nonetheless, the gnucap netlist syntax is extremely similar to SPICE, so SPICE users will feel comfortable with gnucap.

If you're using SPICE for a simulation but want a second opinion, try using gnucap!

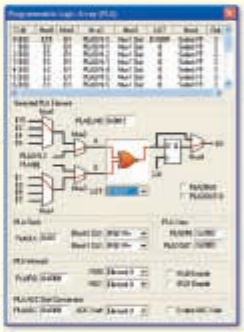
KEIL SOFTWARE

Keil Professional Embedded Development Solutions provide the tools you need for your 8051, 251, C16x, and ARM projects. Tools from Keil help you efficiently create, test, & complete your embedded projects.



The µVision3 IDE/Debugger supports virtual target simulation and includes a software logic analyzer that graphically displays variables and I/O registers.

The µVision3 Development Environment provides complete device simulation and includes peripheral dialogs that aid development of software drivers.



µVISION 3

Supported Architectures

ARM7

Keil development tools for ARM support a wide variety of devices from the most popular chip vendors.

C166

Keil development tools for the XC16x, C16x, and ST10 support over 50 devices from Infineon and STMicroelectronics.

Cx51

Keil development tools for the 8x51 support over 600 different derivatives including devices with extended address space and enhanced instruction sets.

Contact us today for a Free Demo CD

800-348-8051

www.keil.com

YOUR TARGET FOR PROTOTYPE CIRCUIT BOARDS



Successfully selling online since 1997.
Experience you can trust! Try us out today at:
www.PCBexpress.com/cc

Enter the link for a special discount!

For production circuit boards...

Now FREE Shipping option!

Expect the best.
Easy. Fast. Reliable.

Easiest site to quote & order

Best pricing 25 ~ 1000pcs
(No signup required to quote online)

PCBpro
http://www.pcbpro.com/cc

suite has been attracting professional engineers from the beginning. As gEDA has matured, a sizeable community of commercial users has coalesced around the tools. gEDA's openness, ASCII file formats, and liberal distribution policy make it attractive. Also note that the commercial use of gEDA isn't limited to PCBs. The tools are robust enough for chip designs.

A common theme echoed by many commercial gEDA users is that they like the flexibility offered by its GPL license. Clearly, gEDA has found its way into the commercial market. If you're looking for low-cost, flexible, and transparent design tools, you'll probably want to try the gEDA suite.

NONCOMMERCIAL USE

Because downloading the gEDA suite is free, many noncommercial users (e.g., academics) use it. Some of their projects are quite ambitious.

An outstanding example of such usage is the Ronja free space optical data link, which is an opto-electronic communication system comprised of an optical transmitter and receiver assembled with lenses and incorporated in a housing (http://ronja.twibright.com). It operates at 10 Mbps full duplex over a distance of 1.4 km using 625-nm LEDs as the transmit sources. Karel "Clock" Kulhavy, the Ronja project's lead designer, said the system was designed from end to end using the gEDA design suite. Following his philosophy of free technology, Karel has placed all his design files on his web site for others to download and use.

Unlike a traditional networking gadget, Ronja is poised somewhere between a straightforward technology product and a political statement. Although the philosophy behind Ronja differs from a traditional commercial application, the same considerations about tool choice apply: transparency, good user support, and ASCII files. Indeed, in a nontraditional user environment, the toolset's openness may be even more desirable.

GIVE gEDA A GO

If you're in the market for high-quality, low-cost EDA tools, take a look at the gEDA suite. These free high-quality, open-source tools are extremely accessible.

Like Linux during the late 1990s, the gEDA suite has a few minor rough edges; however, it has already become a solid platform for serious work. gEDA's also similar to Linux because many people initially try it out because it's free to download and install. After trying it, most people stick with it!

Stuart Brorson has played with electricity for as long as he can remember, sometimes successfully. All of his university degrees were awarded in the field of electrical engineering. Stuart uses the gEDA suite both professionally and in his spare time. He has contributed several utilities to gEDA/gaf, and occasionally works on ngspice as well. You may contact him at sdb@electroniscript.com.

Ales Hvezda is an embedded real-time software engineer by day and a free software developer the rest of the time. He initiated the gEDA project several years ago and continues to serve as its benevolent dictator. For stress relief, Ales likes to play ultimate Frisbee and fence foil. You can reach him at ahvezda@seul.org

Dan McMahill is a practicing electrical engineer with more than 10 years of analog and mixed-signal design experience. He is a regular contributor to the gEDA and PCB projects. Dan is also a developer for the NetBSD project. When he isn't hacking on circuits or software, he can be found at the ice rink. You may contact him at mcmahill@alum.mit.edu.

RESOURCES

Open-source EDA news, www.opencollector.org.

gEDA/gaf info, www.geda.seul.org.

GNU General Public License, www.fsf.org/copyleft/gpl.html.

ngspice summary, http://sourceforge.net/projects/ngspice.

PCB info, http://pcb.sourceforge.net.



by Ivan Sham, William Hue, & Pete Rizun

Portable FAT Library for MCU Applications

The DOS FAT file system is the industry standard format for flash memory cards. You can use a microcontroller to read and write files on an SD/MMC flash memory card with the FAT system. Read on to learn how to build a portable FAT library for MCU applications.

Many new microcontroller applications require the ability to store large volumes of data. Transferring data between a portable device and a PC has become increasingly simple with the emergence of portable flash memory cards such as the Secure Digital (SD), MultiMediaCard (MMC), CompactFlash (CF), and xD Picture Cards. For the data to be easily readable on the PC, it must be stored using a file system supported by the operating system. The industry standard format for flash memory cards is the DOS file allocation table (FAT) file system. In this article, we'll show you how to use a microcontroller to read and write files on an SD or MMC flash memory card using the FAT system.

If you've ever used a USB memory key or attached a digital camera to your computer, then you've probably noticed that a new drive appears in Windows Explorer. You can drag and drop files to and from this drive. Low-cost flash memory card readers work the same way. Wouldn't it be useful to write files with your microcontroller that anyone can open on a PC with a standard card reader? No special software required! Similarly, wouldn't it be convenient to read files from a flash memory card that were written by a computer? In order to do so, you need to be able to interpret the FAT file system with your microcontroller.

The complete source code needed for using a FAT16-formatted SD/MMC with a Texas Instruments MSP430 microcontroller is posted on the *Circuit Cellar* ftp site. This way you don't have to write your software from scratch. At the end of this article, we'll tell you how to employ the software to write files that can be read with Windows Explorer and a standard flash memory card reader/writer. First, let's examine how to talk to the SD/MMC at the lowest level.

HARDWARE

SD and MMC flash memory cards both support the SPI transfer protocol and share an almost identical electrical interface. Connecting the flash memory card for operation in SPI mode to a microcontroller is similar to connecting a standard SPI device (see Figure 1). A bidirectional chip-select pin facilitates card detection. Pins 8 and 9 can be left open. To create an easily adaptable FAT decoding library, roughly 1 KB of RAM is required on the microcontroller.

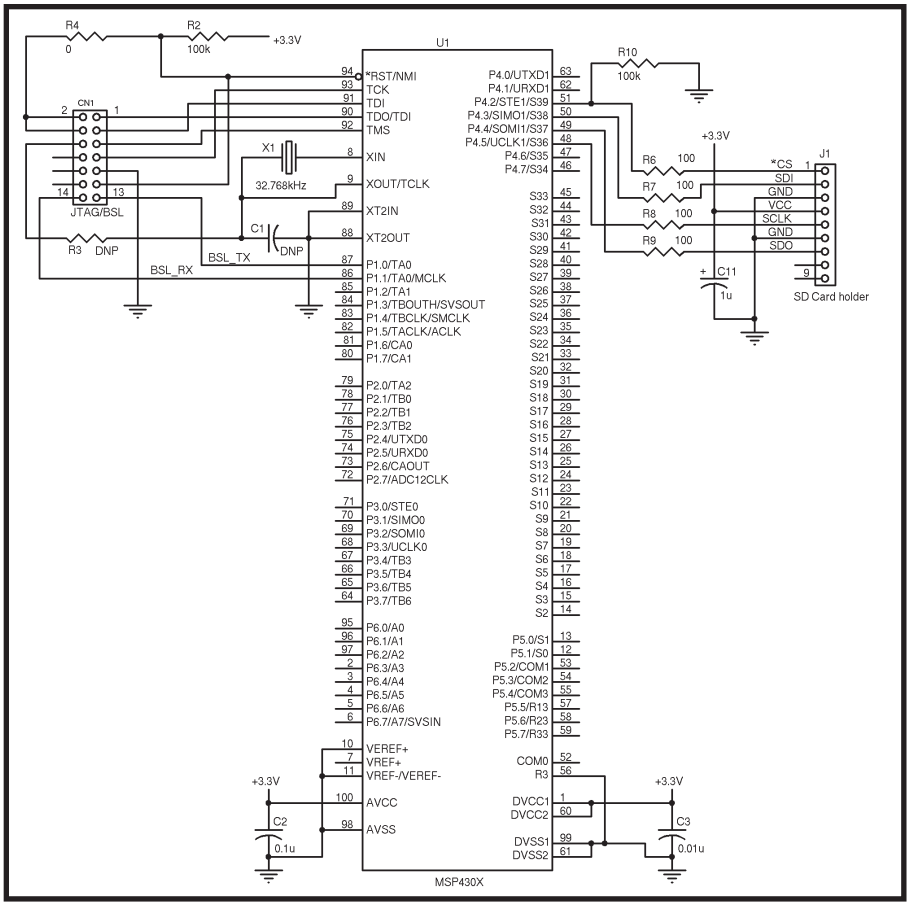


Figure 1—*Connecting an SD/MMC card to a microcontroller is almost as easy as connecting any other SPI device. The CS line serves as the SPI chip select, but it also can be polled to determine the presence of a flash memory card. Because SD cards are slightly thicker, the MMC will fit into an SD cardholder, but not vice versa. We used the Molex MLX54786 surface-mount SD cardholder.*

We chose the MSP430F449 because it has 2 KB of RAM, leaving a full 1 KB for stack and other variables. Photo 1 shows the latest project we completed with SD/MMC flash memory cards.

MEMORY CARD ACCESS

The low-level data transfer between the microcontroller and the SD/MMC is relatively straightforward. An SPI interface permits communication with only four physical connections (CS, SDI, SDO, and SCLK) besides power and ground. In SPI mode, the SD and MMC share an identical set of instructions.

To read or write data stored on a memory card, you must first perform the proper initializations. Because both the SD and MMC cards start in their native transfer modes at power-up, they must be placed in SPI mode. The initialization code needed to accomplish this is shown in Listing 1.

SD/MMC cards sample data input on the rising clock edge and set data output on the falling clock edge, so the microcontroller's SPI module should be configured accordingly. It's critical that the data input line is held high while waiting for the memory card to respond, as well as when the memory card is sending data to the microcontroller.

The first line of the initialization code disables the memory card by raising the active-low chip-select line (MEM_CS). The loop that follows ensures that the memory card has enough time to complete its internal power-up sequence. The `spi_put(byte x)` function clocks its argument out of the microcontroller and into the SDI pin of the memory card. It's important that the argument is 0xFF during this time in order to ensure the memory card powers up properly.

The chip-select line is then lowered and a 0x40 command (initialize card to the idle state) is sent with the arguments 0,0,0,0 and the CRC of 0x95. It's the state of the chip-select line (held low) during this reset procedure that places the memory card in SPI mode. The CRC for this command is precalculated. This is the only time the CRC value is critical, because the CRC is disabled after the card enters SPI mode. The `send_command(byte cmd, byte arg1, byte arg2, byte arg3, byte arg4, byte crc)` function simply transmits its arguments sequentially via six repeated calls to `spi_put(byte x)`.



Photo 1—*The board writes files to an SD card, which conveniently allowed us to store many megabytes of data, and then later transfer the data to a computer using readily available hardware. The MSP430 processor can write files to the flash memory card, which subsequently can be opened in Windows Explorer using a standard card reader. It can also read files from the flash memory card stored using a standard card reader/writer.*

The next line of the code calls `card_response(byte x)` to wait for an appropriate response from the memory card. This function repeatedly calls `spi_get()` and waits for the token specified by the argument before returning `true`. `False` is returned if a match is not received after a number of attempts. The expected token is 0x01 for the initialization command.

To get the card out of the idle state, the 0x41 command is sent repeatedly until the expected response is detected. If the card doesn't respond as expected after 255 tries, `initializeMemCard()` returns `false`, and the application can start anew.

After the expected response is detected, the chip-select line is set high and a dummy byte is sent, generating eight additional clock cycles to permit the memory card to complete its internal operations. The extra clock cycles are necessary for all commands.

We've given you an in-depth look at the initialization process because most events share a similar flow of events. First, lower the chip-select line. Second, clock out the command: command index (1 byte), arguments (4 bytes), and CRC (1 byte). Third, wait for a response byte from the memory card. Fourth, wait for the data token (for read operations), or clock out the data tokens (for write operations). Fifth, transmit or receive a block of data to or

Listing 1—*Initializing the memory card involves powering up the card properly and placing it in SPI mode. The state of the chip select line (held low) during this reset procedure places the card in SPI mode rather than its native Transfer mode.*

```
boolean initializeMemCard(void)
{
    MEM_CS = HIGH;
    for(i= 0; i < 10; i++)
    {
        spi_put(0xFF);
    }
    MEM_CS = LOW;
    send_command(0x40,0,0,0,0x95);
    if(card_response(0x01))
    {
        i = 255;
         do
        {
            send_command(0x41,0,0,0,0,0xFF);
            i—;
         } while (!card_response(0x00) && i > 0);
         MEM_CS = HIGH;
         spi_put(0xFF);
         if(i == 0)
        {
            return false;
        }
         return true;
    }
    return false;
}
```

from the memory card. Sixth, receive 2 bytes of error-checking information. Seventh, raise the chip-select line. Finally, generate eight clock cycles for the memory card to complete the operation.

The fifth and sixth steps were absent from the initialization routine, but they're the meat of read and write operations that we'll cover next. The memory card should automatically initialize to its 512-byte default read/write block size that matches the FAT sector size; nevertheless, it's good practice to ensure this using the set block length command. Table 1 shows all the byte codes for the SD/MMC commands needed to implement the FAT file system software. Although the memory card supports single- and multiple-block read operations, let's stick with single-block 512-byte reads (clock out the read block command, 0x51, with the byte address as the 4-byte argument).

Make sure the address argument for the read/write command is properly formatted for read/write operations. For example, if you want to read the 2,005 physical sector, the 4-byte address should be 0x00, 0x0F, 0xA8, 0x00 because the sectors are numbered starting with zero. Sector 2,004 has a 0x000FA800 byte address, which is determined in the following fashion: (2005–1) × 512 = 1,024,048 = 0x000FA800. Note that SD/MMC byte addresses are in big endian format.

The code in Listing 2 will read sector 2,004 in the memory card. It's similar to the initialization routine. The most notable difference is the presence of a second call to `card_response(byte x)`. The argument (0xFE) in the second call to this function is the data token. All data transfers with the memory card begin with this token. The 512 calls to `spi_get()` in the loop fill `buf` with the contents of sector 2,004. The two extra calls to `spi_get()` clock in the unused CRC bytes attached to every data block sent by the memory card.

Writing data is similar to reading data; the only difference is the direction of the data flow. In a read operation, the memory card provides the data token and the data. In a write operation, it's the microcontroller's responsibility to provide them. To write a block of data to the memory card, you have to send the Write Block command (0x58) with the properly formatted address, wait for the response byte, generate eight clock cycles, send the data token, and begin clocking out the data. Don't forget the 2 bytes of error code at the end of the data block. The code posted on the *Circuit Cellar* ftp site (WriteSDMMC.doc) writes the contents of the character array `buf` to sector 2,004 of the memory card.

Unlike with the read operation, you must make sure the memory card has completed its internal write operation after you clock out the data block. Checking the output from the memory card for a data-response byte can do this. The `checkWriteState()` function repeatedly calls `spi_get()` as it looks for the token 0x05. After the token 0x05 is detected, the completion of the write operation is signaled by the first nonzero byte detected by `spi_get()`.

Now that you know how to interface with the SD/MMC, let's look at how the FAT file system works. We'll provide you with a basic introduction to FAT. There's a wide range of literature describing the FAT file system, from its origins in the PCDOS and MSDOS operating systems to its current applications in Windows.

Command	Byte code	Response byte	Data token received	Data tokens transmitted
Initialize card to idle state	0x40	0x01	–	–
Bring card out of idle state	0x41	0x00	–	–
Read block	0x51	0x00	0xFE	–
Write block	0x58	0x00	–	8 clock cycle, 0xFE
Set block length	0x50	0x00	–	–

Table 1—*Although the SD and MMC flash cards respond to many commands, only four are required to implement our FAT library. It's good practice to set the card's block length to 512 to match the FAT sector size, but it isn't required because this should be the default value.*

FAT EXAMPLES

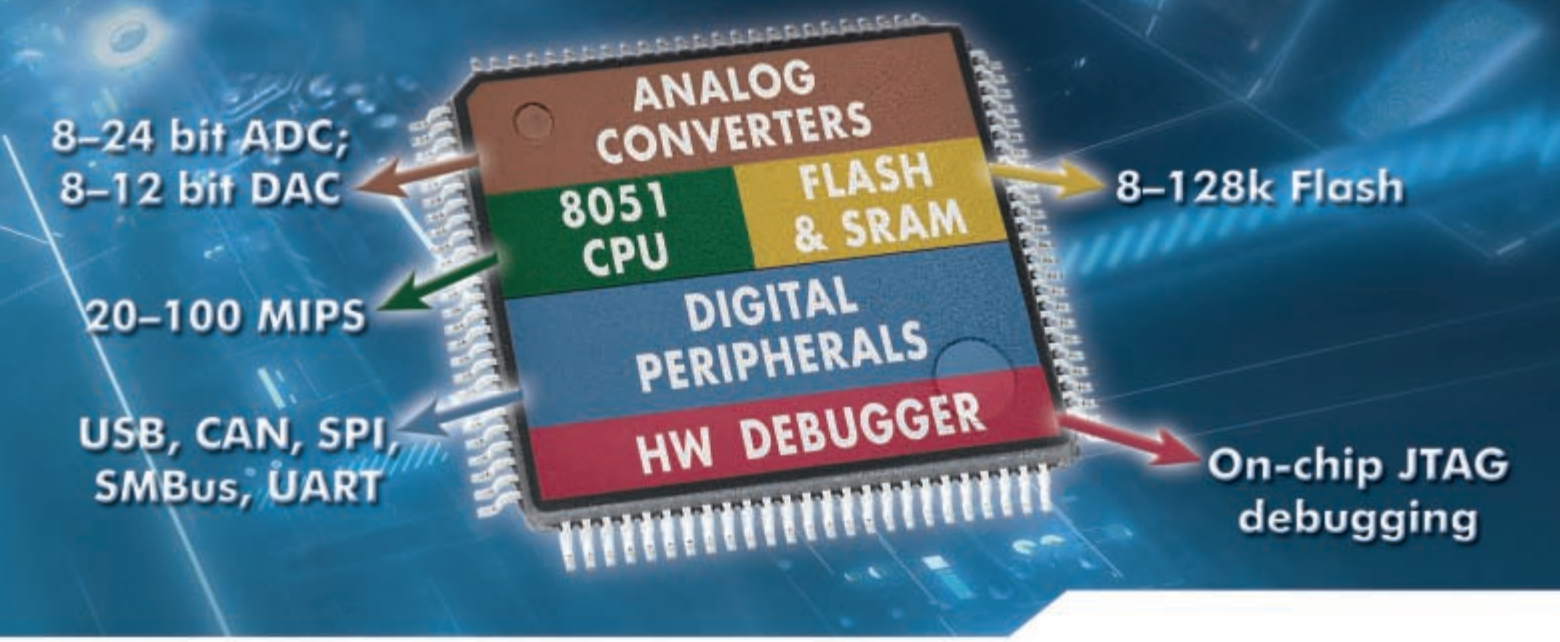
Let's first take a look at two examples of reading and writing files on a memory card. Assume we're working with a card formatted in FAT16, which uses 16-bit numbers to keep track of storage locations in the file system. Approximately 65,000 unique locations may exist in a FAT16-formatted disk.

A memory card is formatted like a hard disk with a single partition (the

Listing 2—*Reading a 512-byte sector involves sending the Read Block command with the address to read, and then generating the SPI clock signals to read the 512 bytes from the memory card, paying attention to the idiosyncrasies of the flash card.*

```
unsigned char buf[512];
boolean read_sector(void)
{
        unsigned short i;
        boolean retval = false;
        MEM_CS = LOW;
        send_command(0x51,0,0x0F,xA8,0,0xFF);
        if(card_response(0x00))
        {
          if(card_response(0xFE))
          {
              for(i = 0; i < 512; i++)
              {
                buf[i] = spi_get();
              }
              spi_get();
              spi_get();
              retval = true;
          }
          MEM_CS = HIGH;
          spi_put(0xFF);
        }
        return retval;
}
```

High-performance analog. Supercharged 8051. Small. Fast. Flexible.



Complete Family of Mixed-Signal MCUs

Silicon Laboratories' MCUs combine high-performance analog with the fastest 8-bit CPU (up to 100 MIPS) packaged as small as 3 x 3 mm. Programmable Flash and on-chip JTAG-based debug in each device provide complete design flexibility and superior system performance.

Precision Mixed-Signal	Precision performance for high resolution analog systems
Small Form Factor	Low power and small footprint with superior performance
General Purpose	Ideal for a broad range of embedded control applications
USB	Integrated USB 2.0 function controller and on-chip transceiver
CAN	Connectivity with integrated CAN 2.0 B controller
Interface Products	Single-chip USB-to-UART bridge for RS-232 to USB

For more information, visit www.silabs.com/MCU



Comprehensive Development Kits

- Software CD
- Serial Adapter
- Target Board
- All Cables
- Power Supply

WIRELESS | WIRELINE | OPTICAL | **MICROCONTROLLERS**

SILICON LABORATORIES

©2004 Silicon Laboratories Inc. Silicon Laboratories and the Silicon Laboratories logo are trademarks of Silicon Laboratories.

www.silabs.com

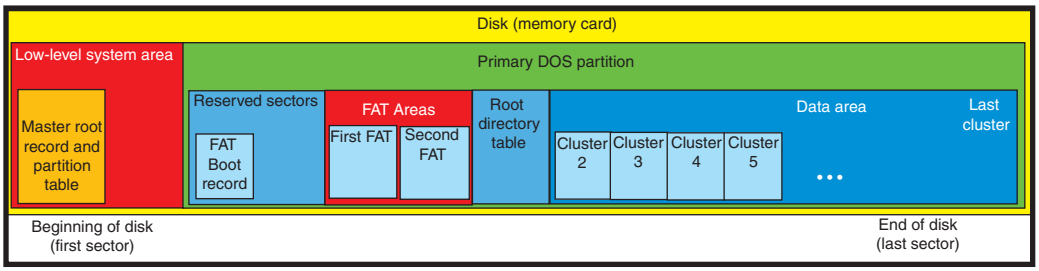


Figure 2—*In this memory map of a FAT16-formatted memory card, the data area is organized as groups of sectors called clusters. The FAT file system keeps track of how much storage is allocated to each file one cluster at a time.*

primary DOS partition). As you can see in Figure 2, the first sector (512 bytes) of the memory card contains the master boot record (MBR). The FAT partition (primary DOS partition) immediately follows it. The partition begins with some reserved sectors (the first of which is the boot record, which shouldn't be confused with the MBR), one or more FATs, and a root directory table. The data area follows the root directory; it's organized as groups of sectors called clusters. Storage space is allocated to files one cluster at a time.

Note that the MBR contains a small bootstrap loader program and the partition table. In a real hard disk, the bootstrap loader finds and runs the secondary bootstrap loader in the active partition's boot record. In a memory card, the MBR still contains the primary bootstrap program, but computers only use the partition table information.

READ A FILE

Consider the task of reading the file called hello.txt stored in the root directory and placing its contents ("Hello World!") in a character array. You can experiment with this example by typing "Hello World!" in a text editor and saving the file in a memory card's root directory. To read the file, you must first locate its entry in the root directory table located in the memory card after the FAT tables (see Figure 2).

The root directory table is organized like a spreadsheet (see Table 2). Each 32-byte row corresponds to a possible file entry. Some rows, however, might be blank (filled with 0 bytes) or refer to deleted files. The spreadsheet's columns provide information about the file. The first column represents the name. The second column represents the extension. (FAT16 uses 8.3 file names. The decimal point isn't recorded.) The penultimate column contains the number of the first cluster that contains the file's data. The final column contains the file size in bytes (little endian format). Other columns (bytes 11 through 25) represent various attributes and creation/modification dates.

To locate the hello.txt file, examine the name and extension columns in the root directory and look for a match. Because the root directory is just data stored in physical memory, you can read the root directory table into your microcontroller sector by sector, using the `read_sector()` function. The first sector in the FAT partition is the boot record, which includes parameters that tell you the sector where the root directory begins. Note that these parameters can be complicated.

Let's assume for now that you know which sectors contain the root directory table. You also know the length, in bytes, of each row in the table. You can easily imagine the process of writing a function to search the spreadsheet for hello.txt. In return you get the row number corresponding to the file entry. If a match isn't found in the sectors assigned to the root directory table, then the sought after file doesn't exist in the root directory.

Byte numbers	0–7	8–10	11–25	26–27	28–31
File 1 properties	name	extension	attrib., date	first cluster	size
File 2 properties	name	extension	attrib., date	first cluster	size
…	…	…	…	…	…
File n properties	name	extension	attrib., date	first cluster	size

Table 2—*The root directory table is organized like a spreadsheet. Each row corresponds to a file entry. The columns provide information about the file. The number of root directory entries (rows) may vary depending on the size of the disk. A common value is 512, in which case this table occupies 32 sectors.*

The original designers of the FAT file system kept things simple. The first characters in the names of deleted files are replaced with a special (non-alphanumeric) value. The first row that begins with a 0 byte (0x00) indicates that no subsequent rows have been used. Therefore, unless the root directory held its maximum number of files at some point, the search can stop without having to read to the last row of the table.

Let's assume that your search is positive and that you know the row corresponding to the file entry for hello.txt. You can read the 2 bytes from the first cluster column in little endian format to determine where the first chunk of information is located. You can read the size column to determine the file's size. To start reading the file contents, you must locate the first sector of the first cluster. The boot record also contains a parameter that tells you how many sectors are grouped together in a cluster.

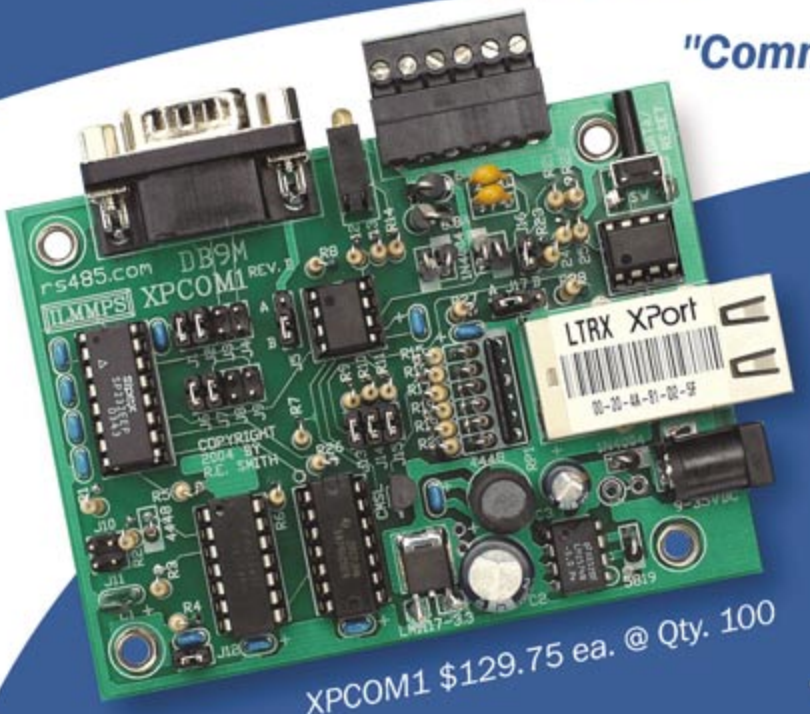
Note that the number of sectors per cluster (the allocation size) is determined when a disk is formatted. Larger clusters allow smaller FAT tables and larger disks to be used within the 65,000-location limit. Nevertheless, they result in wasted space at the ends of files that don't use most of their last clusters.

The data area starts right after the root directory, so you can locate the first sector of the first cluster in the following way. If the first cluster column for hello.txt contains the number 107, the data in hello.txt begins at cluster 107 within the data area. (The first cluster in the data area is cluster number 2, because entries 0 and 1 in the FAT are reserved for holding information about the FAT table.) Now suppose your FAT file system has four sectors per cluster. This means the beginning of the file is located in the absolute sector number:

$$(\text{first sector of data area}) + [(\text{first cluster} - 2) \times (\text{sectors per cluster})]$$

The first sector of data area is computed with the parameter from the MBR to locate the beginning of the FAT partition, and then uses the FAT boot record's parameters to locate the beginning of the data area.

"Communications Glue" ABC<=>XYZ



XPCOM1 $129.75 ea. @ Qty. 100

- RS232/RS485/RS422/TTL
- USB/900 MHz Radio
- Converters/Repeaters
- Fiber Optics/Isolators
- Multi-Repeaters/Hubs
- Extenders/Multidrop
- 10/100 Base-T Ethernet
- Microprocessor Serial I/O
- Code Activated Switches

Extensive Interface Product Line

Custom Units & Smart Units
Digital/Relay I/O & "Extension Cords"
Vending/Security status over TCP/IP
5-Axis Camera PTZ, Focus, Iris Control
Large Multidrop & "Home Run" RS485
Card Reader Electronics & Control
VF/LCD Display Remote Control
"Industrial units" & -40C to 85C
3.0KV Isolation & wide Power range

Call the RS485/Ethernet/Interface/Communications Wizards @ 513 874 4796 Today!



RE Smith

WWW.RS485.COM

4311 R.E. Smith Drive, Hamilton, Ohio 45011

513-874-4796

Download our free Color Coded RS232/485 & Ethernet Terminal Emulators from www.rs485.com

Returning to hello.txt, note that its entry size column reveals that the file is 12 bytes long. Thus, everything necessarily fits in the first cluster (and the first sector). Now the `read_sector()` function can read the file data into your microcontroller. All the bytes after the twelfth are disregarded.

Imagine modifying hello.txt with your PC so it contains 1,000 lines reading "Hello World!". The contents of this modified file might not fit in a single cluster. How can you determine where the rest of the file data is located? The answer is in the FAT. After reading the data from cluster 107, you would go to entry 107 in the FAT to determine the next cluster's location.

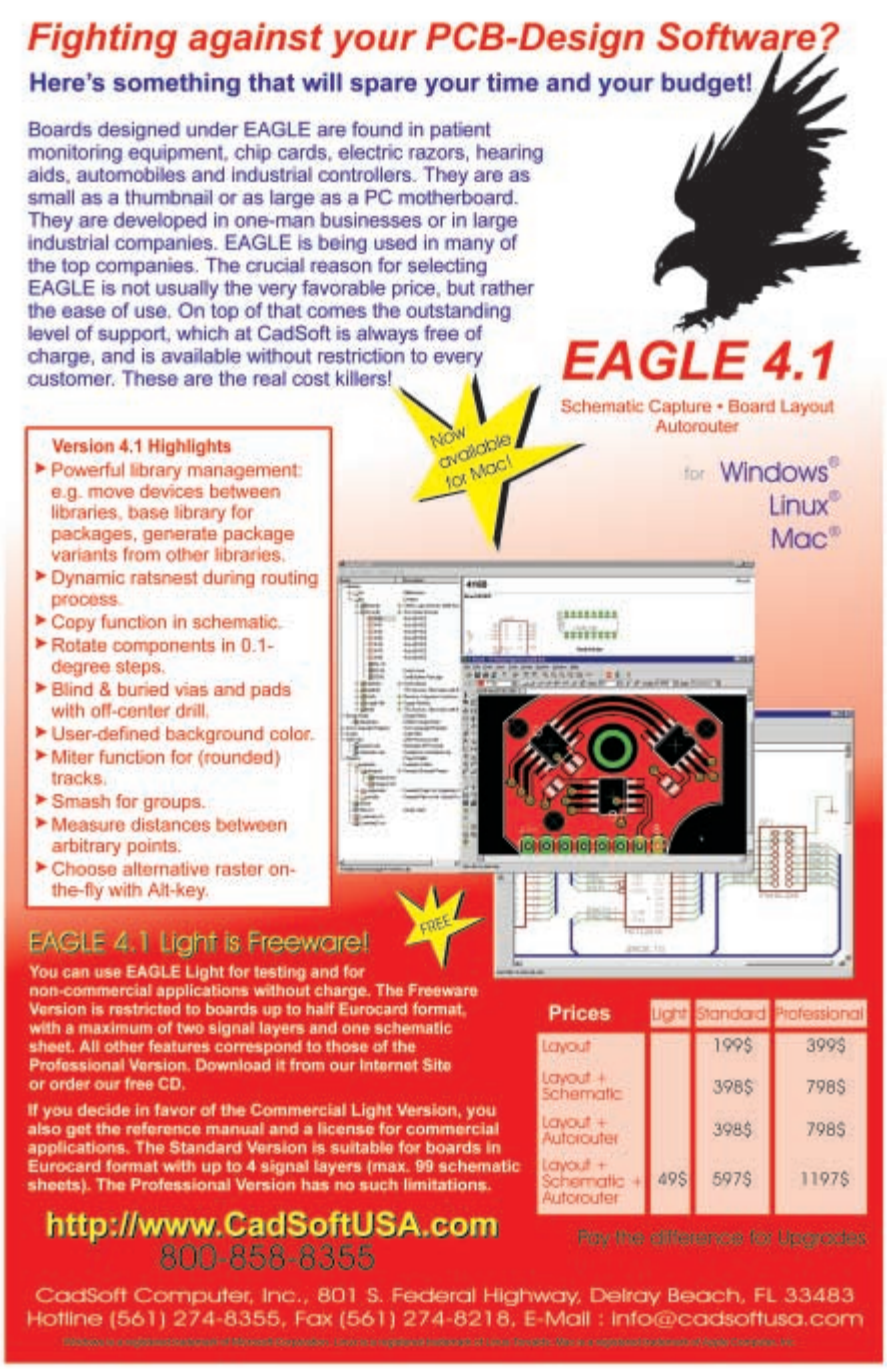
There can be more than one identical copy of the FAT (see Figure 2). The boot record contains parameters that tell you how many FATs are present, the sector length of each FAT, and where the first FAT starts relative to the boot record. Each FAT16 entry is a 16-bit number in little endian notation, so entry 107 is at the 214 and 215 bytes in the first sector of each FAT.

Multiple copies of the FAT are maintained to aid in the recovery of a corrupted file system. The FAT file system structure is simple, but not too robust. If a power failure or computer crash occurs while you're updating information in a FAT, you could lose the allocation chains of numerous files. Each of the FAT copies is updated separately, so only one FAT could get corrupted during a failure. Operating system utilities such as CHKDSK and SCANDISK attempt to minimize data loss by recovering as much information as possible from the uncorrupted copies of the FAT.

To find the next cluster in the file, read the appropriate sector (usually from the first copy of the FAT) into the microcontroller using the `read_sector()` function and then examine the appropriate bytes. Let's say that entry 107 in the FAT contains the number 489. This means the second block data for the file is sitting in cluster 489. The third block of data is sitting at the cluster number written in FAT entry 489, and so on. Repeat this process until you've pieced the entire file together. You'll know when to stop looking for more data, because the directory entry will tell you how many bytes the file contains. Moreover, an entry in the FAT that contains a value of 0xFFF8 through 0xFFFF means there aren't additional clusters associated with the file.

WRITE A FILE

As a second example, consider creating the goodbye.txt file in the root directory with the contents "Goodbye World!". First, you need to find a usable row in the root directory table to make a new entry. You can do this by searching for file names starting with a hexadecimal character, 0xE5 or 0x00, corresponding to deleted or unused entries, respectively. After you find a usable entry, write the file name with the extension, time of creation, and file attribute to the root directory table. The attribute field at byte 11 of a directory entry contains a number of bit flags, including one that indicates whether the entry is a file or subdirecto-



ry entry. When creating regular files, you must set the attribute field to 0x20.

A complication is that you also need to record the location of your file's first cluster in the table. Before doing so, you need to know where that should be. There are two cases you must consider. In the first, you are reusing an entry for a deleted file. The first cluster value and the entire FAT chain for the old (deleted) file is still intact. You can write over the data in the clusters in the existing FAT chain. Add clusters if the new file is longer than the old one.

To reduce disk I/O when a file is deleted, only the first character of its file name is replaced with 0xE5. Its directory entry and the FAT chain are left intact instead of following the chain and setting all its cluster entries to 0x0000 (unused). This is why files could be undeleted. It's also the reason why you could run out of space on a disk that's still reporting available free space if you're trying to create a file in a different subdirectory than one in which the deleted entries were located. Later operating system versions will search for the FAT chains belonging to deleted files when this occurs.

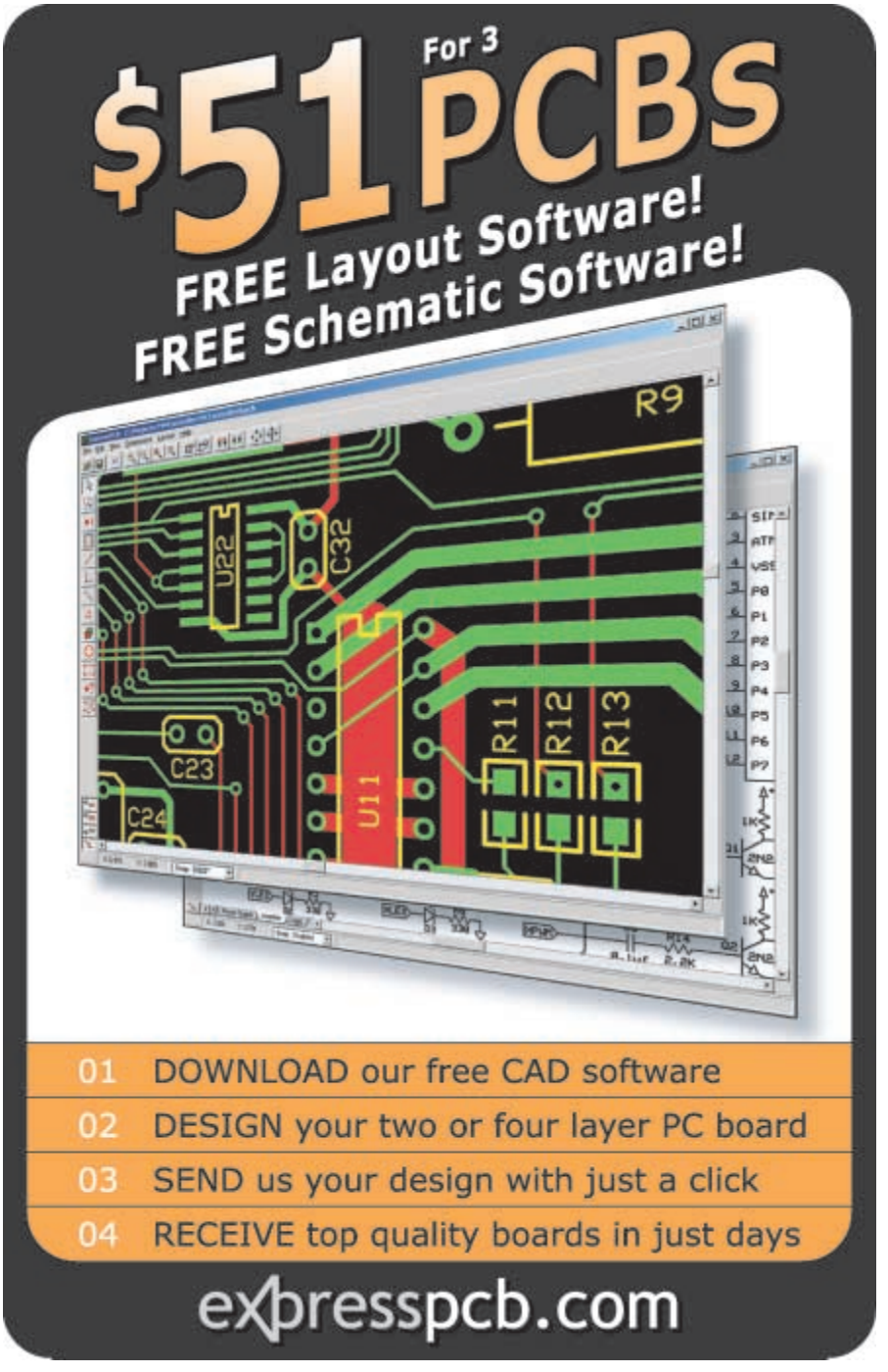
If you are populating a previously unused directory entry and you must find an unallocated cluster, unused clusters are signified by a zero entry in the file allocation table. Simply scan through the FAT (by reading it from our memory card one sector at a time) and look for a zero entry. After an entry is found, change its value to 0xFFFF (by writing the modified sector back to the memory card) to signify the end of the file, and then record this location in the root directory table.

With the location of the empty cluster assigned to the new goodbye.txt file, you can start writing its contents ("Goodbye World!"). You must write at least the first sector of the cluster. Place the string in the beginning of a 512-byte character array buffer and use the `write_sector()` function to send it to the memory card. It doesn't matter what the buffer contains after the last character in your string. We knew that our entire file fit within the first sector of one cluster, so we didn't need to write the remaining sectors in the cluster.

After adding the content to the file, update the file size listed in the directory table. It's good practice to update the access/modify time for the file as well. If a file has grown and it requires more than one cluster of storage space, a new empty cluster must be located in the file allocation table. The location of this new empty cluster is placed in the entry for the previous cluster in the FAT, and the FAT entry for the new last cluster is changed to 0xFFFF to indicate that it's now being used and that it's the end of the allocation chain. In this fashion, a FAT chain (and therefore a file) may grow to any length as long as there are clusters available.

SOFTWARE

The task of creating microcontroller software to read and write FAT-formatted disks can be daunting. The basic software package posted on the *Circuit Cellar* ftp site should allow you to start reading and writing files to the root directory of a memory card formatted in FAT16.



We based this article on FAT16 because it makes the code simple to understand. FAT32 is more complicated, but numerous books and papers have been published that address every version of FAT (FAT12, FAT16, and FAT32).

We wrote the software for the MSP430F449 (see Figure 1). You can easily port it to another platform. We ported our library to the PIC18F*xxxx*. We have a commercial version that works with FAT32, manipulates files with long file names, and performs directory operations. The FAT software package is based on two modules, HALayer and FATLIB, each of which consists of a .C (source) file and an .H (header) file. To write a program that can access an SD/MMC card, simply include the two header files and call the appropriate functions in your main program.

Listing 3 is a sample main program that reads the first 12 characters from the hello.txt file in the root directory. It prints the string to an LCD (presumably the ubiquitous "Hello World!"). The program then creates a goodbye.txt file and writes the "Goodbye World!" string. After the main program is complete, you should be able to insert the memory card into a standard memory card reader/writer on your computer. You'll see the goodbye.txt file in Windows Explorer. Verify its contents.

Listing 3—*The demo program illustrates the process of reading and writing to the SD/MMC card with our FAT library.*

```
int main(void)
{
        signed int stringSize;
        signed char handle;
        char stringBuf[100];
        cpu_init();     //Initialize the CPU clocks, etc.
        eint();         //Enable global interrupts (if required).
        fat_initialize();       //Initialize the FAT library.
    handle = fat_openRead("hello.txt");
    if (handle >= 0)
    {
              stringSize = fat_read(handle, stringBuf, 99);
              stringBuf[stringSize] = '\0';
              lcd_print(stringBuf);
              fat_close(handle);
    }
              handle = fat_openWrite("goodbye.txt");
    if (handle >= 0)
    {
              strcpy(stringBuf, "Goodbye World!");
              stringSize = strlen(stringBuf);
              fat_write(handle,stringBuf, stringSize);
              fat_flush();  //Optional.
              fat_close(handle);
    }
    while (1)
    {
                            //Stay here.
    }
}
```

TIME TO WORK

We'll continue to enhance our FATLIB with features as required by our customers. Now it's your turn to develop an exciting application for an SD or MMC memory card.

Ivan Sham is studying engineering physics at the University of British Columbia. Ivan expects to graduate with a bachelor's degree in April 2005. You may reach him at ivan sham@gmail.com.

William Hue owns HUE-Mobile Enterprises. He specializes in embedded systems, real-time control systems, industrial-grade systems, RF technology, communications systems, mixed signal designs, automotive technologies, and data acquisition systems. He holds B.A.Sc. and M.A.Sc. degrees in electrical and electronics engineering from Simon Fraser University. William may be reached at william@hue-mobile.com.

Pete Rizun designs and manufactures electro-mechanical devices. He has a B.A.Sc. in engineering physics from the University of British Columbia. Pete is currently a Ph.D. student at the University of Calgary, where he's working on an initiative in surgical robotics known as Project neuroArm. You may contact him at pete@rizun.com.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/176.

RESOURCES

J. Bachiochi, "SmartMedia File Storage," *Circuit Cellar* 143–144, June–July 2003.

FAT File systems and SD/MMC commands, www.ntfs.com/fat-systems.htm.

FAT resources, www.systemsmedic.com/SoftwareEdu2.htm; http://students.cs.byu.edu/~cs345ta/labs/winter04_specs/lab_fat_help.htm; www.seas.ucla.edu/classes/mkampe/cs111.fq04/docs/dos.html; www.extonpccouncil.org/Resources/EPCC_Tech_Talk/d003f32.htm.

Microsoft Corp., FAT32 File system specification, www.microsoft.com.

———*Microsoft MS-DOS Programmers Reference*, ver. 5, Microsoft Press, 1991.

MMC Specification, www.mmca.org.

P. Norton, *The Peter Norton Programmer's Guide to the IBM PC*, Microsoft Press, 1985.

M. Sargent and R. L. Shoemaker, *The IBM PC from the Inside Out*, Addison-Wesley, 1986.

A. Schulman, et al, *Undocumented DOS*, Addison-Wesley, 1990.

SD Card info, www.sandisk.com.

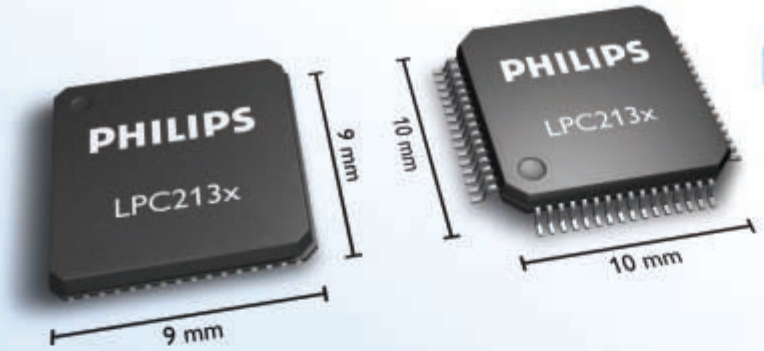
C. Siechert, *The Power of Running PC-DOS 3.3*, 2nd ed., Management Information Source, 1987.

SOURCES

MLX54786 SD memory card connector
Molex
www.molex.com

MSP430x44x Microcontroller
Texas Instruments
www.ti.com

Get your design skills ready. Coming soon...
ARM Design contest 2005



LPC213x
32-bit ARM7TDMI-S
microcontrollers

We've squeezed more in so you can get more out

- 60 MHz operation (54 MIPS) from both on-chip Flash and SRAM
- Two UARTs, two I²C, one SPI and one SPI / SSP interfaces
- 8-channel 10-bit ADC and 10-bit DAC
- User-code security
- Real-time Debugging & Trace
- ISP, IAP, Parallel Programmer Support
- Tiny packages – HVQFN64 (9 x 9 x 0.85 mm), QFP64 (10 x 10 x 1.4 mm)

Ease your migration to 32-bit performance with Philips LPC213x microcontrollers. The industry's fastest ARM7 family provides up to 512 Kbytes of 60 MHz Flash memory with a host of on-chip peripherals and interfaces. All of this comes with low power consumption, tiny packages and full support from recognized third party tool providers. It's the performance you would expect at a price you can afford.
For more information go to www.philips.com/microcontrollers

Part Number	Memory		Timers			I/O	Serial Interfaces			Analog		Interrupts	F. max	Package
	Flash	RAM	# of Timer	PWM	RTC/ WD	Pins	UART	I²C	SPI	A/D ch/ 10b	DAC ch/ 10b	Ext. Levels	(Mhz)	
LPC2138	512K	32K	4	6	Y	47	2	2	2	2x8	1	22(4)16	60	LQFP64, HVQFN64
LPC2136	256K	32K	4	6	Y	47	2	2	2	2x8	1	22(4)16	60	LQFP64
LPC2134	128K	16K	4	6	Y	47	2	2	2	2x8	1	22(4)16	60	LQFP64
LPC2132	64K	16K	4	6	Y	47	2	2	2	1	1	22(4)16	60	LQFP64, HVQFN64
LPC2131	32K	8K	4	6	Y	47	2	2	2	1	-	22(4)16	60	LQFP64

PHILIPS
sense and simplicity

© Koninklijke Philips Electronics N.V., 2005. All rights reserved.

Embedded Security Design (Part 1)

Product Enclosure

Last August, Joe introduced you to the topic of embedded security. Now he's focused on showing you the proper techniques to get the job done. This month he explains how designing the right enclosure can prevent tampering and reduce the possibility of a passive attack like someone eavesdropping on your product's EMI emissions.

In this two-part series of articles, I'll focus on design solutions and protection methods for secure hardware. I'll answer the "how" of embedded security. In doing so, I'll describe previously successful hardware attacks. By studying such attacks, you can learn from the design mistakes and improve your products.

Let's start with the premise that nothing is 100% secure. No matter how many layers of security you use (e.g., tamper proofing, cryptographic processors, and sensors), a determined attacker can always undermine your product. As gloomy as that sounds, your challenge is to figure out what you need to protect your product. You could have a top-of-the-line lock installed on a 4'-thick titanium door, but even the best security is useless if you hide the key to the door underneath the doormat. Or what if you lock the front door, but forget to lock the window? One attack vector is secure (the front door), but another still exists (the window).

You must understand the security mechanisms you choose and ensure that they are correctly implemented. Improper implementation of a security feature gives you a false sense of security and is usually how an attacker will successfully penetrate your product.

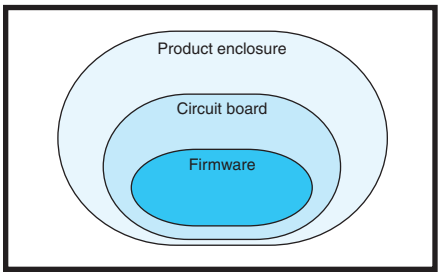


Figure 1—*To protect your hardware with layers of security, you must design multiple security mechanisms on each level of the product.*

A product's enclosure and circuit board are the two layers of security I'll cover in this series (see Figure 1). The design features of each layer are important, and a layered solution (such as implementing multiple security mechanisms) decreases the changes of a successful attack against your product. For more information about the importance of embedded security, the types of threats to your product, and getting into the mind of an attacker, refer to my article "Understanding Embedded Security" (*Circuit Cellar*, 169, August 2004).

A secure enclosure is critical for preventing attackers from gaining access to your product's internal circuitry. An attacker will reverse engineer an accessible circuit board to create a schematic and then identify possible attack vectors. Most products don't incorporate mechanisms to prevent them from being opened. In many cases, it's as simple as loosening a few screws or prying open the side with a knife or screwdriver (see Photo 1).

ANTI-TAMPER MECHANISMS

The purpose of an anti-tamper mechanism is to prevent an attacker from physically or electronically tampering with your device. Anti-tamper mechanisms must be properly implemented to be successful. They can be divided into four groups: resistance, evidence, detection, and response. Such mechanisms are most effectively used in layers to prevent access to critical components.

Normally, it's only by disassembling a product that an attacker can discover the anti-tamper mechanisms. This may require him to obtain more than one device in order to sacrifice one for the purpose of discovering such mechanisms. After the mechanisms are noted, he can form hypotheses about how to attack and bypass them.

In "Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses," Steve Weingart describes anti-tamper mechanism attacks and defenses, ranging from the inexpensive and simple to extremely costly and complex. The paper is a comprehensive guide to many (if not all) known types of attack. It provides a list of solutions for protection against attacks.

Physical attacks are achieved by probing (e.g., passive, active/injector, pico probes, and energy), machining methods (e.g., manual material removal, mechanical, water, laser, chemical, and shaped charge), and electrical tactics (e.g., radiation imprinting, temperature imprinting, high-voltage imprinting, power supply fluctuations, clock glitching, circuit disruption, and electron beam and infrared laser read/write). Corresponding countermeasures for these attacks are described in the paper. You can use various

types of physical barriers (e.g., hard barriers, single-chip coatings, and insulator-based substrates). Tamper-evident solutions (e.g., brittle packages and crazed aluminum), tamper detection sensors (e.g., motion, ultrasonic, microwave, and infrared), and tamper response technologies (e.g., RAM power drop, RAM overwrite, and physical destruction) are also options.

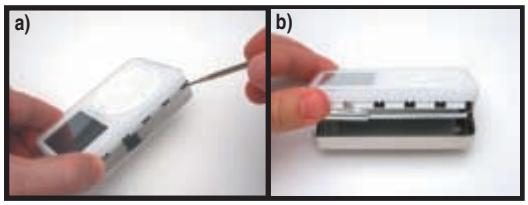


Photo 1a—*Check out how easy it is to open the housing of an Apple iPod with a jeweler's screwdriver.* **b**—*Most products don't incorporate security mechanisms to prevent them from being opened.*[1]

TAMPER RESISTANCE

Tamper resistance consists of using specialized materials to make it difficult to tamper with a device or module. This can include features such as hardened steel enclosures, locks, encapsulation, and security screws. A side benefit of many tamper-resistant mechanisms is that they're often tamper-evident, which means you can observe physical changes in the product.

If you're designing or retrofitting a housing that requires screws, consider implementing one-way screws—which are designed to be tightened, not loosened—that will offer additional tamper resistance. Security screws (i.e., screws that have a specially shaped, non-standard head) are not sufficient because the bits to such screws can be obtained on the 'Net and at most electronics stores. Although an adversary can drill through one-way screws, they make it more difficult than an industry-standard screwdriver, Torx driver bit, or other security screw. The Thomas Register Directory provides a long list of security and tamperproof screw manufacturers and suppliers (www.thomasregisterdirectory.com).

You should also consider sealing both sides of the housing together in such a way that would require an attacker to destroy the device in order to open it. Many plastics are sensitive to heat and melt at fairly low temperatures, so an attacker could easily pry open your sealed housing with a heat gun and X-Aacto knife.

If preventing access to your circuitry is a concern, consider high-temperature glue or ultrasonic welding to seal the housing. If you use high-temperature glue, choose one with a higher softening point than your plastic housing in order to increase visible tamper evidence. (Your product's plastic will melt before the glue, thus providing evidence that it's been tampered with.)

Serviceability may be an issue if your product is meant to be opened by authorized personnel. If a legitimate user can open the device, an attacker can too.

TAMPER EVIDENCE

The goal of tamper evidence is to ensure that there is visible evidence left behind when tampering occurs. Tamper-evident mechanisms are a major deterrent for minimal risk takers (e.g., undetermined attackers or curious hobbyists). Hundreds of tamper-evident materials and devices are available; most consist of special seals and tapes that make it obvious when tampering has occurred.

Tamper evidence features are successful only if there is a process in place to determine if tampering has occurred or if the legitimate owner of the device notices a deformity. Generally speaking, if an adversary purchases a product with the specific intention of attacking it, tamper-evident mechanisms alone will not prevent the attack.

In *Vulnerability Assessment of Security Seals*, Roger Johnston and Anthony Garcia explain how they were able to defeat 94 different security seals (e.g., adhesive tape, plastic, and wire loop) using low-cost tools and readily available supplies. The time for a successful attack ranged from 3 s to 125 min. The cost of the necessary attack tools ranged from $0.15 to $750.

Even though Johnston and Garcia proved that many security seals could be broken, it's still better to employ a solution rather than leave one out because it might be bypassed. A simple solution is Holdtite's Secure 42 super glue, which can provide evidence of tampering when placed on screw holes, potentiometers, and other moveable objects.

More complex solutions include using brittle plastics and enclosures that crack or shatter when an attacker attempts to penetrate them. You could also try covering the housing with "bleeding" paint. When the surface of your product is scratched, paint of one color is mixed with a paint of a contrasting color.

TAMPER DETECTION

Tamper detection mechanisms enable you to be aware of tampering. Such mechanisms typically fall into one of three groups: switches, sensors, and circuitry.

Switches include microswitches, magnetic switches, mercury switches, and pressure contacts to detect the opening of a device, the breach of a physical security boundary, and the movement of a particular component.

The sensor group includes temperature sensors, voltage sensors, and radiation sensors for X-rays and focused ion beams, which are often used for advanced attacks to image and modify integrated circuits at the die level.

The circuitry group includes flexible circuitry, nichrome wire, and fiber optics wrapped around critical circuitry or specific components on the board. These materials are used to detect if there has been a puncture, break, or attempted modification of the wrapper. For example, if the resistance of the nichrome wire changes or the light power traveling through the optical cable fluctuates, the system can assume there has been physical tampering.

W. L. Gore and Associates's D3 electronic security enclosures are designed to protect the physical security boundary of a module (www.goreelectronics.com/products). They combine a number of tamper evidence and detection features. The sensor comes as a foldable sheet that wraps around your product. Conductive ink crisscrosses through the sheet with a maximum distance between traces of 200 to 300 µm (a pitch too small to be drilled through without detection). The electrical state of the sensor changes if the field is broken. This triggers the product to enable its tamper-respondent mechanisms.

W. L. Gore and Associates claims the device is transparent to X-rays, which may be used to determine the location of the sensor within the product, and that it has been tested against a wide range of reagents and solvents. The outer layer has an opaque resin coating that conceals the sensor's surface details and prevents an attacker from seeing any traces. This product also meets the requirements of FIPS 140 level 4 specification for cryptographic modules (csrc.nist.gov/cryptval).

TAMPER RESPONSE

Tamper-response mechanisms are the actions or countermeasures taken after tampering is detected. Most often, the response consists of completely disabling the device or erasing critical portions of memory to prevent the attacker from accessing secret data. Response mechanisms may be simpler, such as those that just log the type of attack detected and the time it occurred. Such logs can provide useful audit information and help with forensic analysis after an attack.

It is unlikely that tamper response mechanisms will trigger accidentally. Even still, as a designer or legitimate user, you must understand the environmental and operational limits of the device, and keep it within those limits. Many tamper-responsive devices are designed and manufactured with the stipulation that they're never to be opened (legitimately or not).

EXTERNAL INTERFACES

Sometimes an attacker can successfully defeat a product's security mechanisms without having to open it. External interfaces, which are a product's lifeline to the outside world, may be used for a number of purposes, including connecting to peripherals, field programming, and testing during product manufacturing. Typical interfaces such as RS-232, USB, FireWire, and Ethernet are usually used for data synchronization (e.g., HotSync for Palm devices and ActiveSync for Pocket PC devices) and device communications. Although physical connections are the focus of this section, it is important to note that wireless interfaces—including but not limited to Wi-Fi, Bluetooth, ZigBee, and IrDA—are at risk and also should be secured. Each protocol has its own set of implementation and security issues that must be considered when they are designed into a product.

Photo 2 shows an example of a proprietary external interface on the Nokia model 3360. The test points (the five metal dots) are accessible behind the phone's battery. If an attack is possible using this interface, the battery can be replaced afterwards, leaving no signs of tampering. Simply obfuscating these interfaces with proprietary connector types or hidden access doors and holes is not suitable because they are easily discovered.

When an attacker gains access to an external interface, the first thing he'll usually do is probe the connections to determine the interface's functionality (if it isn't obvious). Monitoring the test points for device-generated signals (using a multimeter, oscilloscope, or logic analyzer), and then manually toggling the state of the pins to induce a response from the device, usually achieves this. Knowing the state of the pins can help an attacker make an educated guess about the product's type of interface.

After the interface is known, it is trivial for an attacker to monitor the communications using a dedicated hardware tool such as a protocol or logic analyzer. He could also use software-based tools like SnoopyPro for USB (sourceforge.net/projects/usbsnoop), SysInternals's PortMon for serial (RS-232) and parallel ports (www.sysinternals.com/ntw2k/freeware/portmon.shtml), and Ethereal for network protocols (www.ethereal.com). One attack against a known protocol is to generate malformed or intentionally bad data. If the product doesn't handle such data properly, the device may respond in an unintended way that's useful to the attacker. Watching the external interface can also yield interesting product-specific pieces of data.



Photo 2—*Take a look at the hidden external interface on a Nokia 3360. The test points are concealed behind the battery. An attacker can easily probe them.*

Some products also implement development or programming interfaces such as the industry-standard JTAG IEEE 1149.1, which isn't meant for everyday consumer use, but can benefit an attacker immensely. JTAG is a four- or five-wire chip-level interface used for testing and debugging purposes (e.g., system-level testing, boundary-scanning, and the low-level testing of dies and components). An attacker could use it to probe device hardware and communicate with any devices connected to the chip that supports the interface (e.g., flash memory connected to a microprocessor that contains a JTAG module). For example, xda-developers.com discovered an attack against an XDA Pocket PC-based device through its JTAG interface in which wires were simply attached to the proper test points and the system's internal flash ROM could be reprogrammed.[2]

Removing external programming or test interfaces may increase the complexity of manufacturing or field upgradeability at the expense of security. Test points that aren't absolutely necessary should be removed. Those that must remain on the board shouldn't be obvious and should be placed in areas that are hard to probe after assembly. JTAG functionality should be removed from production units if possible. If an interface is simply obfuscated or disconnected from the device (such as by blowing fuses or cutting traces on the PCB during manufacturing), an adversary could easily reconnect or repair the interface and launch an attack.

If you must have an external interface, use caution when connecting to the outside world. No secret or critical components should be able to be accessed through the external inter-

face; only public information should pass. An attacker can easily access plain text and encoded content. For instance, the encoded system password on Palm devices can be retrieved and decoded into the original ASCII password, thus giving an attacker access to password-protected data.[3] If possible, encrypt the traffic to increase the difficulty of reverse engineering and attack.

EMISSIONS AND IMMUNITY

All electronic devices generate electromagnetic inference (EMI) in one form or another. The prevention of compromising emissions is an important aspect of hardware security.

Emission security can be traced back to World War I, when field telephone lines were monitored by way of crosstalk, which is interference caused by the energy from one telephone conversation invading another via electrostatic or electromagnetic coupling. Wim van Eck's 1985 article, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?" was the first academic paper on the subject. He described the results of research concerning eavesdropping on computer monitors by picking up and decoding the electromagnetic interference. Because of the sensitive nature of the subject, most research was classified by the U.S. government and soon disappeared from public literature. Much of the government shielding requirements (known as "TEMPEST") remain classified. However, many unclassified documents are available on John Young's TEMPEST Documents web site (cryptome.org/nsa-tempest.htm).

At the enclosure level, EMI shielding can be easily designed in or retrofitted to a design in the form of coatings, sprays, tapes, or housings in order to decrease emissions and increase immunity. If the enclosure is metal, you can use EMI tape to seal slots and seams in the case. If the case is plastic, you can use EMI spray (a conductive paint) to coat the inside.

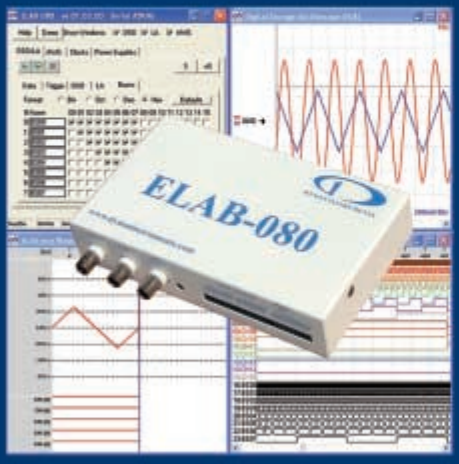
Note that EMI tape can loosen and cause short circuits. EMI paint can chip if the surface isn't cleaned prior to application. Also, be aware of changes in the thermal characteristics caused by EMI shielding. A number of manufacturers—including W. L. Gore and Associates, Schlegel, and GC/Waldom Electronics—offer EMI shielding solutions. The Thomas Register Directory also provides a long list of EMI/RFI shielding manufacturers and suppliers.

To help with EMI reduction, noisy circuitry like power supply components should be compartmentalized to one area of the board and supported with proper filtering. Additionally, power supply circuitry should be as close as possible to the power input.

Eavesdropping on EMI emissions is referred to as a passive attack. An active attack consists of directing high-energy RF (HERF) signals or electrostatic discharge (ESD) at the product in order to cause failures. Properly designing a product to meet defined EMI and RF emissions and immunity conditions are part of many specifications, including those determined by the FCC, European

DYNON INSTRUMENTS

A Complete Electronics Lab for $495



Includes 5 Instruments

- 2 Channel 80 MHz Digital Storage Oscilloscope
- 16 Channel Logic Analyzer, Synchronous with DSO
- Arbitrary Waveform Generator
- 2 Programmable Power Supplies
- 2 Programmable Clocks

Plus....

- Powerful PC User Interface Program
- Probes
- Cables

ELAB-080

www.dynoninstruments.com
Woodinville, WA 98072 (425) 402-0433

Union, International Electrotechnical Commission, and FDA.

Essentially the inverse of emissions testing, immunity testing subjects a product to various RF phenomena to see if they affect the product's performance. ESD protection components are often designed into external connectors and contacts to reduce the chance of failure. One attack uses an ESD simulator tool to generate a high-voltage spike and inject it into a device's external interface or keypad in hopes of causing an unintended condition (e.g., causing a microprocessor's program counter to jump to a different portion of code or changing the values on the address or data bus). However, unless the injection of HERF or ESD can be reproduced in a controlled manner, the results may be too unpredictable to be useful to an attacker for anything other than product destruction.

NEXT STOP: CIRCUIT BOARD

Next month, I'll peel back a layer of the embedded system and describe the security issues at the circuit board level. Many of the weaknesses, security vulnerabilities, and design flaws of a product can be identified when analyzing a product's circuit board.

Joe Grand is the president of Grand Idea Studio, Inc., a product development and intellectual property licensing firm, where he specializes in embedded system design, computer security research, and inventing new concepts and technologies. Joe holds a B.S.C.E. from Boston University. His interests include competitive running, collecting classic video games, and banging on drums. You may reach him at joe@grandideastudio.com.

REFERENCES

[1] J. Grand, R. Russell, and K. Mitnick, eds., *Hardware Hacking: Have Fun While Voiding Your Warranty*, Syngress Publishing, Rockland, MA, 2004.

[2] xda-developers.com, "JTAG for the XDA," xda-developers.com/JTAG.

[3] J. Grand (Kingpin), "Security Analysis of the Palm Operating System and its Weaknesses Against Malicious Code Threats," Tenth USENIX Security Symposium, 2001, www.usenix.org/publications/library/proceedings/sec01/full_papers/kingpin/kingpin.pdf.

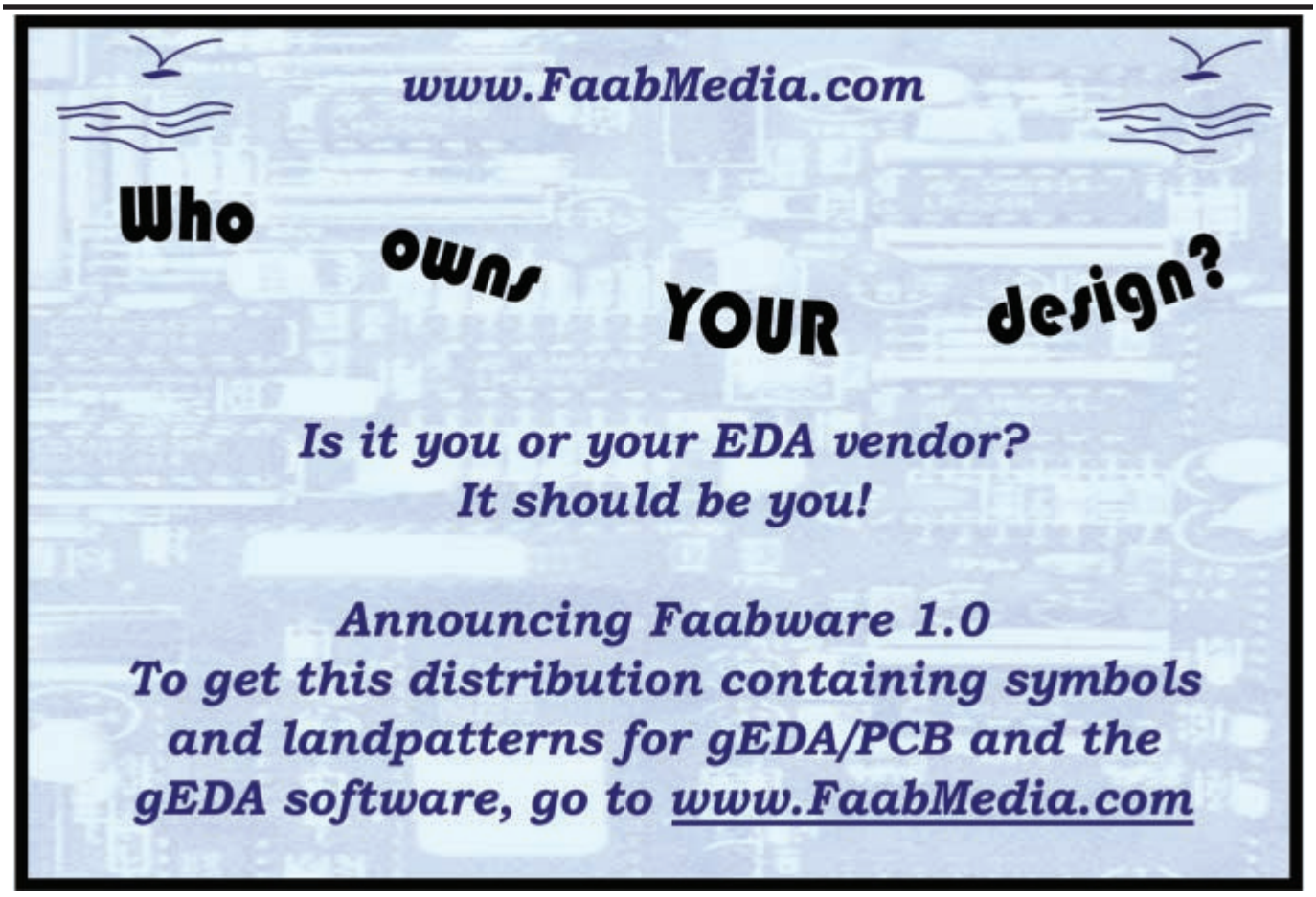
RESOURCES

D. Chaum, "Design Concepts for Tamper Responding Systems," Advances in Cryptology: Proceedings of Crypto 83, Plenum Publishing Corp., New York, NY, 1984.

R. Johnston and A. Garcia, "Vulnerability Assessment of Security Seals," Journal of Security Administration, 1997, www.securitymanagement.com/library/lanl_00418796.pdf.

S.H. Weingart, "Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses," Workshop on Cryptographic Hardware and Embedded Systems, 2000, http://link.springer.de/link/service/series/0558/bibs/1965/19650302.htm.

W. van Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?" Elsevier Publishers, North-Holland, 1985, www.jya.com/emr.pdf.





Everywhere you imagine. RENESAS

MCU Solutions Matched To Your Design Imagination

RENESAS IS THE #1 MCU SUPPLIER IN THE WORLD. **www.renesas.com**

M16C Advantages

- Advanced 16-bit and 32-bit CISC cores that perform at up to 32 Dhrystone 1.1 MIPS
- Banked-register architecture optimized for fast context switching
- Wide range of high-performance flash memory: 8 KB to 512 KB
- Best-in-class EMI/EMS noise immunity
- Designed to be pin- and function-compatible between families and packages
- Extensive peripheral integration: everything from POR and LVD to CRC counters

Low-cost Renesas Toolchain

- Fully integrated suite of hardware and software tools
- Highly-optimized C compiler and assembler — FREE (64K limited version): Download latest version from web (www.renesas.com/skp)
- Low-cost Starter Kit (target board, USB debugger, complete toolchain), starting at $49
- Easy online access to application notes, example code, software development tools

Get your product to market faster using Renesas Interactive!

Renesas INTERACTIVE

Evaluate, test, and learn about Renesas products in an interactive online environment.

www.renesasinteractive.com

SKP	Device	C Compiler, Assembler, IDE	Cost
SKP8CMINI13	R5F21134FP	FREE (64K limited version)	$49
SKP8CMINI17	R5F21174FP		$49
SKP16C26A	M30260F8AGP		$49
SKP16C28	M30280FAHP		$49
SKP16C62P	M30626FHPFP		$49
SKP32C83	M30835FJGP		$119
SKP32C84	M30845FJGP		$49

Special pricing is in US dollars from Renesas authorized North American distributors only, and is subject to change.

Popular Devices

Series/ Group	Part Number	CPU (Bits)	Package	Freq. (MHz)	Dhrystone 1.1 MIPS (Max)	Program Flash (KB)	Data Flash (KB) (10K Rewrites)	SRAM (Bytes)	Serial I/O Channels	I²C	I/O	A/D 10-bit	Ring Osc.	LVD POR	Timers	Price *
R8C/17	R5F21172SP	16	20 Pin SSOP	20	9	8	2	512	1	1	15	4	X	X	3+WDT	$1.90*
R8C/13	R5F21134FP	16	32 Pin LQFP	20	9	16	4	1K	2	–	24	12	X	X	4+WDT	$2.38*
M16C/26A	M30260F8AGP	16	48 Pin LQFP	20	13	64	4	2K	3	1	39	12	X	X	8+WDT	$4.76*
M16C/28	M30281FAHP	16	64 Pin LQFP	20	13	96	4	8K	4	1	55	13	X	X	9+WDT	$5.90*
M16C/62P	M30624FGPGP	16	100 Pin QFP	24	16	256	4	20K	5	3	87	26	X	X	11+WDT	$8.25*
M16C/62P	M30626FHPGP	16	100 Pin QFP	24	16	384	4	31K	5	3	87	26	X	X	11+WDT	$10.05*
M32C/84	M30845FJGP	16	144 Pin LQFP	32	32	512	–	24K	5	5	124	34	X	X	11+WDT	$16.88*

* Suggested 10,000 piece price. Please contact your local Renesas distributor for volume pricing.

Practical Application for TDD (Part 1)

Write Software in a Test-Driven Development Environment

Test-driven development (TDD) in the embedded environment is an important concept to learn. In this series of articles, a group of imaginative designers puts the idea to the test. The end result is an interesting temperature calculation project.

Agile software development is one of the hottest concepts coming out of the software engineering community. In general terms, agile processes are collaboration-intensive styles of specifying, designing, building, and testing software. To guarantee a high level of collaboration, agile processes frequently involve team-based and pair-based activities. Agile proponents prefer face-to-face communication to the process of swapping documents among team members.

Test-driven development (TDD) is an important topic to consider if you're serious about agile software development. TDD involves running tests upfront in order to specify target requirements and to describe a target design (see Figure 1a). This contrasts to test-last development (TLD), which is a more traditional procedure. TLD involves requirement gathering and project design prior to the process of running software tests (see Figure 1b).

TDD doesn't just let you get on with the coding. The requirements are analyzed, but analysis is performed in close consultation with a customer who provides a series of tests. You then write the simplest possible code to satisfy the customer's tests. TDD tests should be automated and easy to run, so you're encouraged to use them frequently during the project's various stages of development. Prior to developing code, you must generate additional tests. With the code working, the tests will be used like traditional post-development unit tests. And with the product now capable of meeting your customer's requirements, code development moves in a new direction. You can change, or "refactor," the code, making modifications for reliability rather than functionality. In this phase of the project, the mode for using the tests developed upfront has switched to something that looks more conventional: post-production testing.

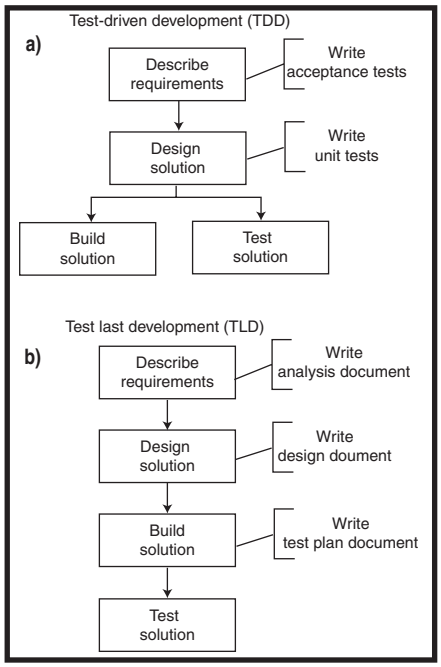


Figure 1a—*The requirements analysis and design solution are written in terms of "tests to be run" for test-driven development (TDD).* **b**—*Test-last development (TLD) requires the production of numerous documents before the majority of testing starts.*

Upfront test development sounds like a good idea, especially because the all-important customer and developer tests aren't left until the end of the project. However, designers have different feelings about which technique, TDD or TLD, is the most efficient (highest developer productivity). Which technique provides the best code (reduced number of post-release problems)? Which technique best handles the numerous other customer- and developer-related issues that arise during code production and delivery?

Many one-off project amateurs write tests for their code, but the tests tend to be in a debug format rather than a formal testing format (see Listing 1). Other tests are of a more practical nature, but they're still informal. Our design team includes both one-off project amateurs and software project developers who have considerable industrial experience with testing environments like the popular xUNIT frameworks (JUnit, NUnit, CppUnit, etc.). However, such big desktop systems have mega-megabytes for running a testing environment. As a result, they aren't limited by the mem-

Listing 1—*Many one-off project developers use debug statements to provide a simple informal test structure.*

```
#define USE_TESTS
    value = Process_Array(array1, array2);
#if defined(USE_TESTS)
    printf("Value at this point %d", value);
#endif
```

ory and real-time constraints found in an embedded systems environment.

In this series of articles, we'll show you how to use TDD to develop software for an embedded system. We chose a simple engineering project for high school students as the target for our TDD investigation. This month we'll show you how to use a TDD environment to develop a video game with the Analog Devices ADSP-BF533 Blackfin microcontroller. Our thermal arm wrestling project was perfect because it was easy to set up, was a little flashy, wasn't too overpowering, and was just corny enough to interest high school students. Next month we'll show you how to adapt an open-source TDD tool to work within a constrained system environment.

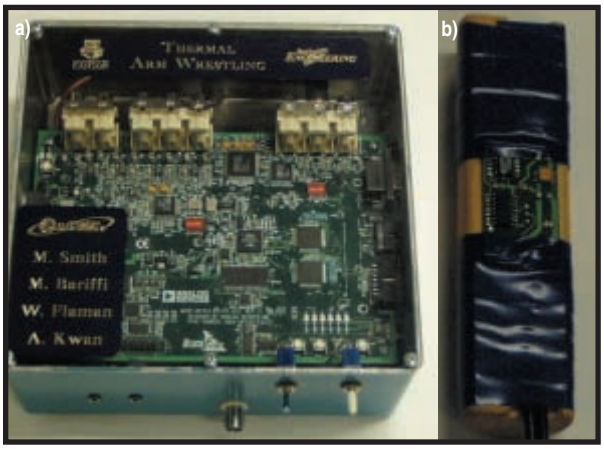


Photo 1a—*Take a look at the game box* **(a)** *and paddle* **(b)**. *Each game paddle consists of a pulse-width modulated thermal sensor (TMP03) and voltage protection hardware.*

PROJECT BASICS

The thumbs of two competitors warm the two pulse-width modulated thermal sensors (Analog Devices's TMP03) in our thermal arm wrestling project (see Photo 1). The sensor signals are fed to an embedded system and converted to temperature values. The difference in temperature between the two competitors is used to control a cursor on a TV screen (see Photo 2).

To introduce an engineering design element into the game, two game strategies were provided for each competitor. You can thermally overpower your opponent and move the cursor into your hot winning area. Alternately, you can psych-out your opponent by having the sensor cool down; this causes the cursor to overshoot your opponent's hot winning area into your cold winning area.

TDD ENVIRONMENT

Prior to developing the game, we had been modifying a C++ language automated testing environment to run with-



Photo 2—*The game screen is generated as a background video task running on an ADSP-BF533 Blackfin EZ-Lite evaluation board.*

Networking Multiple Serial Devices? It's Easy

The Multi-Port Serial-to-Ethernet Application Kit

Enables you to simultaneously connect up to 4 serial devices to an Ethernet network from a single hardware module. Application samples demonstrate the Serial-to-Ethernet, Ethernet-to-Serial, and data collection tasks. Additional application samples show you how to monitor and control your device from a web page (RabbitWeb purchase required).

Complete Application Kit Based on an RCM3700 RabbitCore™

The Multi-Port Serial-to-Ethernet Application Kit provides all the hardware and software to network sensors and other serial devices. This kit includes all the hardware, software, and power that you need to immediately demonstrate Serial-to-Ethernet conversion. Optional software modules provide Web Access and Control (RabbitWeb), Security (SSL), Encryption (AES), and Data Transfer and Storage (FAT, PPP).

Network Your Sensors

Purchase this kit and receive a coupon for a $50 discount on the web-simplifying software module, RabbitWeb.

www.rabbit4networking.com

KIT PRICE $399



Application Kit Features

- Software support for up to 6 serial ports
- Serial-to-Ethernet conversion
- Ethernet-to-serial conversion
- Royalty-free TCP/IP stack
- RCM3700 RabbitCore
- Full prototyping board
- Temperature sensor demo board
- Dynamic C development environment

RABBIT Semiconductor

2932 Spafford Street, Davis, CA 95616
Tel 530.757.8400

9543

in the available memory space on an embedded system. We chose to adapt the small, non-scripted CppUnitLite, which was developed by Michael Feathers and distributed under the terms and conditions of the GNU license (http://c2.com/cgi/wiki?CppUnitLite). Non-scripted means that the framework detects the tests that will run automatically. This frees you from having to write a script that specifically runs the appropriate tests.

We added specialized embedded system-specific capabilities. These included the ability to watch (in real time) the performance of specific processor peripheral registers during system initialization and time-critical sections of code. Additional tests were added to overcome an unexpected problem when testing unsigned short variables. Such variables are used extensively when programming devices with application registers of various byte sizes. Next month we'll provide the details of the current TDD environment tested on the integer Blackfin (ADSP-BF5*xx*) and VLIW TigerSHARC (ADSP-TS*xxx*). Refer to our TDD on Embedded Systems web site for more information (www.enel.ucalgary.ca/People/Smith/embeddedTDD/).

CUSTOMER TEST

In a real TDD environment, we could immediately talk to a customer and develop customer tests for our code to satisfy. But we're neophytes at this TDD game. The best customer requirement tests we could develop are shown in Listing 2. `main()` calls `run_tests();`, which is a key element within the TDD environment. Figuratively, `run_tests()` sets up a list of available tests (customer, unit, developer, expert, etc.) and runs them. `TEST()` and `CHECK()` are examples of the available test macros. The macro format permits descriptive strings to be automatically constructed during the program compilation to provide information about which successes and failures are being reported. Automation and ease of use are key elements if you're actually going to use the testing environment.

There will be four failures when the test runs. This is an expected feature of TDD at this point in the hardware/software codesign process. The failures occur because code and hardware aren't available to satisfy the test. This is, after all, TDD where the tests are written before the code. The single pass is essentially an indication that the test has run (simple positive reinforcement).

About the only truly positive things that have to do with the customer test in Listing 2 are that it's automated and one item (ease of use) is tested in two ways. A key point of the test, however, is a clear demonstration of a weak point in TDD. It's extremely difficult, even for experienced coders, to get into the mind of the customer and out of the old TLD mode. The translation of customer requirements directly into test cases is one of the skills that you must pick up in order get the full advantage of the TDD concept.

Let's accept that this initial set of test cases isn't correct, complete, or validated. However, we had a conversation with the customer regarding the requirements—a key part of TDD. So, let's investigate the advantages (or otherwise) of TDD in a context with which you're more familiar: developing the code.

Making the typical developer's best guesstimate of what a customer really wants, we could proceed with the overall system requirements and high-level design. By describing the target architecture, we used tests to guide our work. We also used tests to describe the intended behavior of the target components as they interact to satisfy the perceived customer requirements. Because these were lower-level tests and the customer may not have been aware of the need for them, the tests were referred to as developer tests.

Listing 2—*Expressing customer requirements in the form of a test can be difficult. There are four failures because code (other than function stubs) didn't exist at the time. The single pass was associated with the fact that the test series ran. The test was automated. One item was tested in two different ways. That must count for something!*

```
File main.cpp
*********************************************************
#include "TDD_Environment.h"

void run_tests(int output_code, int tests_code);
int main(void) {
       int output = SCREEN_OUTPUT | SHOW_FAILURES;
       int test_code = ALL_TESTS;

       run_tests(output, test_code);
       return 0;
}
*********************************************************
File cutomertests.cpp
*********************************************************
TEST(REQUIREMENTS, CUSTOMER) {
char buff[80];
// Software tests
    CHECK(true);              // Able to link to the test code

    puts("Did the software crash? – Y/N");
    gets(buff);
    CHECK(buff[0] = = "N");    // Reliability test
// Hardware tests
    puts("Are there blinking lights? – Y/N");
    gets(buff);
    CHECK(buff[0] = = "Y");    // Interesting

    puts("Did the power plug fit? – Y/N");
    gets(buff);
    CHECK(buff[0] = = "Y");    // Easy to set up (test 1)

    puts("A cable to connect the TV available?– Y/N");
    gets(buff);
    CHECK(buff[0] = = "Y");    // Easy to set up (test 2)
}

TEST:CUSTOMER: REQUIREMENTS:
One Test: Assert Statistics: 4 failed, 1 passed
```

(The developer identifies the test cases and the expected results.)

Exploiting the synergy between customer tests and developer tests is another one of those hard-to-learn skills needed for effective TDD. Given that our problem was small in this particular example, it was unlikely that the relationship between customer and developer tests would have had too much of an effect on the way we needed to proceed.

GAME PADDLES

The output from the ADSP-TMP03 thermosensors is basically a 33-Hz logic-level (5 V) square wave whose duty cycle varies with temperature (see Figure 2). You can calculate the temperature with the following formula:

$$A - B\left|\frac{\text{TIME_HIGH}}{\text{TIME_LOW}}\right|$$

A key feature of the device is that it depends on the ratio of the times for the high and low signal cycles. This means that the repeatability of measured times is more important than the absolute accuracy of them. The constants (*A* and *B*) depend on whether you want the output in Celsius or Fahrenheit. (Let's think ahead, we might want to sell this game in Canadian and U.S. markets!) Because we are dealing with simple logic levels (0 and 5 V), a simple digital signal input must be analyzed rather than an analog signal or digital values from a parallel interface.

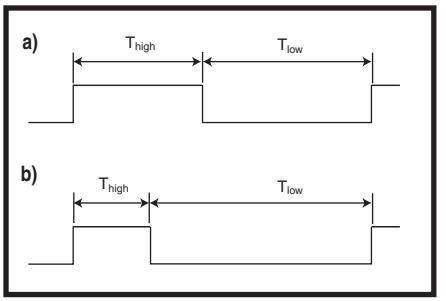


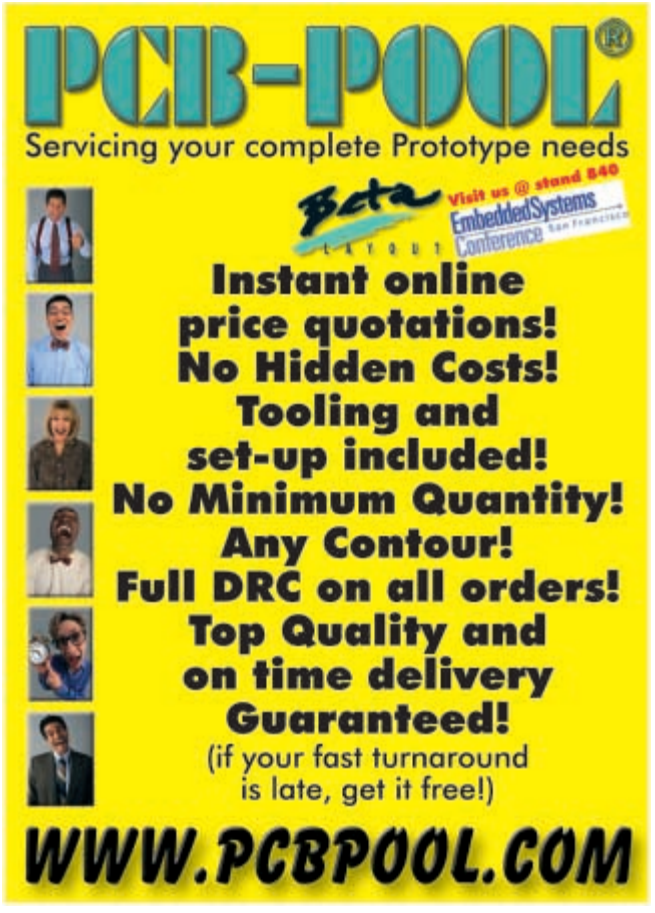
Figure 2—*The timing diagram for the ADSP-TMP03 thermosensor output shows the results for lower* ***(a)*** *and higher* ***(b)*** *temperatures. As the thermosensor temperature increases, the duty cycle of the square wave changes (although not as dramatically as shown here).*

VIDEO REQUIREMENTS

Our long-term goal has been to develop video images on the fly and output images that change in response to information from the game paddles. However, we didn't know how to generate and display images simultaneously. To keep things simple, we stored 30 predefined images in the evaluation board's SRAM. Each image had the same background, but the cursor was in slightly different positions. We used DMA-controlled interrupts to transfer different images to the TV screen on demand. This was essentially a multiple buffer approach to handling the game screen. It was the simplest possible solution.

DEVELOPMENT SYSTEM

We chose the ADSP-BF533 (Blackfin) EZ-Lite evaluation board for several reasons. First, its flexibility and availability made it an ideal target for the thermal-arm wrestling project. Second, it has available SRAM for storage of images, and its flash memory enabled simple updates and automatic boot-up without an external control computer. Third, the board has programmable flags (PF) for digital input and available audio and video





signals. The second version of the game will involve using phase shifts to control the apparent position of a sound source in response to temperature changes from the game paddles. Nintendo, here we come!

Finally, note that the processor has a lot of speed. We can perform most of the prototyping in optimized C++ language and anticipate not having to worry about timing issues (moving code into parallel assembly). This is important if you're short on time. Details of the current TDD environment tested on Blackfin (ADSP-BF5*xx*) and VLIW TigerSHARC (ADSP-TS*xxx*) processors can be found at the aforementioned TDD on Embedded Systems web site.

CODESIGN WITH TDD

Although it may be difficult to develop appropriate customer tests ahead of time, it isn't the case for developer tests. You'll have a specific result in mind before you develop the code. Whether you develop a test for this result before or after the code is a matter of preference.

The key TDD tasks involve writing the simplest code to satisfy the customer test and refactoring the code to improve reliability rather than functionality. The first element makes sense. Refactoring sounds good, but if it isn't broke, why fix it? Nevertheless, it would be nice to have a test around just in case you want to modify, rather than fix, existing code. After all, you must expect program modifications to be suggested during customer consultations. In addition, because tests can be created and run so easily, developers will be encouraged to use more tests in the battle to develop software. An additional advantage of having proper customer tests available is that unrecognized scope creep is less likely to happen.

Most of the code for generating and displaying the signals with the video codec on the evaluation board is a modification of provided examples. If the picture on the screen doesn't flicker, then you can assume everything is fine. Just remember that you must eventually generate a series of acceptance tests to test the reused code's reliability.

The code for controlling the game paddles revolves around nonstandard software and hardware components. It is more critical that you write customized tests for them.

Listing 3—*The tests were developed for the* `CalculateTemperature( )` *function. Note that both final tests were for possible failure modes. They were generated in addition to expected successful tests. Note that the tests also indicated certain basic design decisions like the format of temperature values (integers with values that represent the temperature to the nearest quarter of a degree).*

```
#define CELSIUS 1
#define FAHRENHEIT 2
#define GARBAGE 3

typedef temperatureValue4 signed short;

TEST(CELSIUS_20, CalculateTemperature) {
// Time high/low values for 20°C
     temperatureValue4 value;
     value = CalculateTemperature( 400, 215, CELSIUS);
     SHORTS_EQUAL(value, 80);
}

TEST(CELSIUS_20_5, CalculateTemperature) { // 20.5 C
     temperatureValue4 value;
     value = CalculateTemperature( 4000, 2145, CELSIUS);
     SHORTS_EQUAL(value, 82);
}

TEST(FAHRENHEIT_NEG20, CalculateTemperature) {
// Time values for -20 F – special Canadian test
     temperatureValue4 value;
     value = CalculateTemperature( 720, 475, FAHRENHEIT);
     SHORTS_EQUAL(value, 0xFFb0);
}
// Error tests
TEST(GARBAGE_ERROR, CalculateTemperature) {
     temperatureValue4 value;
     value = CalculateTemperature(7200, 4745, GARBAGE);
     SHORTS_EQUAL(value, TEMPERATURE_ERROR_CODE);
}

TEST(tempZERO, CalculateTemperature) {
     temperatureValue4 value;
     value = CalculateTemperature( 0, 0, CELSIUS);
     SHORTS_EQUAL(value,TEMPERATURE_ERROR_CODE);
}
```

TESTS

We used the ADSP-TMP03 temperature sensor datasheet and a calculator to design a series of tests for the major software component:

```
CalculateTemperature(hightime,
  lowtime, mode)
```

The mode is either Celsius or Fahrenheit. The high time and low time are the periods obtained by measuring the transitions in the temperature sensor signals (see Figure 2). The tests are shown in Listing 3.

You must generate tests to measure the expected successes and the possible error modes. Note that certain design decisions are specifically expressed within the test code. Some decisions are simple, and others are more complicated. The simple ones involve defining the constants. One of the more complex is the `temperatureValue4` definition, which indicates that temperatures will be stored as block floating-point integer temperatures returned to the nearest quarter of a degree (21.5°C returns as the short integer value 82).

The top section of Listing 4 shows the obvious way to write the code to pass the tests (doing the calculation in floating-point math). The software floating-point emulation mode is slow on an integer processor, but the Blackfin has a high system clock frequency, and you only have a simple task with limited calculation. This particular code doesn't pass all the tests because of the way it has been written. Temperature values like 20.5

Listing 4—*The top section of code didn't satisfy all the tests, because the cast from the floating-point math occurred at the wrong time and introduced a loss of precision. The lower code was a refactored version that provided the required precision. It was more appropriate for the integer Blackfin processor.*

```
temperatureValue4  CalculateTemperature(
        short int lowTime, short int highTime,
        const unsigned short int celsiusFahrenheit) {

    float temperature;

    if (lowTime == 0)
        temperature =  TEMPERATURE_ERROR_CODE;
    else if (celsiusFahrenheit = = CELSIUS) {
              temperature =
            235 - 400 * (float) highTime / (float) lowTime;
            return 4 * (temperatureValue4)  temperature;
    }
    else if (celsiusFahrenheit = = FAHRENHEIT) {
     temperature =
            455 - 720 * (float) highTime / (float) lowTime;
            return 4 *  (temperatureValue4) temperature;
    }

    return  TEMPERATURE_ERROR_CODE;
}

temperatureValue4  CalculateTemperature(
            short int lowTime, short int highTime,
            const unsigned short int celsiusFahrenheit) {

    temperatureValue4 temperature;

    if (lowTime == 0)
            temperature = TEMPERATURE_ERROR_CODE;
    else if (celsiusFahrenheit = = CELSIUS)
            temperature = (short int) (4 * 235 - 1600 * highTime /
lowTime);
    else if (celsiusFahrenheit = = FAHRENHEIT)
            temperature =  (short int) (4 * 455 - 720 * 4 *
highTime / lowTime);
    else temperature = TEMPERATURE_ERROR_CODE;

    return temperature;
}
```

are incorrectly calculated, because the conversion from floating point to integer occurs prior to multiplication by four (to give the intended quarter-degree accuracy). This information is revealed after you run the predefined tests.

The lower section of Listing 4 shows the refactored code taking into account the fact that floating-point emulation on the integer Blackfin processor is slow. Note that the code passes the tests, but it doesn't necessarily indicate that the code is correct. Additional tests may be required.

ADAPTING TDD

As you can see, there are many advantages to TDD, which is a primary component of the agile development technique. It's intended to minimize the amount of upfront documentation associated with customer requirements and developer testing in favor of up-front tests that offer improved precision and less ambiguity.

Using a video demonstration running on the Blackfin ADSP-BF533 evaluation board, we showed you that it takes more than 5 min. to replace your developer's hat with your customer's hat. You can use the TDD environment for developing tests before coding, or you can use it as an automated testing environment if you prefer to test after coding.

Next month we'll take a closer look at the process of adapting an automated TDD environment for embedded systems. You'll learn how to use it for real hardware testing.

Authors' note: We'd like to thank Mimi Pichey, Bob Miessenholder, and Kamran Sharifi of Analog Devices for providing hardware and software support. The Natural Sciences and Engineering Research Council (NSERC) of Canada and the University of Calgary funded components of this project.

Mike Smith has been writing computer-related articles since the early 1970s. He is a professor of electrical and computer engineering at the University of Calgary, Canada. He develops and tests algorithms for the real-time analysis of stroke images from magnetic resonance imaging. He has been the Analog Devices University Ambassador since 2002. You may contact him at smithmr@ucalgary.ca.

Moreno Bariffi is an international internship student visiting from the University of Applied Science, Fribourg, Switzerland. You may contact him at mbariffi@hotmail.com.

Warren Flaman is an electronics technician at the University of Calgary. You can reach him at wflaman@ucalgary.ca.

Adam Geras is a graduate student at the University of Calgary. You can reach him at ageras@ucalgary.ca.

Lily Huang is a graduate student at the University of Calgary. You may contact her at fhuang@ucalgary.ca.

Andrew Kwan is a student at the University of Calgary. You may contact him at akckwan@ucalgary.ca.

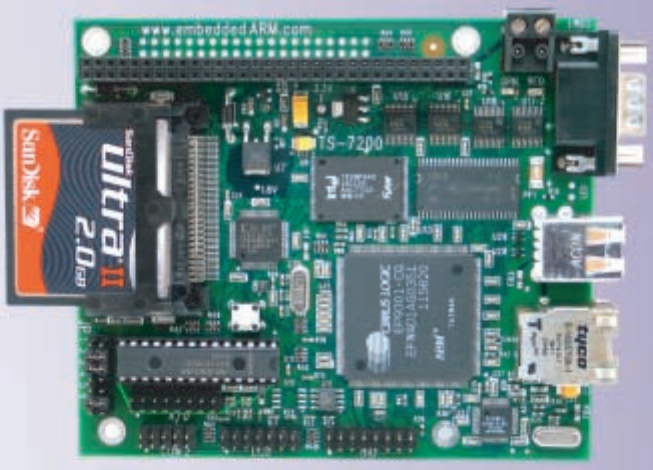
Alan Martin is an electrical engineering student at the University of Calgary. You can reach him at ajcmarti@ucalgary.ca.

James Miller is a professor of electrical and computer engineering at the University of Alberta, Canada. His interests include the development of small self-contained embedded systems and software-testing procedures. You may contact him at jm@ee.ualberta.ca.

SOURCE

ADSP-BF533 Blackfin microcontroller, TigerSHARC family, and TMP03
Analog Devices
www.analog.com

200 MHz ARM - PC/104



Shown with optional A/D converter, Compact Flash and RS-485.

TS-7200

$149
$119 qty 100

- 32 MB SDRAM (64 MB optional)
- 8 MB Flash (16 MB optional)
- Compact Flash socket
- 10/100 Ethernet
- 2 USB ports
- 20 Digital I/O
- 2 Serial Ports
- 2 ch. 12 bit A/D (optional 10 channels)
- optional USB WiFi adapter

- EP9302 ARM9 processor with MMU
- Boots TS-Linux from on-board flash in 5 seconds
- TS-Linux for ARM is open source GNU/Linux
- Low power +5v @ 400mA ▪ 3.8" x 4.5"

Linux 2.4 NetBSD

200 MHz ARM - PC/104



Shown with optional A/D converter, USB Flash drive and RS-485.

TS-7250

$149
$119 qty 100

- 32 MB SDRAM (64, 128 MB optional)
- 32 MB Flash (128 MB optional)
- 10/100 Ethernet
- 2 USB ports
- 20 Digital I/O
- 2 Serial Ports
- 5 ch. 12 bit A/D (optional 13 channels)
- optional USB WiFi adapter

- EP9302 ARM9 processor with MMU
- Boots TS-Linux from on-board flash
- TS-Linux for ARM is open source GNU/Linux
- Low power +5v @ 400mA ▪ 3.8" x 4.5"

The 256 MB USB drive shown holds our Debian 3.0 self-hosting development environment for ARM. Booting to the development file system allows code development to take place directly on the target hardware eliminating any issues with transfering newly compiled files from the desktop to the target device.

Cross-compiler tools are also available for Windows and Linux.

www.embeddedx86.com

(480)-837-5200
16610 E. Laser Drive #10
Fountain Hills, AZ, 85268 USA

Technologic SYSTEMS

133 MHz 5x86 - PC/104

TS-5600 $279



- PC/104
- PCMCIA slot
- 10/100 Ethernet
- SPI bus
- 2 Serial Ports
- Compact Flash
- 24 Digital I/O
- 12-bit 8 chan. A/D option

- AMD Elan 133 MHz 586 with 32 MB RAM (64 MB optional)
- PC compatible: BIOS, DOS, RTC, I/O & memory map
- TS-Linux 3.0 is open source GNU/Linux - Glibc 2.2.5
- Power +5v @ 830mA
- 4.3" x 5.6"

Shown with optional flash modules, A/D, RS-485 and Merlin cellular modem

33 MHz 386 - PC/104

TS-3400 $159



- PC/104
- Low EMI
- DiskOnChip
- DiskOnChip
- 37 Digital I/O
- 2 Serial Ports

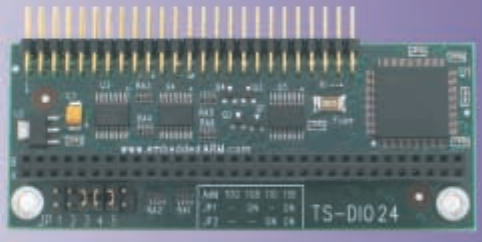
- 33 MHz Intel 386EX with 8 MB RAM (16 MB optional)
- DOS-ROM or TS-Linux pre-installed
- PC compatible: BIOS, DOS, RTC, I/O & memory map
- 1 MB flash drive
- Power: +5v @ 500 mA
- Dimensions 4.0" x 3.8"

Shown with optional flash module

24 Point Digital I/O Board

TS-DIO24 $39

- PC/104 peripheral board
- Opto-22 compatible 50 pin conn.
- 4 outputs drive up to 48 mA
- 4 outputs drive up to 1 Amp
- 16 programmable I/O points
- I/O address jumper selectable
- 4 inputs are IRQ capable



Technologic SYSTEMS

- PC/104 Computer Boards
- Enclosures
- PC/104 Peripherals
- Custom Designs
- Exceptional Support

PC/104 Peripherals

Enclosures $85

Metal enclosure for Single Board Computers provides regulated +5 VDC from 12-38 VDC input power. Connections available at back panel. Suitable for office or factory.

TS-SER4 $89

Four PC compatible COM ports using the 16550 UART, full handshaking on two ports, RS-485 option on two ports, jumper configurable for COM1-24 and IRQ's 3-15.

TS-ETH10 $59

10 base-T Ethernet daughter board using Cirrus CS8900 Ethernet chip, features indicator LED's, jumper configurable, comes with Linux and DOS drivers.

TS-Modem $129

33.6K baud, jumper configurable as COM1-6, supports IRQs 3-15, V.32 V.42 compliant, standard AT commands, caller ID, 56K baud optional.

TS-Linux v3.0

- Introducing TS-Linux version 3.0 Our latest distribution is built from open-source featuring a 2.4.18 kernel, Apache web server, FTP client/server, bash, etc. With a footprint of 15 MB or less this is a great place to start your application.
- The TS development environment compiles application code using the libraries and headers installed in TS-Linux. Applications developed on your desktop computer copy directly.
- Our User Mode Linux patched kernel runs TS-Linux 3.0 kernel and file system as an application on your desktop computer during development.

Zeroing in on ZigBee (Part 2)

Chipsets and Source Code

Last month Pete introduced the ZigBee low-power radio standard and underlying IEEE 802.15.4 layers. In this article, he describes the development resources you'll need to create your own ZigBee projects. He covers both the chipsets and the source code.

Last month I described the ZigBee low-power radio standard and the underlying IEEE 802.15.4 layers. Now I'll focus on the development resources you can use in your own ZigBee projects. I'll describe affordable chipsets and source code.

SIMPLE PROTOCOL

ZigBee exhibits robust, reliable communications through an auto-configuring mesh network architecture. You'll soon be able to concentrate on the application by relying on ZigBee to provide the transceiver, RF channel, and protocol. This is achieved with only one transceiver IC incorporating the physical layer and software on the microcontroller that provides part of the medium access controller and higher layers. These layers coexist with the application running in the same code space.

The protocol is simple and efficient. You can use a medium-powered, 8-bit microcontroller. As little as 16 KB of flash memory is required for a simple device. ZigBee was designed for extremely long battery life. It will eventually enable home automation, cable replacement, and industrial sensors.

USABILITY

You've seen press releases stating that some new IC has been tested in an exciting new area of electronics. Frequently, however, access to the new technology is blocked, because the samples are only for approved development partners only, or because of the sheer complexity of the devices. Let's take a look at how to obtain affordable ZigBee-compatible hardware and software and use it at the chip level.

If you can place the transceiver and controller ICs on your own PCB in a suitable way for use with an 8-bit microcontroller, you can markedly reduce the cost of a project. This is desirable because a typical application for which ZigBee is suited may be comprised of many nodes. Using the naked IC on your own PCB makes key fob-size projects a reality.

Sometimes it seems as though the growing complexity of these evermore capable standards are moving beyond the scope of designers with limited amounts of time and test equipment. Would you leap into a quick 802.11.x project on a rainy Sunday afternoon given the bare chipset and a soldering iron? Even using a ready-made solution with a PCMCIA card interface requires 30-odd I/O lines.

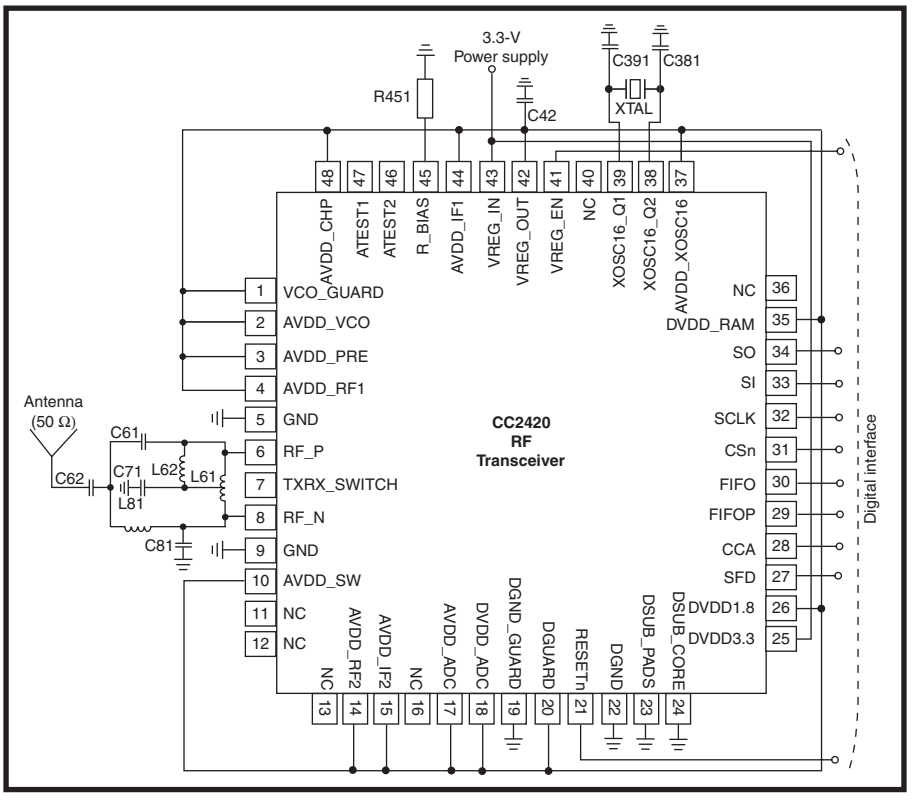


Figure 1—*Sure, you say, here comes another one of those single-chip transceivers that somehow has managed to morph itself into a complicated, messy RF jungle by the time it makes its way from the sales brochure to the application diagram. With the CC2420, however, this promise is almost entirely kept.*[1]

The packaging is also against you because of the advent of ball grid array packaging. It's nearly impossible to work at the chip level with chip-scale packing. I define the birth of a new standard when the average electronics designer can buy a product sample in a leaded package at a reasonable low-quantity price without having to rely on $100,000 of RF test equipment to get it working. The CC2420 was the first product to deliver that ability on the workbench.

CC2420 802.15.4 KIT

Chipcon boasts that the CC2420 is the first transceiver IC to meet both the IEEE 802.15.4 and ZigBee standards. Of course, there's meeting the standards, and then there's the ability to produce a workable cost-effective product. After using the evaluation kit, I can say that Chipcon has produced both in the first version of silicon.

Chipcon's hardware support and software tools keep you ahead of the complexity curve. The single-chip implementation limits the RF circuitry to just the antenna and its matching network (see Figure 1). The PC-based SmartRF Studio software makes configuring and using the evaluation kit a breeze.

Fortunately, soldering the CC2420 (although fairly tiny) is just under that critical threshold where doing it yourself is still doable. The CC2420 QLP-48 package with a 0.5-mm lead pitch is a challenge, but it's one I've been able to cope with. Products like PCB iron-on artwork transfer paper has allowed amateur PCB makers like me to easily produce tracks and clearances with 10-mil (0.01") track widths and clearance. A friend of mine regularly does 7 mils with a clothes iron and a slosh of etching acid!

As well as the physical challenge of the intricate surface-mount packages, software drivers are the other big impediment to experimenting at home. Often, a lack of information in the datasheet and the need to program 100 registers consigns hopeful efforts to the waste bin. When success is achieved and the results are posted on a web site, the person responsible is rightly proclaimed a hero.

ZigBee, or even a full 802.15.4 link, would take more than an afternoon to develop your own software for, but you'd be able to get something useful fairly quickly. To prove the RF link, simply use the serial port to confirm two-way serial communication with the CC2420. A simple register setting then places it in Transmit or Receive Test mode, so you can check if the hardware works.

Making a simple one-way, point-to-point link is the next step. Then, a simple two-link is accomplished by implementing mode control by manipulating the CC2420 state machine. If you need more than a master/slave relationship, add a simple MAC protocol such as aloha (time slots) or check for a clear channel before transmission. You can add more features at this point. Encryption, link quality indication, and dynamic power output level control would be nice. But wait. This is starting to sound like 802.15.4! Why don't you just download the free code from the Chipcon web site?

Most of what you need is included with the CC2420DBK demonstration board kit. You can download the files from Chipcon to make your own PCB. The schematic, BOM, and PCB layout



are all there. You can use the free source code for the IEEE 802.15.4 layer. If you don't want to make the PCBs yourself, they're available on the web site for $50. The software isn't a full implementation of the IEEE 802.15.4 MAC, but it's enough for basic message sending and acknowledgment. The full IEEE 802.15.4 MAC software is free if you sign a licensing agreement. Although both versions were written in C for the Atmel ATmega128L microcontroller, they port easily to other microcontrollers.

The easiest approach would be to copy the PCB exactly and use the free GNU AVR gcc cross-development tool. Chipcon strongly recommends that you copy the reference design as closely as possible. If you want to develop a two-layer PCB, you should keep the ground via positions as close as possible to the location on the original board. Of course, FCC licensing is another story. The first stage is to get to a working prototype!

The documentation for the CC2420DBK demonstration board kit states that practical testing outdoors with line of sight (LOS) resulted in a range of 170 m at 250 kbps (zikes!). The simplicity of this information surprised me, because IC vendors usually provide ambiguous range performance information.

I actually used the CC2420DK development kit for range checking. It consists of a pair of motherboards (CC2400EB) and their respective transceiver plug-in boards (CC2400EM), as you can see in Photo 1.

I was initially discouraged by the presence of a Xilinx Spartan II FPGA on the motherboard. Why so complex? But my disappointment vanished when I discovered that the FPGA is only used to configure the evaluation board in a variety of different setups. The FPGA routes signals so that the on-board microcontroller controls the plug-in transceiver board. In another configuration, the plug-in transceiver board connections are routed via the headers to your processor board when prototyping with your own microcontroller.

The process of setting up, including the installation of the SmartRF Studio evaluation software on the PC, took less than 20 min. I found that it wouldn't work on some of my USB ports. Apart from this, things went extremely well. Getting the two boards set up for a range check was as simple as selecting Normal View, Packet TX, Start Packet Transmit on one of the boards, and Start Packet Receive on the other. I then checked for the lights that flash after a reception. Link quality statistics were displayed in real time on the PC. The CRC and packet count information was great, but the RSSI didn't work. A separate piece of software can be used for packet sniffing.

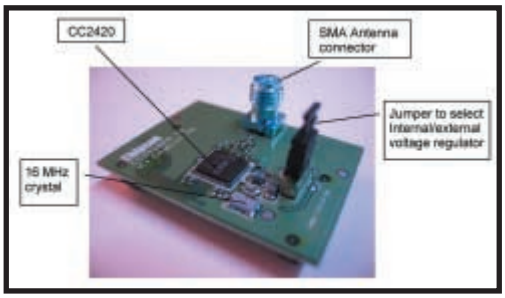


Photo 1—*Your next 802.15.4 project will look this simple. The IC is 7 mm x 7 mm. Chipcon encourages you to copy the PCB layout and adapt it for your own needs. You need a four-layer PCB to guarantee the quoted performance.*[2]

I unplugged the board configured for reception, performed a walk-around range check, and continued to look for the flashing of the LEDs. Using this technique, I found reliable reception everywhere in my 1,100-square-foot, single-level wood house. Reliable transmission occurred when transmitting through three walls with the doors closed, even when partially shielding the receiver antenna with my hand and placing it right on top of other operating electronic equipment.

Taking it outside, I found reliable reception 90' away through one wooden wall out to the backyard. Walking down the street, I got intermittent operation as far as 100 yards away from the transmission end inside my house. I even got reliable reception from 12' away after disconnecting the antenna on the receiver!

The version of the IC I tried suffered from excessive current draw in Shutdown mode when the internal 1.8-V regulator was in use. This was corrected in a later silicon revision, along with an address recognition problem.

Now let's take a look at a simple software application example. A pair of the CC2420DB evaluation kits running the example program will establish a point-to-point RF link, sending data packets containing a 5-byte payload (see Listing 1). The payload will be transmitted when the potentiometer is turned, or the switch, S2, is held down. The first byte of the payload contains the potentiometer value, which controls the PWM duty cycle of the LED at the receiving node. This example isn't a full-blown ZigBee application, or even a proper 802.15.4 implementation, but it illustrates the basics of doing something useful with the hardware. (Well, assuming blinking LEDs enthrall you!)

Listing 2 shows how pressing the button and noting the status of the yellow and green LEDs can perform a simple range check. This is a simple but useful example. To keep things simple, only a small subset of the IEEE 802.15.4 standard is implemented in this example. A pair of CC2420DBs running this program will establish a point-to-point link on a single channel. The frame format conforms to 802.15.4, but it doesn't implement any other MAC properties. The full IEEE 802.15.4 MAC software can be obtained for free after signing a licensing agreement.

DIFFERENT CORE CHIPSETS

Now let's review some of the transceivers implementing 802.15.4 and ZigBee standards. I'll concentrate on describing each of the different core chipsets rather than commenting on each implementation.

More than a few companies use the Chipcon 2420. I'm sure there are good and bad implementations out there. So, the following review will be beneficial if you want to muck in at the chip level rather than the module level. The advantage is that if you place one of the chips on the PCB, you're looking at $15 for the entire transceiver in low quantities, assuming you are already doing your own PCB for the project.

I'll also describe some alternatives to implementing a strict ZigBee solution on the transceiver ICs. The topic of this article is cutting edge. The ZigBee Alliance ratified the upper layers of the ZigBee standard in October 2004.

The 802.15.4 portion is more firmly established. There are alternatives to the upper layers of ZigBee if you want an easier ride until ZigBee is more freely available to people who are on a budget.

Let's pick up where I left off by focusing on alternative protocols that can run on the CC2420. Remember, chips like the CC2420 only give you the physical and MAC support layers. Chipcon is going to package the CC2420 with a ZigBee software stack for $2.30 per unit in large quantities. I'm also excited about the upcoming launch of the CC2430, which will include a microcontroller, RAM, and flash ROM in the same package as the RF transceiver.

An alternative to ZigBee is to choose your own simplified protocol or buy one off the shelf. One such alternative comes from Moteiv, a company founded by three Berkeley students who worked on the open-source TinyOS wireless network system. Their Telos product with a Texas Instruments MSP430 microcontroller doesn't require any programming boards or development software. It has integrated USB and the tools for compiling. It's approximately $130, but offers an instant on approach backed by the availability of open source code.

Ember's EM2420 was co-developed with Chipcon. It's the same IC, but it isn't currently available to the general public in low quantities. According to the datasheet, the chip is only available with a licensed Ember networking stack. It's targeted to approved 8-bit processors.[3] This means Ember only has relationships with serious developers. It can't sell small quantities of hardware for casual developers to play around with. Still, it may ship one to you if you can come up with $13,950 for the EM1020 developer kit (plus shipping!).

The 802.15.4 hardware in the form of the EM2420 and CC2420 was ready long before the ZigBee standard was ratified. This didn't stop companies from developing their own ZigBee-like transport/network layer. Ember's is called EmberNet.

If you want ZigBee included without running the ZigBee layers on your own controller, then Oki Electric Industry Co. might be able to help. Last May it announced the first IEEE802.15.4 and ZigBee single-chip solution. This leaves just the application layer for you to implement. CompX sells PCBs based on this IC.

A key member of the Zigbee Alliance, Freescale recently released the MC13192 2.4-GHz RF transceiver data modem. The 13192DSK evaluation kit is similar in principle to the Chipcon CC2420DBK. You get two nodes for $199. Each node has the MC13192 transceiver and MC9S08GT60 low-power microcontroller. The microcontroller is preloaded with sim-

Listing 1—*A simple blinking LED example grounds some of the theory in easily understood C code courtesy of Chipcon's simple wireless dimmer/RF range tester demonstration. This excerpt shows how your application uses a function prototyped in the RF library to perform an application-specific task based on the contents of the received payload.*

```
// BASIC_RF_RX_INFO* basicRfReceivePacket(BASIC_RF_RX_INFO *pRRI)
// DESCRIPTION: This function is a part of the basic RF library,
// but must be declared by the application. Once the application
// has turned on the receiver, using basicRfReceiveOn(), all
// incoming packets will be received by the FIFOP interrupt
// service routine. When finished, the ISR will call the
// basicRfReceivePacket() function. Please note that this function
// must return quickly, since the next received packet will over
//  write the active BASIC_RF_RX_INFO structure (pointed to by pRRI).
// ARGUMENT: BASIC_RF_RX_INFO * pRRI
// The reception structure, which contains all relevant info
// about the received packet.
// RETURN VALUE:  BASIC_RF_RX_INFO*
// The pointer to the next BASIC_RF_RX_INFO structure to be used
// by the FIFOP ISR.
// If there is only one buffer, then return pRRI.
BASIC_RF_RX_INFO* basicRfReceivePacket(BASIC_RF_RX_INFO *pRRI) {
    // Adjust the led brightness
    PWM0_SET_DUTY_CYCLE(pRRI->pPayload[0]);
    // Blink the green LED
    SET_GLED();
    halWait(10000);
    CLR_GLED();
    // Continue using the (one and only) reception structure
    return pRRI;
} // basicRfReceivePacket
```

Listing 2—*Things are just as simple on the transmission side. This excerpt from the same file as Listing 1 shows you how to respond to events by transmitting information using the* `basicRfSendPacket` *function provided in the RF library.*

```
// The main loop:
while (TRUE) {
    // Sample the pot meter value
    ADC_SAMPLE_SINGLE();
    ADC_GET_SAMPLE_8(ledDutyCycle);
    // If the dimmer value has changed by more than 1, then
    // transmit the new value automatically
    // Transmit also when the S2 button is pressed
    dimmerDifference = (ledDutyCycle & 0xFF) - pTxBuffer[0];
    if((ABS(dimmerDifference)>2)||(JOYSTICK_CENTER_PRESSED())) {
        pTxBuffer[0] = ledDutyCycle;
        if (basicRfSendPacket(&rfTxInfo)) {
            // OK. Blink the yellow LED
            SET_YLED();
            halWait(50000);
            CLR_YLED();
        } else {
            // No acknowledgment received -> Blink the red LED
            SET_RLED();
            halWait(50000);
            CLR_RLED();
        }
    }
}
```

ple MAC (SMAC) software with which you can establish simple point-to-point or star proprietary network topologies. You can also download the 802.15.4 MAC source code for free and use it with higher-level routines. Metrowerks's CodeWarrior development studio for HCS08 microcontrollers is included. The on-board peripherals include two accelerometers, some switches, LEDs, and an RS-232 port for monitoring and flash memory programming.

The MC13192's lack of an on-chip transmit/receive switch is a potential disadvantage. The evaluation kit overcomes this with no increase in the bill of materials by using separate transmit and receive antennas. This less integrated approach results in a smaller package size, but you'd need an external transmit/receive switch in applications requiring a single antenna. The advantage of excluding the internal transmit/receive switch on-chip is that an external power amplifier can be used to boost the signal for greater range. However, both the Freescale and Chipcon ICs already have a transmission power of 1 mW, which is twice the minimum required by the ZigBee specification.

Chipcon, Atmel, and Freescale will bundle their hardware with a ZigBee software stack developed by Figure 8 Wireless. ZMD is another main player with silicon. Its ZMD44101 is in the 868- to 928-MHz bands. ZMD is optimizing for a lower data rate of 40 kbps, which should make it a good choice for devices that require the most reliable communications at the maximum range.

According to William Craig, program manager for wireless communications at ZMD, the IEEE 802.15.4 standard provides for one channel (868.3 MHz) in Europe because of bandwidth limitations. The sub-1-GHz band is desirable for RF characteristics, where range and attenuation are at issue. Even more desirable is an increased frequency range providing more channels for the IEEE 802.15.4. ETSI may consider this requirement, William said. The ZMD44101 Fractional-N RF_PLL design provides for software-controlled frequency selection that will accommodate emerging frequency standards. There is a 1% duty cycle restriction in Europe for 868 MHz, which means that it's currently suitable only for RFD end devices.

Atmel is also initially focusing only on the low bands with its AT86RF210 Z-Link transceiver and matching AT86ZL3201 Z-Link controller. It will offer a 2.4-GHz version of the transceiver IC before developing a single-chip solution.

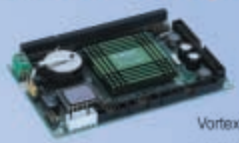
As with Freescale, the ZigBee craze seems to be an excuse to hawk general-purpose components relabeled as "ZigBee-compatible." There are a few special features on these microcontrollers such as hardware AES encryption and a specialized random number generator. However, transceiver chips such as the CC2420 already do this in hardware, thereby allowing you to choose a true general-purpose microcontroller or reuse one from an existing application.

One advantage is that the Atmel two-chip solution turns into a one-stop package for an entire ZigBee solution. The transceiver and microcontroller, including 802.15.4 in ROM and a ZigBee protocol stack sharing the 32-KB

x86 Embedded Processor Module & Single-Board-Computer


Mity-SOC-1
386 Embedded system module
1 RS-232, 1 RS-232/485, 16 GPIO
IDE, FDD, RTC, Parallel, Watchdog
2MB RAM, 2.56" X 1.77" (Optional 4MB)
$65.00 ea. Single unit
$50.75 ea. Qty. 100 pricing


Mity-Mite
386 Embedded system module
1 RS-232, 1-RS-232/485, 16 GPIO,
Ethernet, IDE, RTC, Watchdog, K/B
4MB RAM, 3.14" X 1.96"
$100.00 ea. Single unit
$78.00 ea. Qty. 100 pricing


Vortex86-6082
Vortex86 Embedded system module
1 RS-232, 1-RS-232/485, Parallel, USB
Ethernet, IDE, VGA, K/B, Mouse, RTC,
Watchdog, 128MB RAM, 3.94" X 2.60"
$262.00 ea. Single unit
$182.00 ea. Qty. 100 pricing


Vortex86-6071
PC/104 Vortex86 Embedded SBC
3 RS-232, 1-RS-232/485, Parallel, USB, IDE
Audio, Ethernet, CRT/LCD, RTC, Watchdog
K/B, Mouse, 128MB RAM, 3.77" X 3.54"
$308.00 ea. Single unit
$214.00 ea. Qty. 100 pricing


ICOP-6027VE
3.5" 386 Embedded SBC
CRT/LCD, 1 RS-232, 1 RS-232/485, K/B
Parallel, DiskOnChip, IDE, FDD RTC,
Watchdog, 4MB RAM, 4.01" X 5.67"
$222.00 ea. Single unit
$154.00 ea. Qty. 100 pricing


ICOP-6050
PC/104 386 Embedded SBC
1 RS-232, 1-RS-232/485, 1 parallel, K/B
DiskOnChip, IDE, FDD, RTC, Watchdog
4MB RAM, 3.77" X 3.54"
$142.00 ea. Single unit
$99.00 ea. Qty. 100 pricing


Vortex86-6075
Vortex86 Embedded SBC
1 RS-232, Parallel, 3 USB Ethernet, IDE,
VGA, K/B, Mouse, RTC, Video-in, TV-out
Audio, Watchdog,128MB RAM, 4.37" X 5.24"
$266.00 ea. Single unit
$185.00 ea. Qty. 100 pricing


Vortex86-6047
3.5" Vortex86 Embedded SBC
3 RS-232, 1-RS-232/485, Parallel, USB, IDE
Audio, Ethernet, CRT/LCD, RTC, Watchdog
K/B, Mouse, 128MB RAM, 4.01" X 5.67"
$326.00 ea. Single unit
$226.00 ea. Qty. 100 pricing

Supported OS & Development Environment

DOS
Using C/C++, DOS application can be develop to run on all of our processor modules. DSock, a TCP/IP library for DOS, is provided to develop application with Internet connectivity. Sample implementation for BOOTP/DHCP, FTP, SMTP, HTTP, TELNET & TALK are available for download from our Web site. www.dmp.com.tw/dsock

Embedded Linux
Linux application can run on all of our processor modules. We provide X-Linux, an embedded Linux kernel based on the current popular distribution. X-Linux is a head-less kernel approx. 3MB in size. It includes Linux Kernel 2.4.18, SysLinux Loader, BusyBox Shell, FTP4ALL, udhcp Client, WIN HTTP, glibc & Web based administration.

Windows CE .NET
Vortex86 BSP for Windows CE .NET has been certified by Microsoft Windows CE .NET BSP certification program. Windows CE .NET applications can be develop using Embedded Visual C++, Visual Basic .NET & Visual Studio .NET with Compact .NET Framework library. Please check our Web site for Windows CE .NET SDK information.

ICOP
intelligent control on processor

ICOP Technology Inc.
Tel: (626) 444-6666 Email: info@icoptech.com
URL: www.icoptech.com

Microsoft® Windows® Embedded Partner
ICOP is a Gold-Level-Partner of Microsoft Windows Embedded Partner program. Gold WEP has been created by Microsoft to identify companies with demonstrated expertise to support Windows Embedded Technologies.

Contact us for custom design & OEM/ODM services.
Product and service names mentioned herein are the trademarks of their respective owners. Not responsible for error. Prices are subject to change without prior notice.

Hobbyist? You might be if you used a **SCHEMATIC** to explain something **ON A FIRST DATE**, or your **SOLDERING IRON** is next to the **COFFEE MAKER**. If you wired up the **DOG RUN** with a **TEMPERATURE CONTROLLED MISTING SYSTEM** that activates via **ETHERNET**... and you don't have a dog...

imaginetools.com

Learn programming and networking by doing it!

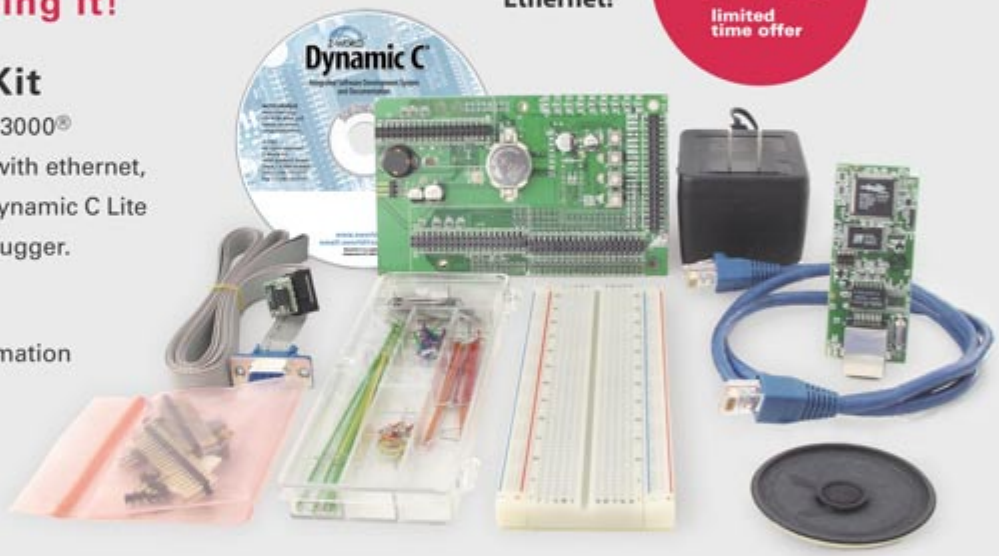
Now With Ethernet!

regular $179
$149
limited time offer

Ethernet Starter Kit

C-Programmable 8-bit Rabbit 3000® Microprocessor core module with ethernet, experimentation board, and Dynamic C Lite with compiler, editor, and debugger.

Included Application Notes:
- X-10 Web Based Home Automation
- Ethernet Proximity Sensor
- Web Controlled Thermostat
- Network Lighting Control
- Many More...





$99
limited time offer

ALSO AVAILABLE

Microprocessor Starter Kit

C-Programmable, 8-bit Rabbit 3000® Microprocessor, application based, experimentation board, Dynamic C Lite featuring compiler editor and debugger.

Show off your work!

Want your application featured on our site?
When we publish your application we will send $150.
Visit **www.imaginetools.com**

imagine tools™

Think
Learn
Build

	Chipcon CC2420 (Ember EM2420)	Freescale MC13192	ZMD ZMD44101	Atmel AT86RF210
PHY frequencies	2.4 GHz	2.4 GHz	Sub 1 GHz	Sub 1 GHz
Transmit receive switch	Built in	Not built in: Use two antennas or an external TR switch	Built in	Built in
Package size	7 x 7 mm	5 x 5 mm	7 x 7 mm	7 x 7 mm
Output power	1 mW	1 mW	1 mW	4 mW
Receive sensitivity (with 1% packet error rate).* 802.15.4 requires only –85 dBm.	–94 dBm	–92 dBm	–100 dBm	–95 dBm
Adjacent channel rejection (5 MHz).* 802.15.4 requires only 0 dB.	39 dB	23 dB	30 dB	23 dBm
Alternate channel rejection (10 MHz).* 802.15.4 requires only 30 dB.	53 dB	35 dB	40 dB	40 dBm
AES encryption	Supported by hardware	Relies on the companion microcontroller to carry out this function	Relies on the companion microcontroller to carry out this function	Relies on the companion microcontroller to carry out this function
CRC	Supported by hardware	Supported by hardware	Supported by hardware	Companion microcontroller has hardware support for this function
Other MAC support functions (link quality, clear channel assessment, transmit and receive FIFOs, and CRC checking)	Supported	Supported	Supported by hardware and microcontroller	Supported
Available now?	Yes	Yes	Yes	Yes
*Rejection and sensitivity figures may not be quoted with the same test setup in all cases. Consult the respective datasheets for more information.				

Table 1—*On paper, the transceivers appear to offer similar functionality. The CC2420 costs $2 in large volumes, and is obtainable for approximately $10 per unit.*

flash memory with your application, are sold together for $6.75 in large quantities. Figure 8 Wireless supplies this ZigBee protocol software to Freescale, Chipcon, and Atmel. This will surely be one of the easiest ways to implement a ZigBee device at home. With the controller available in a 64-lead TQFP, or QFN, and the transceiver in a 48-pin QFN, it's still feasible to place these ICs on your own hand-soldered board.

Refer to Table 1 for a comparison of the four aforementioned transceivers. Some criteria (e.g., sensitivity) cannot be compared directly when considering transceivers supporting different PHY frequencies.

FAST ENOUGH?

ZigBee is optimized for low-power operation, so its data rate isn't as high as Bluetooth or 802.11. If throughput is your biggest concern when designing your new wireless network, you should first address the question of speed. If you have any doubts about 802.15.4 throughput for your application, you can download a free network simulation tool and a model of 802.15.4 supporting Beacon mode.

By altering the numerous parameters, you'll be able to see if the network performance you want will be possible in terms of latency, throughput, power, and the number of nodes. The network simulation software is called ns2 (from the University of California, Berkeley). The 802.15.4 networking model was developed by the University of Southern California's Autonomous Networks Research Group. (Refer to the Resources section of this article for more information.) With these tools you can quell your doubts before committing to buying or developing hardware and software.

WHERE TO NEXT?

After ZigBee is established, it will take over some applications where Bluetooth and other PANs or cable replacement technologies are currently in use. For the consumer, however, the biggest change will have to do with new applications that suddenly become economical after ZigBee reaches critical mass.

A wireless smoke detector serves as an example of a cost-reducing application. A smoke detector that hooks up to a home security system costs approximately $40. Add the wiring and installation charges, and your looking at a $90 price tag. Smoke alarms without built-in communications cost as little as $6. After manufacturers implement industry-wide ZigBee profiles for smoke alarms and security systems, you'll be able to add in a wireless smoke detector for roughly $10. Upgrading an existing security system base station with a ZigBee adaptor would be a cost-effective solution, even if you're adding only a single ZigBee smoke sensor or intruder detector.

The key hanger is an example of a new application. Right now, you probably use an RF key fob to remotely lock and unlock your car. Well, Freescale recently demonstrated a ZigBee solution that adds functionality to these devices. The ZigBee network will alert you when you misplace your keys. Say your cell phone and key fob are ZigBee nodes. The cell phone is the coordinator. The cell phone knows when it's home because it recognizes the other ZigBee nodes (e.g., a garage door lock, a universal remote, etc.). When these nodes are out of range, the cell phone assumes you're out and about. If your keys leave the cell phone's communication range, the phone will alert you before you get too far away.

BitScope
Pocket Analyzer

USB Oscilloscope & Logic Analyzer

8 Channel 40MS/s Logic Analyzer
Capture digital signals down to 25nS with arbitrary trigger patterns.

3 Input 100MHz Analog DSO
Classic Analog Scope using a standard x1/x10 BNC probe. Additional inputs on the POD for dual channel operation.

8 + 1 Mixed Signal Scope
True MSO to capture an analog waveform time-synchronized with an 8 channel logic pattern triggered from any source.

Real-Time Spectrum Analyzer
See the spectrum and waveform of analog signals simultaneously and in real-time

Waveform Generator
Load up to 32K arbitrary waveform and replay via the onboard DAC (10MS/s) or a digital pattern from the POD (40MS/s)

The new generation Scope for the age of microelectronics.



Turn your PC or NoteBook into a powerful Scope and Logic Analyzer!

See inside your circuit in the analog and digital domains at the same time to make tracking down those elusive real-time bugs much easier.

Pocket Analyzer combines a high speed sample-synchronized storage scope and logic analyzer with a programmable waveform and logic pattern generator. Also included is an integrated real-time spectrum analyzer and powered "Smart POD" expansion interface so you've got all bases covered!

About the same size and weight as a Pocket PC, this USB powered BitScope needs no bulky accessories. It's the perfect low cost "go anywhere" test and debug solution.





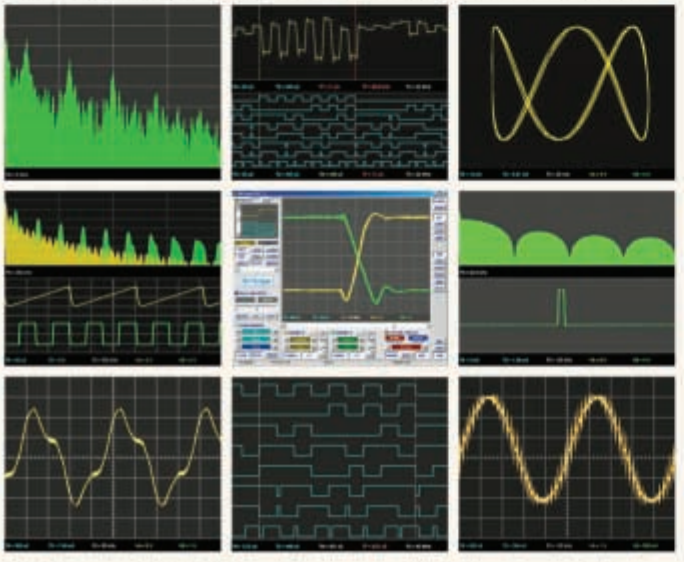
Standard 1M/20pF BNC Input
200uV-20V/div with x10 probe
S/W select AC/DC coupling
S/W select 50ohm termination
Arbitrary Waveform Generator

BitScope "Smart POD" Connector
8 logic channels, 2 analog channels
Dual channel capture from POD A/B
Async serial I/O for external control
Logic Pattern generator 32K 40MS/s

BUS Powered USB 2.0 Device
Single USB cable to your PC
Compressed data transmission
Simple ASCII control protocol
BitScope Scripting Language

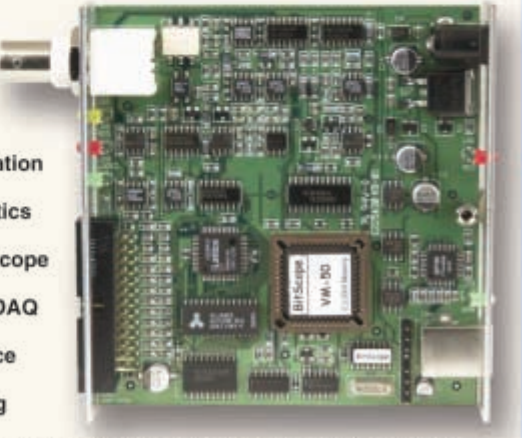
External/Passthru Power Supply
Auto senses an external supply - removes power load from USB for use with unpowered hubs. Supplies up to 500mA via POD

BitScope and your PC provide an array of Virtual Instruments



BitScope DSO 1.2 software for Windows and Linux

- R&D
- Education
- Robotics
- Lab Scope
- Fast DAQ
- Service
- Debug



BitScope Pocket Analyzer uses highly integrated Surface Mount technology to provide functionality you would expect from scopes many times the size and price. Its programmable Virtual Machine architecture means new functionality can be added via software. For custom Data Acquisition, export directly to your spreadsheet.

www.bitscope.com

The scheme could be applied to other devices too. No more lost sunglasses!

ZigBee is poised to become ubiquitous in home offices and factories, but I think the market may be overestimated. I'm a little skeptical about wireless technologies like ZigBee being relied on to control at least half of the functions inside commercial premises and certainly in most applications inside a wafer fabrication plant or auto factory. Sure, security is built in, but what if some malevolent person turns up with a high-power, wideband jamming device? DSSS is good at ignoring wideband noise up to a certain threshold, whereupon performance rapidly decays to nothing. Ditto for most other wireless systems.

Such an approach doesn't target the security layer. It simply makes transmission impossible because the clear channel assessment would always indicate a busy channel. Short-range versions used for prohibiting cell phone usage have been around for years. You can bet that these will be sold illegally on the 'Net to target the appropriate bands, if they aren't already.

Obviously, ZigBee nodes in a building will have sensible built-in behavior just in case communication fails. At worst, a sophisticated arsonist might want to defeat all smoke detectors in a certain area. At best, customers at a petrol station will be unable to pump petrol or pay for their purchases while some larrikin smirks knowingly from across the road. What if the business next door to the scene of the prank is operating a crane with a ZigBee-enabled emergency stop switch? Put simply, you need a secure physical medium such as copper or fiber (not air) connecting serious business assets. Wireless is for convenience rather than critical functions.

NOW'S THE TIME

Until now, it has been difficult to make a real ZigBee node on a tight budget. Although ZigBee-compatible hardware has been readily available for more than six months, the software stack above the 802.15.4 layers is only now obtainable for free. The modest Atmel chipset price includes the ZigBee stack under sub-license at no additional charge. Other vendors are sure to follow suit.

For hardware, ready-built PCBs in the form of evaluation kits are the easiest approach. Transceiver boards from Chipcon with the CC2420 are $50 each. Freescale's 13192DSK evaluation kit with a microcontroller implementing an 802.15.4 MAC and RS-232 interface costs $200 a pair.

There are a number of options for building your own hardware at a much lower cost. The components alone are less than $15 per node in low quantities. Refer to the Resources section if you're interested in obtaining the schematics, BOMs, and PCB layouts.

Several vendors currently offer full-featured ZigBee development tools for the higher layers, but they're expensive. Sounds like an excellent idea for an open-source community-based project! In the meantime, you can use the same physical layer that ZigBee uses along with the free MAC source code from Chipcon and Freescale. You can do a simple star topology network using the MAC layer and a few of your own functions to complete a rudimentary ZigBee-like network. If you take this approach, your hardware will be fully ZigBee-compatible. If you plan on applying more sophisticated software as it becomes easier to obtain, you can begin ZigBee development immediately. Do you act now, or do you let the ZigBee zephyr pass you by? ◙

Pete Cross lives in Hamilton, New Zealand, where he helps design online sensors that use a range of optical, electronic, and assay-based techniques for measuring biological components in fluids. You may reach him at pete.cross@clear.net.nz.

REFERENCES

[1] Chipcon, "CC2420: 2.4 GHz IEEE 802.15.4/ ZigBee-Ready RF Transceiver," rev. 1.2, 2004.

[2] ———"Quick Start Instructions: SmartRF C2420DK Development Kit," rev. 1.0, 2003.

[3] Ember Corp., "EM2420: 2.4 GHz IEEE 802.15.4/ZigBee RF Transceiver," 2004.

RESOURCES

J. Adams, "Meet the ZigBee Standard," *Sensors*, June 2003.

V. Bahl, "The ZigBee Alliance Overview," ZigBee Alliance, September 2002.

802.15.4 Networking Model for ns2 information, The USC Autonomous Networks Research Group, http://ceng.usc.edu/~anrg/downloads.html.

IEEE 802.15.4-2003, IEEE Standards Association, http://standards.ieee.org/getieee802/802.15.html.

IEEE 802.15 WPAN Task Group 4, IEEE, www.ieee802.org/15/pub/TG4.html.

The Network Simulator (ns2) information, The University of California, Berkeley, www.isi.edu/nsnam/ns.

Penton Media, "The ZigBee Buzz Is Growing," supplement to *Electronic Design*, January 2004.

Software example for blinking LED on CC2420DBK evaluation kit, Chipcon, release 1.2, 2004.

TinyOS information, The University of California, Berkeley, www.tinyos.net.

SOURCES

AT86RF210 Transceiver, ATmega128L MCU, and AT86ZL3201 controller
Atmel Corp.
www.atmel.com

CC2420 RF transceiver and 802.15.4 MAC sublayer in C for the AVRmega
Chipcon, Inc.
www.chipcon.com

MICAz CC2420-based mote transceiver PCB
Crossbow Technology, Inc.
www.xbow.com

EM2420 Transceiver
Ember Corp.
www.ember.com

MC13192 RF Transceiver and S-MAC and 802.15.4 for the MC9S08GT60
Freescale Semiconductor
www.freescale.com

Telos CC2420-based mote transceiver PCB
Moteiv Corp.
www.moteiv.com

ZMD44101 RF transceiver IC
ZMD America
www.zmda.com

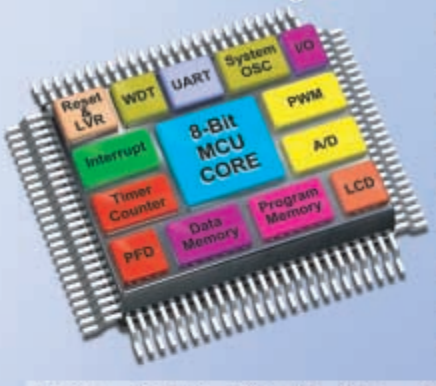
HOLTEK
www.holtek.com

A/D with LCD MCU

HT46R6X



Holtek A/D with LCD MCUs for Intelligent Home Appliance and Automotive Peripheral Control Applications



- Cost-effective solutions
- Industrial Specifications:-40˚C~+85˚C
- High noise immunity and electrostatic protection
- Wide Operating Voltage 2.2V~5.5V
- Up to 4-channel 8-bit PWM outputs
- Up to 184 Pixel LCD display capacity
- Up to 8-channel 12-bit A/D converter
- UART serial port (HT46R66 only)
- OTP and Mask device compatibility

Part No.	Program	Data	I/O	LCD	Timer	RTC	Interrupt	A/D	PWM	PFD	Package
HT46R62	2Kx14	88x8	20	19x4	8bitx1	V	6	9bitx6	8bitx3	V	56SSOP
HT46R63	4Kx15	208x8	32	19x4	16bitx1	V	6	8bitx8	8bitx4	--	56SSOP,100QFP
HT46R64	4Kx15	192x8	24	32x4	8/16bitx1	V	7	10bitx8	8bitx4	V	56SSOP,100QFP
HT46R65	8Kx16	384x8	24	40x4	16bitx2	V	7	10bitx8	8bitx4	V	56SSOP,100QFP
HT46R66*	16Kx16	576x8	32	46x4	8bitx1,16bitx2	V	8	12bitx8	8bitx4	V	56SSOP,100QFP

*Under development, available in 1Q, 2005.

Holtek
starting as low as $125 USD
In Circuit Emulator

New range of development tools and volume production type OTP writers now available !



Order your Holtek In-Circuit-Emulator today and get 10% discount
(valid for USA and Canada customer only; offer expires on 12/31/2005; discount code: FHSIUCC)

HOLTEK SEMICONDUCTOR INC.

USA Sales Office : Holmate Semiconductor, Inc.
46712 Fremont Blvd., Fremont, CA 94538 Tel: (510) 252-9880 Fax: (510) 252-9885
www.holmate.com

AVR® 2004 DESIGN CONTEST

Contest Winner

Stealth Keyless Entry System

That 10-button keyless entry pad might be more hazardous than helpful. An intruder can discover your code by studying the wear on the buttons. David tells us how he built a better system.

Many houses feature a 10-button keyless entry pad that controls a garage door. Anyone who knows the numerical entry code can operate the system. The buttons are usually labeled 0 through 9 along with the three-letter groupings included on telephone keypads.

Keyless entry systems are secure at first. After all, it would be practically impossible for an intruder to guess the correct four-digit code on a 10-button keypad. However, these systems tend to become less secure over time. The more you use the pad, the easier it becomes for someone to identify the numbers in your code from the worn and dirty buttons. By recognizing the worn buttons, an intruder greatly reduces the number of permutations needed to guess your entry code.

My stealth keyless entry system allows you to overcome the disadvantages of a worn 10-digit keypad. The idea is simple: use a doorbell button for user input to activate a door opener. An AT90S2313 microcontroller monitors the doorbell button (switch) openings and closures. It also allows you to activate a doorbell and garage door opener. Pressing the switch triggers the doorbell (no surprises here). However, if you key in a special code via the switch, the doorbell won't ring and the garage door will open.

This system brings a new level of sophistication to home security. Few intruders would guess that the switch controls your garage door. If an intruder somehow figures this out, he'll end up ringing your doorbell if he attempts to guess your code.

CODE SEQUENCE

One of the challenges this project presents involves coming up with a reasonable sequence of button presses that your family members will be able to remember. Morse code is the obvious choice for an entry code sequence.

When I was a kid, I was into Ham radio. All those early years of clacking away on a telegraph key embedded Morse code in my brain. If you aren't a Ham enthusiast, will you have to learn Morse code? Ugh! Well, don't be discouraged. You don't need to memorize the entire Morse code to use this system. A few letters are all you need. Moreover, the AT90S2313 software is tolerant in terms of what it accepts as dots and dashes.

SYSTEM BASICS

The AT90S2313 has 2 KB of in-system programmable flash memory, 128 bytes of SRAM, and 128 bytes of EEPROM. The system stores a master key string in EEPROM that's read into the SRAM when it boots. The key can consist of between one and 10 Morse characters. Each character is one of the 26 letters of the alphabet.

To operate the system, key in a sequence of Morse characters via the switch input. A keyed in character consists of a sequence of one or more dots and dashes. Each switch-closed interval, which I'll call an element, defines a potential dot or dash for the MCU. Fixed time thresholds decode received elements into codes that potentially represent Morse characters. These fixed thresholds are tunable parameters in the software and are set for relatively slow Morse code transmission. Currently, the switch closure time for a dot is between 30 and 180 ms; it's between 180 and 1,200 ms for a dash. Switch closure times less than 30 ms or greater than 1,200 ms trigger an immediate key match failure on input.

Fixed time thresholds are also provided for inter-element and inter-character times. As a result, the MCU can assemble elements into Morse characters and distinguish Morse character boundaries. Moreover, there are fixed thresholds that terminate the switch input if the switch is held closed too long or if a partial key sequence is entered and too much dead time (no switch activity) occurs.

The MCU debounces the switch closures and openings. It times the input sequence in order to classify the switch closure times as valid Morse elements, to identify the Morse characters represented by a sequence of elements, and to compare the Morse characters to the key in SRAM (see Figure 1).

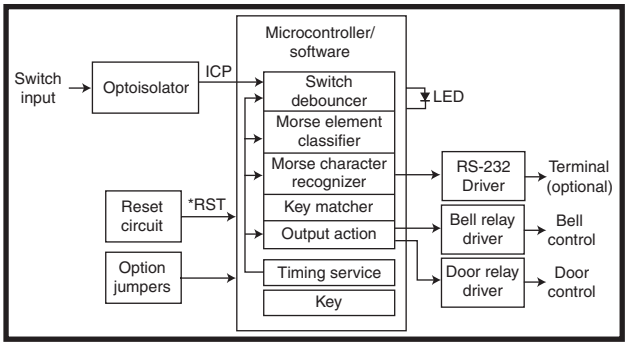


Figure 1—*The MCU software monitors doorbell button activity via the switch input. Morse code characters (letters A through Z) are deciphered by debouncing switch operations, classifying switch closures as valid Morse code elements (dots and dashes), and recognizing element ensembles as valid Morse characters. If a sequence of received Morse code characters matches a master key string, the door relay driver operates. Otherwise, the bell relay driver operates.*

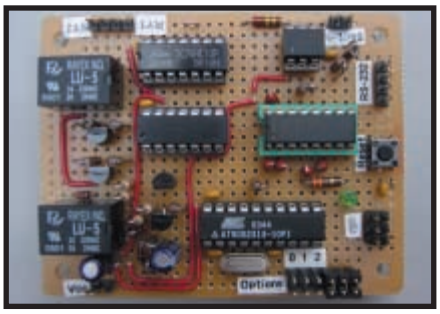


Photo 1—*Five-volt power is routed on top of the circuit board and copper foil is applied to the bottom of the board for a ground plane. Various headers are used for all of the circuit board connections.*

As soon as the system detects an invalid Morse character or a key mismatch, the bell relay contact pair closes for a short time. If the input sequence matches the key in SRAM, the door opener relay contact pair closes for a short time. After the bell or door relay activates, a 4-s dead time period begins during which switch action is ignored. Further switch processing is disallowed until the switch is open for the entire dead time period. This combats an intruder's repeated attempts to guess the entry code.

The default key string "OPENME" is downloaded to the EEPROM when the MCU is programmed. This is easily changed by one of the several options provided by the hardware.

To enter a new key, use the strap for the "Enter new Key" option on the hardware Options header. Then reset the MCU. Next, key in the new key in Morse code via the switch input. After a new key entry starts, the entry is terminated in one of the following situations: when an element isn't a valid Morse code dot or dash; when an invalid Morse code character is entered; when 10 valid Morse code characters are entered (the maximum key length allowed); or when a switch dead time period expires.

If the hardware's 2,400-bps output is connected to a terminal (like HyperTerminal running on a PC), new key characters will be printed as they're entered. When the key entry is terminated, text is sent on the 2,400-bps line to instruct you to change the board's Options strapping and reset the MCU. The system then boots up using the newly entered key.

I wrote the software for this project in assembly code. It's contained in a single source file, keyless.asm, which you may download from the *Circuit Cellar* ftp site. I used Atmel's AVR Studio 4 design software for development. I used an AVR in-system programmer to download the object code to the MCU flash memory and EEPROM.

HARDWARE

The circuit was constructed on a 2.5" × 3.25" prototyping board with copper pads per hole on one side of the board. I applied adhesive copper foil to the bottom of the board for the ground plane and then cut away portions of it to allow for component placement. Be sure to provide a reasonable ground system and power bypassing close to the ICs. I routed 5-V power on top of the board. Signal connections were made point-to-point on the bottom. Connections to and from the board were made via the headers shown in Photo 1 and listed in Table 1.

Board header	Function
V_{CC}	5 V and ground power connection
Options	System options selection (via pin pairs 0, 1, and 2)
ISP	Connection for the AVR ISP in-system programmer
RS-232	2,400-bps output connection
Switch	Doorbell switch connection
RY1	Relay contact pair for activating a door opener
RY2	Relay contact pair for activating a doorbell

Table 1—*The board headers apply to the circuit in Photo 1.*

The Options header contains 2 × 6 pins. Jumpers are placed in the pin pairs labeled 0, 1, and 2 to activate a particular option. The other three pin pairs aren't connected in the circuit; they're there to store the jumpers used on the pin pairs. The jumper placements for the Options header are listed in Table 2.

Pin Pair			Option Selected
2	1	0	
0	0	0	Normal system operation
0	0	1	Print switch closure times
0	1	0	Print Morse code characters entered via switch
1	0	0	Enter new key via switch
1	0	1	Print key match successes and failures
1	1	0	Print key on system boot
0	1	1	Unused
1	1	1	Unused

Table 2—*The circuit's Options header provides the jumper placement for the various options. A one indicates the placement of a jumper. A zero indicates the absence of a jumper.*

Figure 2 (page 54) is a schematic of the system. The doorbell button connects to the header labeled Switch, where optoisolator IC2 isolates the external doorbell button's wiring from the main circuit (see Figure 2a). Note that C1, which is connected to the optoisolator diode current-setting resistors R1 and R2, has nothing to do with switch bounce. The time constant of this RC input is approximately 9 µs. It provides some RF immunity. The optoisolator drives an inverting Schmitt trigger for a connection to the MCU's input capture pin.

Let's focus for a minute on the MCU's reset circuitry. IC6 is a brownout detector that resets the MCU when the 5-V rail begins to dip. Capacitor C2 charges slowly through R5 and is quickly discharged through R6 when you press the board's Reset switch. An inverting Schmitt trigger that's connected through transistor Q1 will hold the MCU in Reset mode until the voltage on C2 reaches the threshold of the inverter. This circuitry is intended to delay the removal of Reset mode until well after switch bounce in the reset switch has ended.

For the component values chosen for R5 and C2, the MCU reset is asserted for approximately 100 ms after the reset switch has been released. This guarantees a well-defined reset pulse for the MCU no matter how shabby the reset switch is or how quickly its closure might quit. Diode D2 provides ESD protection.[1] LED1 is used for software development and user-training purposes.

Each of the two relay drivers use three MCU output port pins (see Figure 2b). A driving bit pattern of 101 is required to turn on a relay. This method is more robust than using a single MCU port pin as a relay driver. The intent here is to reduce the probability of errant software operating a relay. Pull-

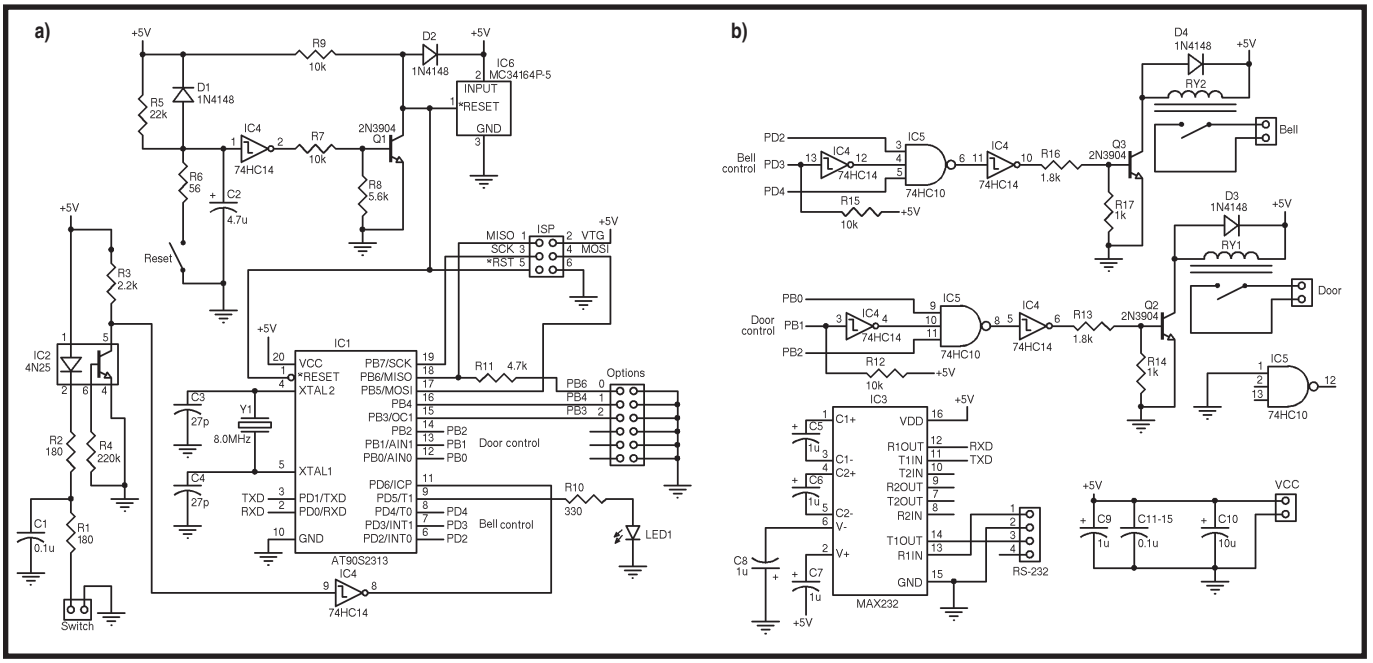


Figure 2a—*An optoisolator isolates the doorbell button's external wiring connected to the switch input from the MCU circuitry. An MC34164 brownout detector along with a collection of parts drive the MCU reset input.* **b—***Each of the two relay drivers are controlled by three MCU output pins. A 101 bit pattern is required to activate a relay. This process is less error-prone for false relay operation compared to a single MCU output pin driving a relay.*

up resistors R12 and R15 are provided to ensure that a relay doesn't operate when the MCU is in Reset mode.

SOFTWARE ARCHITECTURE

I defined a finite state machine (FSM) containing five system states. The system rests in one of the states when it's waiting for an input stimulus.

Software operation is event-driven. Events are sent from ISRs as a result of input switch action or the firing of a guard timer (timing out). An event consists of setting an event flag (bit) in a global Events byte from within an ISR. The system's idle loop spins on the Events byte waiting for an event flag to be set:

```
EvLoop:
  tst Events
  breq EvLoop
```

When the system is initialized following an MCU reset, a series of functions is called to initialize system parameters and peripherals. Next, the idle loop is entered. After an ISR sets an event flag, processing breaks out of the idle loop and continues in one of five major functions (one for each of the five system states). Each major function has code to handle the events that are expected in that state.

Processing in these functions and the other functions invoked by them implements the software components illustrated in Figure 1. Table 3 identifies the seven event flags assigned to bit positions in the Events byte.

SYSTEM TIMING

The AT90S2313 contains two timer/counters: the 8-bit Timer/Counter0 and the 16-bit Timer/Counter1. All system timing, except switch debouncing, is derived from Timer/Counter0. Timer/Counter1 alternates between the Input Capture and Output Compare Match modes to detect input switch openings and closings and to perform switch debouncing. The MCU's timer/counter prescaler is driven from an 8-MHz clock and drives each timer/counter at one of the available prescaled selections.

Timer/Counter0 is programmed so the overflow interrupt ISR associated with the timer/counter is called at a 500-Hz rate. This is accomplished by selecting the right combination of timer/counter prescaler and initial counter values for an interrupt every 2 ms.

A 16-bit counter is incremented on each entry to the ISR to provide a system clock with 2-ms resolution. All time measurements performed in deciphering the switch input keying are based on this 2-ms system clock. Five separate guard timers are also administered by the Timer/Counter0 ISR. A guard timer restarts processing after a specific time period has elapsed. For example, a Morse code character consists of a sequence of one to four elements. While receiving elements from the input switch, the MCU must know when to stop waiting for another element so that a received sequence of elements can be interpreted. To accomplish this, the Morse code character inter-element guard timer is set at the end of each element reception. If the timer fires (times out) before another

Event flag	Meaning
Sw_Open	Switch opening detected
Sw_Close	Switch closure detected
IETimeout	Morse code character inter-element guard timer timeout
ICTimeout	Morse code character inter-character guard timer timeout
DPTimeout	Dead time period guard timer timeout
EXTimeout	Excessive switch closure time guard timer timeout
NKTimeout	Excessive time entering new key guard timer timeout

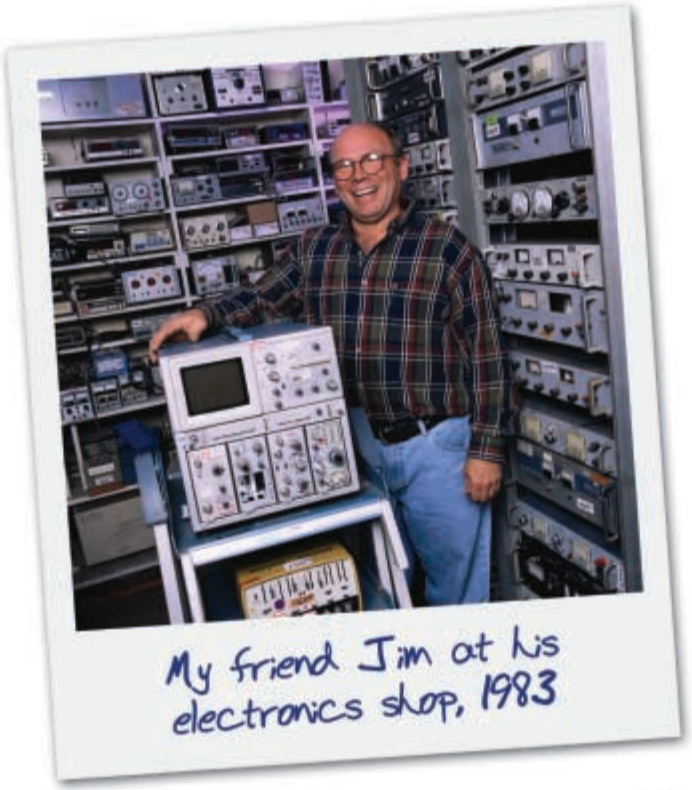
Table 3—*Seven events are sent from two ISRs. The first two address switch action. The others identify guard timer timeout. Processing breaks out of the idle loop to perform work when an event flag is set.*

Remember when...

...electronics stores were stocked to the rafters

with every component imaginable and there was ALWAYS

a knowledgeable person there to help you?





@ HOME
JAMECO®
1-800-455-6119

Think those days are gone?
Well they're not!
You just need a little Jameco@Home.
Call for a FREE catalog today.

We have the electronic components and knowledgeable Technical Support people who can help **bring back the fun in your electronics hobbies, inventions, and DIY projects.**

Be sure to visit our new website at
www.Jameco.com/cci

Jameco@Home
—where electronics is still fun!

element starts, the MCU is informed to process the received element collection.

Guard timer implementation requires two pieces of memory: a timer-running flag and a 16-bit timer timeout value. The former has two values. The timer is running or it isn't. A guard timer is set by adding the desired guard time to the current 2-ms system clock value, storing this in the timer timeout value, and then setting the timer running flag to "running." The Timer/Counter0 ISR checks the timer-running flag once every 2 ms after the system clock has been incremented. If the flag indicates the timer is running, the timer timeout time is compared with the current 2-ms system clock. When the two match, it means a guard timer timeout has occurred.

After a timer timeout, the event flag for the guard timer is set and the timer running flag is changed to timer "not running." The event flag is then noticed in the idle loop where the appropriate function will be called to deal with the event. A guard timer can be cancelled by simply setting the timer-running flag to the timer "not running" value.

Timer/Counter1 has the capability of generating an interrupt in Input Capture mode, Output Compare Match mode, and Counter Overflow mode. Two of these modes are used. Input Capture mode detects switch openings and closures. Output Compare Match mode is used in switch contact debouncing. The use of Timer/Counter1 is best explained by describing switch debouncing.

SWITCH DEBOUNCING

I investigated switch debouncing for an earlier project in which a number of inexpensive switches were observed during switch openings and closures. An MCU was programmed to sample the state of the switch, beginning with the initial switch opening or closure and continuing every 200 µs for approximately 15 ms. The generated data clearly illustrated how long bouncing activity went on relative to the initial switch opening or closure. Bounce times ranged from almost nothing to as much as 10 ms. This experience led to setting the switch debounce time software tune parameter in this project to 10 ms.

The debounce algorithm in this project recognizes and timestamps the initial opening or closure of the switch. In addition, it prevents the debounce transitions from driving the MCU insane, and it verifies that a real switch transition has taken place after all the bouncing activity has stopped.

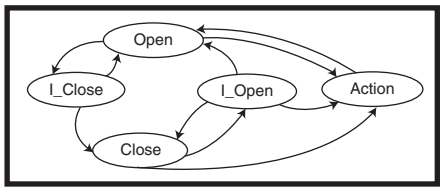


Figure 3—*This state diagram shows the five major states of the software and allowable state transitions. Four of the states deal with switch debouncing and Morse code element and character timing. The fifth state, the action state, deals with relay operation.*

Figure 3 shows the five states of the software FSM and the allowable state transitions. When the switch is open and the system is in the open state, a switch closure detection causes a transition to the `I_Close` state (initial close). After entering the `I_Close` state, the time is recorded from the 2-ms system clock and a switch debounce timer (10 ms) starts.

Switch openings and closures during the debounce time are ignored. When the debounce timer times out, the state of the switch is sampled. If the switch is open, then the system moves back into the open state and nothing else occurs. However, if the switch is closed, then the system moves into the close state, where the time recorded during entry to `I_Close` is taken to be the switch closure time. Here, an event flag is set to signal that a switch closure has occurred. The same process is used in moving from the close to open state via the `I_Open` (initial open) state when a switch opening occurs.

Timer/Counter1 is used to detect switch openings and closures and to implement the switch debounce timer. Input Capture mode is used for switch opening and closure detection. Output Compare Match mode is used for debounce timer implementation. Only one of these modes is needed at a time to implement the switch debounce algorithm. The timer/counter is conveniently switched between the two modes. Because two timer/counter modes are being used, there are two ISRs associated with the timer/counter. One is for the Input Capture function. The other is for Output Compare Match mode.

CHARACTER RECOGNITION

As previously stated, each switch closure interval (the length of time the switch is closed) is defined to be an element. Morse characters consist of a sequence of from one to four elements. (The Morse code character set appears at the end of the keyless.asm file posted on the *Circuit Cellar* ftp site.)

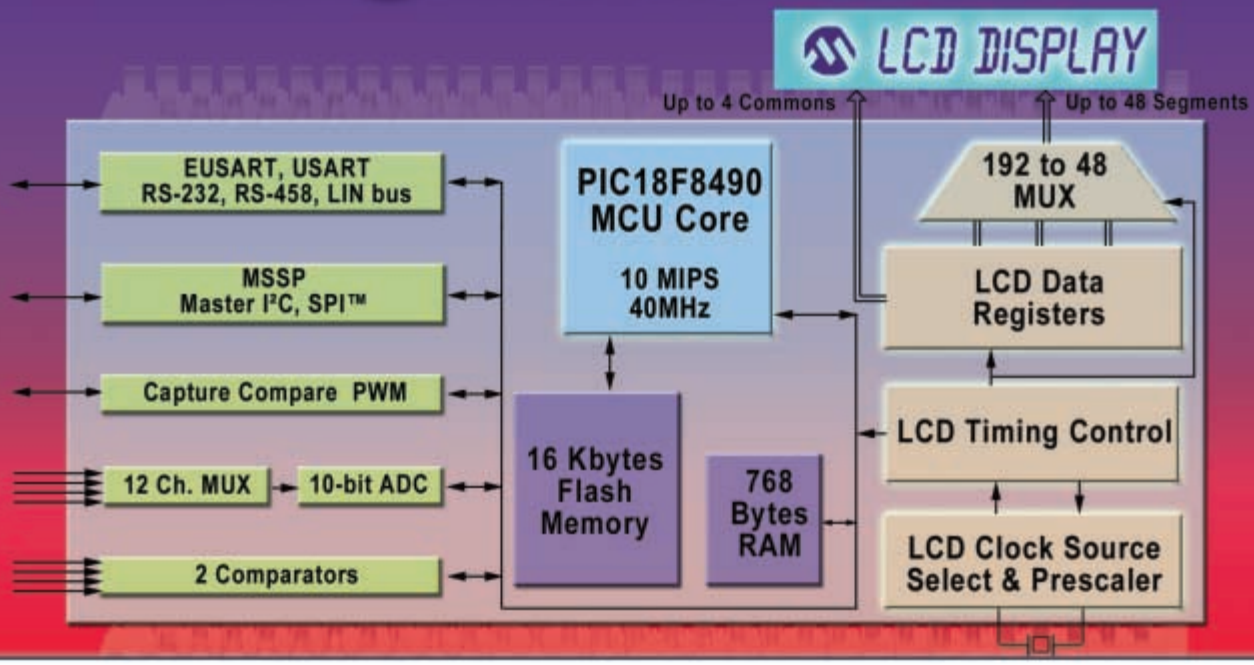
There are tunable thresholds for classifying an element as a dot, a dash, or an invalid Morse code element. There is also a timing constraint on the time interval between elements in a Morse character (the inter-element guard timer) where the current threshold on the maximum time between elements is 500 ms. If more than 500 ms of switch open time occurs following the ending of an element, the previously received element ensemble is interpreted as a potential Morse code character. As you gain experience using this system, the 500 ms may become annoying. If you want to speed up the process of Morse code character entry, this likely would be the first software tune parameter to shorten.

The switch debounce algorithm detects each element and records the element length. Received elements are validated and classified as dots or dashes. As elements are received, a 2-bit code representing the dot or dash is assigned to a 2-bit position within a dot/dash code byte. The first element received is assigned to bits 0 and 1. The second element is assigned to bits 2 and 3, and so on. If a received element is too short or long to be classified as a dot or dash, or if too many elements are received, an invalid Morse code character is declared. When the inter-element guard timer fires and terminates the reception of a Morse code character, the received Morse code letter represented by the dot/dash code byte is checked against what's expected in the master key string.

KEY MATCHING

Character storage in the master key string uses the same coding scheme used for constructing the dot/dash code of received Morse code characters. Key matching progresses by checking the first Morse code character received against the first position in the key. The second Morse code character is checked against the second position in the key,

Flash PIC® microcontrollers with integrated LCD module



The new LCD PIC® Microcontrollers from Microchip are Flash, power-managed microcontrollers with on-chip LCD driver control modules. This family provides up to 192 LCD segment drivers, a variety of package sizes and integrated features for embedded control applications. These microcontrollers featuring nanoWatt Technology meet low power design requirements including driving the LCD display in Sleep mode. For more information and to learn about associated development tools, please visit the LCD Design Center at www.microchip.com/lcd.



PICDEM™ LCD
Demo Board

Device	Flash Program Memory (bytes)	Data Memory		I/O	10-bit ADC	Capture/ Compare (PWM)	MSSP			EUSART/ USART	LCD Segments	Comparators	Pins
		EEPROM (bytes)	Data RAM (bytes)				SPI™	MI²C	I²C™				
PIC18F8490	16384	—	768	66	12	2	Y	Y	N	1/1	4x48 (192)	2	80
PIC18F8390	8192	—	768	66	12	2	Y	Y	N	1/1	4x48 (192)	2	80
PIC18F6490	16384	—	768	50	12	2	Y	Y	N	1/1	4x32 (128)	2	64
PIC18F6390	8192	—	768	50	12	2	Y	Y	N	1/1	4x32 (128)	2	64
PIC16F917	14336	256	352	36	8	2	Y	N	Y	1/1	4x24 (96)	2	40/44
PIC16F916	14336	256	352	25	5	1	Y	N	Y	1/1	4x15 (60)	2	28
PIC16F914	7168	256	256	36	8	2	Y	N	Y	1/1	4x24 (96)	2	40/44
PIC16F913	7168	256	256	25	5	1	Y	N	Y	1/1	4x15 (60)	2	28

buy MICROCHIP ONLINE CATALOG
One-stop shopping for MCUs, digital signal controllers, analog and serial EEPROMs.

Now Pb-free!

MICROCHIP
www.microchip.com/LCD

The Microchip name and logo, and PIC are registered trademarks of Microchip Technology Incorporated in the USA and in other countries. All other trademarks mentioned herein are property of their respective companies. ©2005 Microchip Technology Incorporated. All rights reserved.

and so on. If a key match is found, a check is made to see if the end of the key string has been reached where a null character in the next key position identifies the key end. Consequently, as each Morse character is received, key matching will indicate one of the following: no match, a partial match (character matched that in the key, but the key end hasn't been reached), or a total match (character is matched and the key end has been reached).

If key matching indicates a partial match, the system will wait for additional Morse code characters to be received. In this situation, the guard timer for Morse code inter-character time is set. The value for this timer is currently set high at 4 s. This gives you adequate time to begin entering the next Morse code character. Firing of this guard timer will terminate waiting for more input from the switch. This in turn is interpreted as a key match failure. If key matching doesn't indicate a partial or total match, then the system won't wait for additional input. Obviously, no match is a key match failure.

RELAY ACTIVATION

After it's determined whether or not there was a match, the system enters Action mode. Relay RY1 opens the door if a total key match occurs. Otherwise, relay RY2 rings the doorbell. In either case, the relay is held in the operated state for 1.5 s.

Following relay operation, the dead time period guard timer is set. Four seconds is the current value for this timer. This time must expire before the system is allowed to advance to the open state and arm itself for new switch input. As I mentioned earlier, the dead time is a safety measure against repeated attempts to guess the key. A switch closure during the dead time will reset the dead time period guard timer.

INSTALL A NEW KEY

During normal operation, a received Morse code character is checked against the key. When the Enter New Key option is made on the Options header, a received Morse code character is validated to make sure it's a letter. When the received character is valid, the software does five things. First, it sets a guard timer to terminate reception of new key characters if another character isn't received in a reasonable amount of time. Second, the software prints the received character on the 2,400-bps output. Third, it writes the received character's dot/dash code to the master key in EEPROM. Fourth, it advances an index into the master key to the next key byte. And fifth, the software checks to see if the maximum key length has been reached.

If the received character isn't valid, or if the maximum allowable length for the key has been reached, three steps terminate new key character reception. First, a null character is added to the master key string in EEPROM as long as at least one new character's code is written to the key. Second, instructions are printed on the 2,400-bps output instructing you what to do next. And third, an endless loop is entered where LED1 continuously flashes. Next, you must change the Options strapping and reset the MCU. If the MCU is reset without changing the Options strapping, then the system will boot up and expects a new key to be entered again.

USER TRAINING

Can anyone just walk up to my keyless entry system and use it? It's unlikely. Operating the system takes practice, even if you are skilled in the art of Morse code. For this reason I built a few training modes into the system. As with the "Enter new key" option, the training modes are selected by the Options header (see Table 2). The two most important ones are "Print Morse code characters entered via switch" and "Print switch closure times."

The "Print Morse code characters entered via switch" option prints the Morse code characters entered via the switch over the 2,400-bps output. When an optional terminal is connected to this output, you can easily get the feel for the system by seeing the Morse code characters keyed into the system printed on the terminal screen. Note that key matching and relay operation aren't performed with this option. If an illegal character is entered, a question mark will be printed. I recommend taking your time when entering Morse code via the switch to cut down on potential errors.

Finally, the "Print switch closure times" option prints each switch closure time that is longer than the debounce interval via the 2,400-bps output. With this option, only the switch closure times are printed. Elements aren't assembled into Morse code characters for matching against the key. This is provided for setting the tune parameters in the software that are used for identifying valid Morse code dots and dashes. There is one inconvenience with this option: switch closure times are printed in hexadecimal. MCU memory limitations prevented the addition of a hexadecimal-to-decimal conversion function.

If you try this project, I encourage you to experiment by changing the software tune parameters defined at the beginning of source code file keyless.asm. These time threshold parameters are used for decoding elements received from the input switch into Morse code characters. By experimenting with these parameters, you can further adjust the system's response to your liking.

David Brown is a retired telecom engineer who studied electrical engineering at Penn State and Purdue. His interests include embedded microcontroller projects, amateur radio, and cycling. You may contact him at k8ax@wowway.com.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/176.

REFERENCE

[1] Atmel, "AVR Hardware Design Considerations," 2521B, 2004.

RESOURCES

Atmel, "AVR042: AVR Assembler User Guide," 1022A-A, 1998.

———"8-bit AVR Instruction Set," 0856D-AVR, 2002.

———"AVR ISP User Guide," 2004, www.atmel.com/dyn/products/other_docs.asp?family_id=607.

SOURCE

AT90S2313, AVR ISP, and AVR Studio 4
Atmel Corp.
www.atmel.com



TINY DYNAMO

ZiLOG

Reach New Heights with

Z8 Encore! XP®



Meet the Z8 Encore XP® 8-pin

- 8-pin package in PDIP, SOIC, or MLFs
- Internal Precision Oscillator
- Built-in Temperature Sensor
- 1K to 4K Flash Memory
- Standard and Extended Temperature Ranges
- Single Pin On-chip Debugger

Begin innovating today using our comprehensive suite of Development Tools, including a full C-Compiler.



Kits Available for only

$49.95

Z8 Encore! XP® 4K Series Block Diagram

1–4KB Flash	256B–1KB RAM	16B–128B NVDS	Up to 8 Channels 10-Bit ADC
Two 16-Bit Timers/PWM	20MHz eZ8 CPU		Trans-Impedance Amplifier
Watch-Dog Timer with RC Oscillator			POR/VBO & Reset Control
UART with IrDA	On-Chip Debugger		Crystal/RC Oscillator
Temperature Sensor	Analog Comparator		Internal Precision Oscillator
Up to 25 General-Purpose I/O Pins			

Z8 Encore! XP® 4K Series Development Kit



Real innovation means no limits. Think small.
Visit **www.zilog.com/Z8XP8** today and register to win FREE stuff! Go online and use promo code **DYNAMO** to find out more.

Control an FPGA with I²C

Is the I²C bus a good candidate to control an FPGA? Roberto thinks so. In this article, he makes a case for I²C based on a careful study of its capabilities. He also shows us how to drop an I²C module into an FPGA.

Although modern microcontrollers and FPGAs offer large numbers of I/O pins, and SMD technology makes it possible to accommodate many more pins on a square inch of a PCB, this hasn't discouraged the use of serial buses for component connection on a board. In addition to traditional buses like I²C and SPI, new ones have been invented for particular applications such as the Dallas 1-Wire bus, which is growing in use for smart sensors as thermometers, intelligent keys, and so on.

In Fast mode, SPI links microcontrollers with fast ADCs, DACs, and EEPROM devices. High data rates can be sustained because it uses separate input and output pins. There isn't an embedded address in the protocol, so a chip select signal must be supplied for each device connected to the bus. Each slave device has a Slave Select input. The master device must have as many Slave Select outputs as there are slave devices, or it must have an external decoder. The SPI bus allows for the easy connection of non-intelligent peripherals like a shift register. Chained shift registers can implement large input and output expanders.

On the other hand, a good level of intelligence is required for a connection to a Dallas 1-Wire bus. In this protocol a single line with a pull-up resistor is taken low for 1 to 15 µs in order to transmit a one. A zero is transmitted by pulling the line low for more than 60 µs and less than 120 µs. In this way, commands and data are sent to the slave device. Data is transferred from the device to the controller by having the controller transmit a stream of ones for synchronization and having the slave turn some of them into zeros. In a similar way, reset and presence pulses are exchanged to synchronize the stream between a master and slave device.

Philips Electronics developed the I²C standard in the early 1980s. The I²C bus is a good compromise with regard to speed and complexity. Two maximum transfer speeds are defined: 100 and 400 kHz. But there is no restriction in the length of a single bit, so even software-driven I²C devices can be easily implemented on a microcontroller without, for example, a restriction on the use of interrupts that may cause the bit period to be extended arbitrarily.

Only two wires are required to implement an I²C bus, and because of the open-collector nature of the electrical interface, a slave can be connected or disconnected at any time without affecting any other device on the bus. I²C includes an arbitration procedure that takes advantage of the wired AND connection. This allows a true multimaster bus to be implemented. In practice, if two or more masters try to put information on the bus (the first to produce a one while the others produce a zero), they will lose the mastership. Also, if a slave is a slow device, it can slow down any transaction on the bus by holding down the clock line.

Each device on the bus is selected by a unique 7-bit address. A simple master/slave relationship exists at all times. The only other limit on the number of slaves connected to the same bus is the maximum bus capacitance of 400 pF.

For all these reasons, the I²C bus is a good candidate for controlling an FPGA. You can implement the slave interface with the custom logic in the FPGA.

#USE I2C
Syntax: #USE I2C (options)

Elements: Options are separated by commas and may be:

MASTER	Set the Master mode
SLAVE	Set the Save mode
SCL=pin	Specifies the SCL pin (pin is a bit address)
SDA=pin	Specifies the SDA pin
ADDRESS=nn	Specifies the Save mode address
FAST	Use the fast I²C specification
SLOW	Use the slow I²C specification
RESTART_WDT	Restart the WDT while waiting in I2C_READ
NORFORCE_SW	Use hardware I²C functions

Purpose: The I²C library contains functions to implement an I²C bus. The #USE I2C remains in effect for the I2C_START, I2C_STOP, I2C_READ, I2C_WRITE, and I2C_POLL functions until another USE I2C is encountered. Software functions are generated unless the NORFORCE_SW is specificied. Slave mode should be used only with the built-in SSP.

Figure 1—*The CCS C compiler for the Microchip PIC includes support for I²C implemented in hardware or software.*

INSIDE I²C

The I²C bus is a well-known interface, so there are numerous microprocessor peripherals that can be attached to the bus, starting with I/O expanders (e.g., the PCF8574) to display controllers and EEPROMs. The amount of storage associated with an I²C device is unlimited; it can range from a few bytes to 256 Kb or more.

At the same time, many microcontrollers (e.g., the Cypress CY7C68013 and Microchip PIC18C*xxx* family) now include an I²C bus master peripheral on-chip. If a hardware interface isn't provided, it can be achieved by controlling a pair of I/O pins in software.

Many compilers offer driver routines to control I²C devices

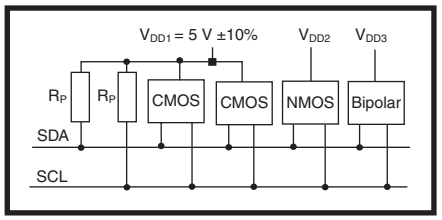


Figure 2—*Devices connect to the I²C bus using just two signals, SCL and SDA. Any supply voltage may be used as long as the device can tolerate having the bus lines pulled up to 5 V.*

with or without a hardware interface. Figure 1 shows how the declaration of an I²C peripheral on a PIC microcontroller is done using the CCS C compiler.

Two speeds are available. Slow is the original speed incorporated into early devices; it's limited to 100 kHz. Some EEPROMs and early I²C peripherals use this speed. Fast is typically 400 kHz, which is supported by a wide variety of newer devices. The other options are used to specify whether the I²C interface is a master or slave interface. They also specify whether the hardware interface or software emulation should be used, and whether or not the pins should be used for the clock and data.

I²C BUS CONNECTION

The two wires in the I²C bus are usually referred to as SDA and SCL. They're used to transmit data and clock, respectively. Devices connected to the two wires must have open-drain outputs and mustn't clamp the input line to their own supply voltage with ESD protection diodes. This allows the bus to be shared among devices using different supply voltages. The pull-up resistors on SDA and SCL must be connected to 5 V ±10% (see Figure 2). Each device should include on-chip filtering to reject spikes and improve the overall robustness of the bus.

The I²C bus allows for multimaster configurations. A master that wants to relinquish the bus must simply stop driving the SCL line. If a system has only one master and the slaves are fast devices that don't require clock stretching, the pull-up resistor on the SCL line can be deleted and the master can actively drive the SCL line both low and high.

The two-wire protocol is based on the identifications of start and stop conditions. A start condition is defined as a high-to-low transition on the SDA line while the SCL line is high. A stop condition is defined as a low-to-high transition on the SDA line while the SCL line is held high.

At all other times, it is mandatory that SDA remain stable whenever the SCL line is high during the data phase of the transaction. The SDA set-up time (to SCL rising) is specified based on the nominal bus speed: 250 ns on a 100-kHz bus and 100 ns on a 400-kHz bus.

The number of data bytes between a start and stop condition is unlimited; the only requirement is that each data byte must be acknowledged. The acknowledge condition is a low on the SDA line in the ninth bit position following each 8-bit byte (see Figure 3). In a write cycle, the slave provides the acknowledge on the data bytes. In a read cycle, the master provides the acknowledge on all but the last data byte. The missing acknowledge is how the master signals to the slave that it intends to terminate the transaction.

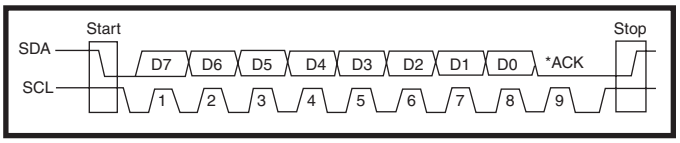


Figure 3—*I²C bus transactions are bracketed by unique start and stop conditions. Each 8-bit byte is followed by a ninth clock pulse for an acknowledge provided by the receiving device.*

DEVELOPMENT TOOLS

Technology improvements and the availability of powerful PCs have made the design of custom chips available not only to big companies, but also to engineers and hobbyists. FPGA vendors have developed more complex design software packages. They also have begun to make them available for free (or at a low cost) on the 'Net.

Xilinx's free WebPACK development system supports project design (schematics, VHDL, and Verilog) and the verification of the design with the powerful Mentor Graphics ModelSim logic simulator. The latter makes it easy to test the design by provide virtual tools like oscilloscopes, logic state analyzers, and test points.

The Xilinx package also allows you to migrate the design to its various chip families, including Virtex, Spartan-II, Spartan-3, CoolRunner, and XC9500 CPLDs. It's easy to port a microprocessor peripheral built for a big FPGA to a modest CPLD device.

The Spartan-II family in this implementation offers a decent number of logic gates at a reasonable cost. The I/O resources allow for a direct connection to the I²C bus. The project also could be ported to the new Spartan-3 family, but the I/O scheme must be redesigned because you can't disable the ESD protection on this family.

I originally developed my project on the Spartan-II, which has the ability to disable both the internal pull-ups and the ESD protection diodes in its I/O buffers. This allows the chip to be used in mixed-voltage systems because the pins can tolerate being pulled up by external resistors to 5 V.

The Spartan-3's ESD protection diodes can't be disabled. This has no effect on single-voltage systems, but in a mixed-voltage system, it's necessary to split the voltage domains using a pair of MOSFET transistors (see Figure 4).

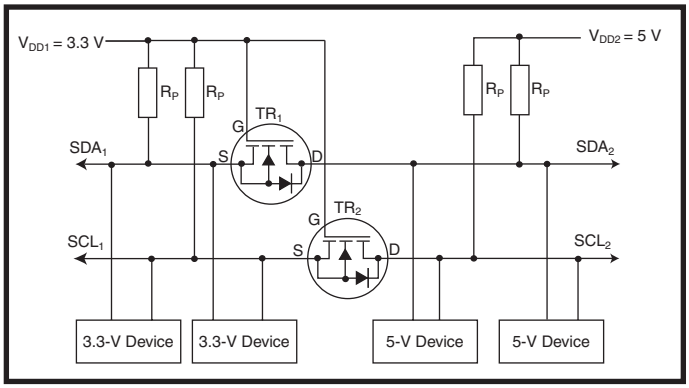


Figure 4—*Two N-channel MOSFETs can be used to separate an I²C bus into separate 5- and 3.3-V domains.*

FPGA DESIGN

The sample FPGA design contains the module ports for the two I²C bus pins, the module port and the dedicated buffer interface for the externally supplied clock, a readable register, and a single bit register that you can write to and that drives an LED for testing (see Figure 5). The I²C bus module in the middle does all the work. It provides two 8-bit buses: one is for input (I2CIN(7:0)) and one is for output (I2COUT(7:0). You can use the 16-bit decoded write strobe as an asynchronous write strobe signal. You can even use it as a clock enable signal with SCL as the activating clock.

The revision block is a trivial example of a read-only register. Reading register 5 will return the 0x33 value.

Inside the I²C bus module, the SDAIN and SCLIN input signals are sampled and filtered with a 125-ns clock. The resulting signals are used to create start pulses (every falling edge of SDA while SCL is high) and stop pulses (every rising edge of SDA while SCL is high).

A shifter and a comparator capture the first 3 bits of the serial stream and compare them to 110. If the received address doesn't match, the INVALIDADDRESS signal is asserted. If the address does match, the next 4 bits are taken as the register address. The next (and last) bit is the write or read flag bit. A low on the SDA line in the ninth bit position acknowledges the received byte. (This repeats for every subsequent byte in a write transaction.)

If the operation is a read, a read strobe (RDREG(7:0)) is generated at the same time the acknowledge is sent to the master. The input data is latched in a data shifter in preparation for the next group of clock pulses from the master. After the first byte, subsequent read strobes are generated only if the master acknowledges the current byte.

In a write operation, the output strobe (WRREG(15:0)) is generated only after the complete reception of the next byte. The output strobe is generated for each subsequent byte sent prior to a stop condition. This allows the realization of multi-byte registers or the initialization of RAM via I²C.

The user constraints file (.ucf) allows you to fix the device's pinout.

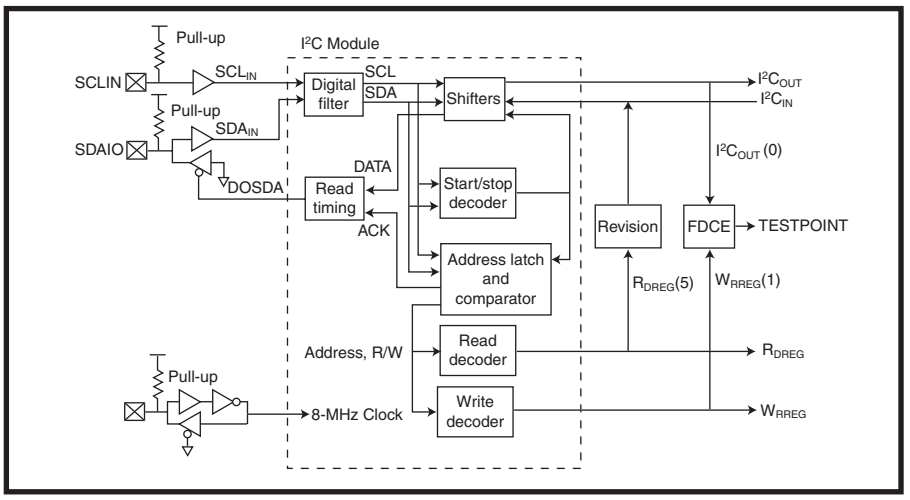


Figure 5—*The I²C module drops into an FPGA, requiring only two pins plus a system clock. This sample application shows an 8-bit read-only revision register and a single output bit. A low-precision oscillator for clocking internal logic can be created using an I/O buffer's pull-up resistor and pin capacitance, as shown in the lower left corner.*

Listing 1—*The behavior of the two external signals, SCLIN and SDAIO, is defined by the VHDL test bench.*

```
--***Test bench: User-defined section***
process
begin
SCLIN <= '0' ;
wait for 1000 ns ;
SCLIN <= '1' ;
wait for 1000 ns ;
end process ;

tb : PROCESS
BEGIN
SDAIO  <=  '1',
           '0' after 1500 ns,
           '1' after 8900 ns,
           '0' after 10200 ns,
           '1' after 17100 ns,
           '0' after 17600 ns,
           '1' after 18500 ns,
           '0' after 22500 ns,
           '1' after 30500 ns,
           '0' after 32500 ns,
           '1' after 50500 ns,
           '0' after 52100 ns,
           '1' after 55100 ns,
           '0' after 55700 ns,
           '1' after 56800 ns,
           '0' after 60500 ns,
           '1' after 68500 ns,
           '0' after 70500 ns,
           '1' after 93200 ns,
           '0' after 93800 ns,
           '1' after 94800 ns,
           '0' after 98500 ns,
           '1' after 102800 ns,
           '0' after 104500 ns,
           '1' after 106500 ns,
           '0' after 110500 ns,
           'Z' after 113000 ns;
           wait; — will wait forever
END PROCESS;
```

If the microcontroller or FPGA's pinout is tight, and if the initialization of the FPGA is done via the microcontroller, then the SDA and SCL pins can be assigned to the DIN and INIT pins, respectively.

A handy feature of the original Spartan devices was an available low-tolerance clock oscillator built into the FPGA itself. This feature isn't present in more recent Spartan devices. A way to overcome this limitation is shown in Figure 5.

This approach requires a dedicated I/O pin driven by an I/O buffer in Slow mode. An RC oscillator is created using the configurable I/O pull-up resistor and the inherent capacitance of the pin. This produces a clock that's approximately 8 MHz. This can be useful if there is otherwise no need for precise timing inside the FPGA. The clock also can be used outside the chip, in which case the additional capacitance of the attached circuitry will reduce the working frequency.

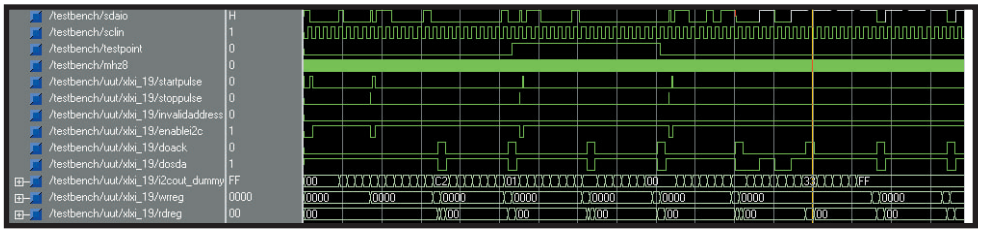


Photo 1—*The ModelSim waveform output shows three complete I²C transactions: a write operation that sets the test point high, a write that sets it low again, and a read of the revision register.*

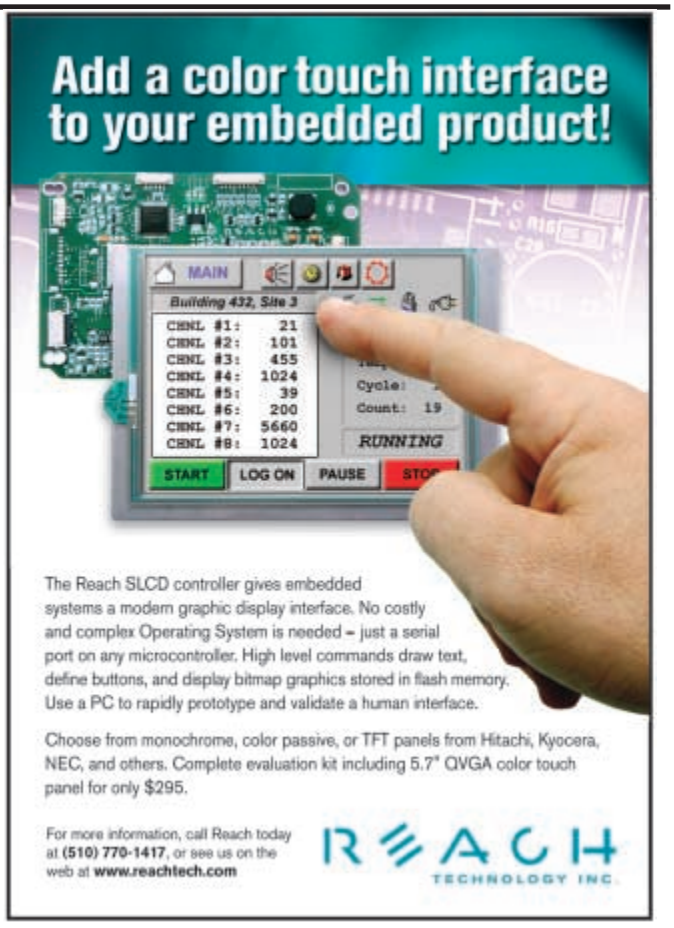
TEST BENCH

Another interesting feature of the Xilinx software package is the simulator, which allows you to test your design without having to implement it. Without this, you would have to create multiple test versions of your chip, bringing out different internal nodes to a limited set of external test points each time. The ModelSim XE II simulator accepts a VHDL test bench as input. The WebPACK design wizard automatically generates the file's skeleton when you create a new project. In this case, the test bench defines two processes to simulate the behavior of the two external signals, one for SCLIN and one for SDAIO (see Listing 1).

Photo 1 shows three full transactions on the I²C bus, starting with a write operation (the LSB of the first byte is cleared) from the microcontroller to the FPGA. This operation writes to register 1 with data equal to 1, setting the TESTPOINT output pin. The next transaction resets the TESTPOINT output by writing to register 1 with data equal to 0.

The last transaction in the simulation display is a read operation. The read bit (high) is easily identified as the last of 8 bits in the control byte.





The value of the I^2C_{IN} bus is shifted out onto the SDA line, as indicated by the white dashed line on the right end of the display.

FIRMWARE

Most compiler vendors provide library functions that support I²C interfaces. To implement an I²C master controller, it's usually just necessary to prepare the data to be sent and to call one or more library functions that do the work. This project was tested with PIC18C*xxx* and PIC16C*xxx* microcontrollers, and also with the Cypress EZ-USB FX2 8051-derived microcontroller, using software tools from CCS and Keil (see Listings 2 and 3).

The Keil compiler makes access to FPGA I²C interface particularly easy. In this case the address argument in the call shifts left by 1 bit inside the library routine.

Listing 2—*Use the CCS compiler with the FPGA access routines for PIC microcontrollers.*

```
void write_fpga (uchar regfpga, uchar *data, uchar numdati)
{
   write_firstfpga (regfpga);
   while (numdati– != 0) {
         i2c_write (*data);
         data++;
       }
       i2c_stop();
}
void write_firstfpga (uchar regfpga)
{
       i2c_start();
       i2c_write ((0xc0 | regfpga << 1) & 0xfe);
}
uchar read_fpga (uchar regfpga)
{
       uchar data;
       i2c_start();
       data=(0xd0| (regfpga)<<1) | 0x1;
       i2c_write ((0xc0 | (regfpga)<<1) | 0x1);
       data = i2c_read (0);  // Do not ack
       i2c_stop();           // Always to do after read!!!!!!!
       return (data);
}
```

VHDL IMPLEMENTATION

The schematic implementation is good as long as you're working with only one FPGA family. But if you have to migrate the project to another technology, the work becomes cumbersome and annoying. That's what happened to me. For the first time after a long period of using Xilinx design tools exclusively, I had to use the Altera design package because of a customer requirement.

The first design in the new environment was hard. I obtained a license and got the first simple project working. I soon discovered that the two systems are not so different, so I began to appreciate the unique features of using VHDL for designing.

After briefly studying some articles, I became more confident about the process of translating working projects to VHDL. My first attempts closely followed the structure of the schematic project. I took the schematic page and divided it into blocks (input filter, SDA and ack generator, read/write signal generator, and input shift register), replacing parts of the schematics with the newly written blocks of VHDL code and using the simulator tools extensively until I gained confidence. The last step involved replacing the entire sheet with a VHDL file in which the various blocks were included as components.

At this point, with the project running well on a Xilinx FPGA, I took the files and included them in a sheet in the Quartus II development package. It took little effort to have the project running both in simulation and on a real chip.



Listing 3—*Use the Keil MicroVision2 compiler with the FPGA access routines for the CY7C68013 EZ-USB microcontroller.*

```
BOOL WriteFpga (BYTE regaddress, BYTE length, BYTE xdata *dati)
{
      return (EZUSB_WriteI2C (regaddress+0x60, length, dati));
}
unsigned char ReadFpga (BYTE regaddress)
{
      unsigned char xdata temp;
      EZUSB_ReadI2C (regaddress+0x60, 1, &temp);
      return (temp);
}
```

GREAT EXPERIENCE

After having such a good experience with VHDL design, it's nearly impossible to imagine using only schematic entry for new projects. Contrary to expectations, it isn't true that you lose control of the allocation of hardware resources with this type of design.

Roberto Ferrabone earned a B.S.E.E. at the Politecnico di Torino in Turin, Italy. He co-owns Siro Automazione Srl, which is a firm involved in hardware and software solutions. You may reach him at roberto.ferrabone@sirioautomazione.it.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/176.

RESOURCES

P.J. Ashenden, "The VHDL Cookbook," University of Adelaide, South Australia, 1990.

Philips Semiconductors, "The I²C-Bus and How to Use It," April 1995, www.mcc-us.com/i2cHowToUseIt1995.pdf.

Synario Design Automation, "VHDL Reference Manual," March 1997.

SOURCES

Quartus II 4.1 web edition
Altera Corp.
www.altera.com

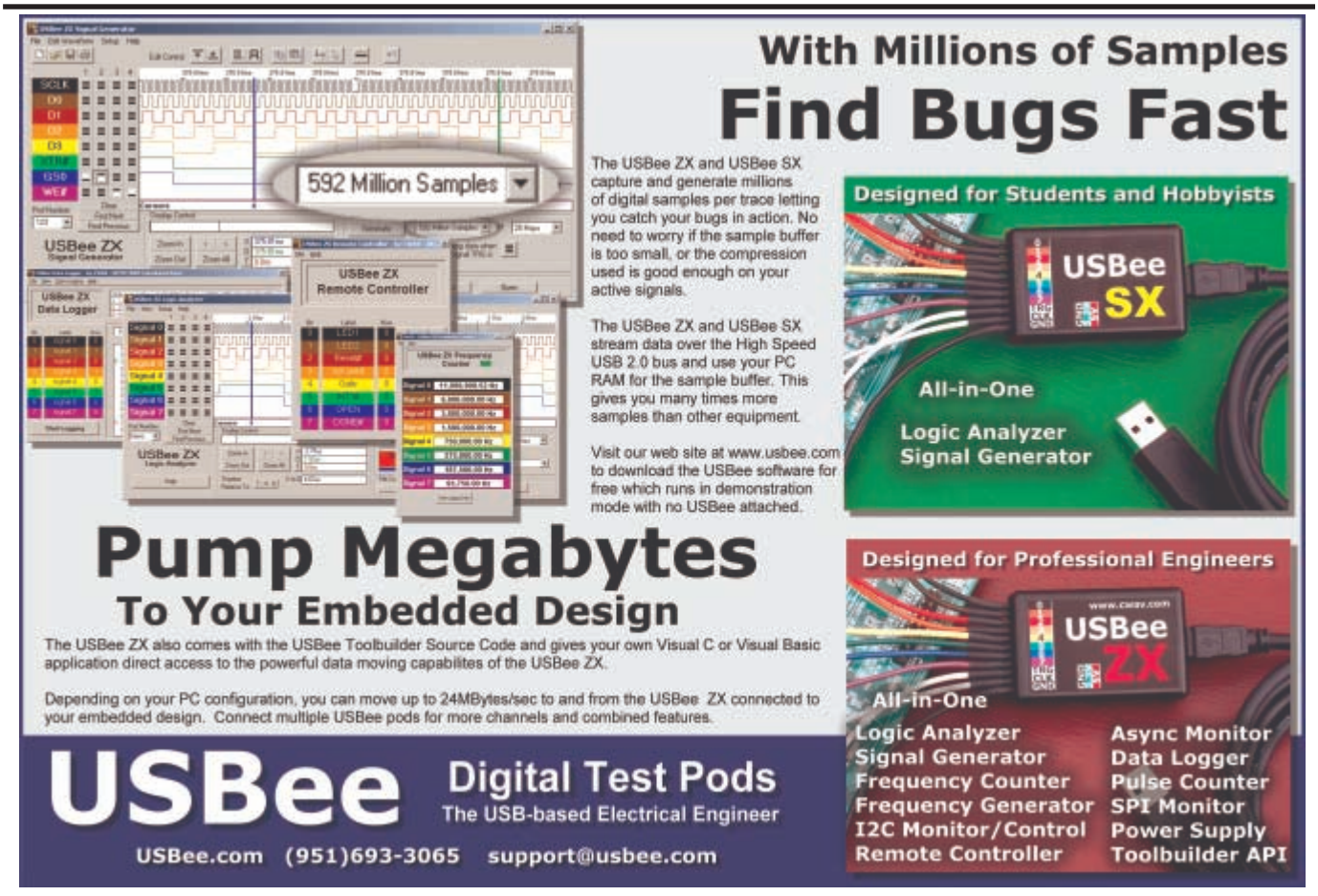
C compiler
CCS Inc.
www.ccsinfo.com

CY7C68013 EZ-USB microcontroller
Cypress Semiconductor Corp.
www.cypress.com

C51/Cx51 C compiler and MicroVision2 compiler
Keil Software
www.keil.com

ModelSim logic simulator
Mentor Graphics
www.mentor.com

WebPACK 6.2 design package
Xilinx
www.xilinx.com





Single-IC 10/100 Ethernet Solution

Thinking about implementing embedded Ethernet hardware? Try Freescale's MC9S12NE64. With this powerful tool, single-IC Ethernet is a no-brainer.

An ad describing a new Freescale Ethernet product recently caught my attention as I thumbed through my weekly bundle of electronic industry magazines. Freescale was marketing a single-IC 10/100 Ethernet solution. Hmm. Many vendors have claimed to be able to provide single-IC Ethernet, but most of them have required an extra PHY device to complete the single-IC package. With nothing more than a picture to go by, I figured the catch was that it would require a million external components to support the Ethernet IC. Or, maybe I'd have to completely retool for the new device. Or, even worse, maybe I'd have to write all the driver code from scratch and the IC would end up costing a fortune. Well, even with such negative notions, I wanted to see the thing in action.

MC9S12NE64 EXPOSED

Freescale isn't your father's Motorola. My new MC9S12NE64 development kit arrived in a multicolored box with pictures suggesting how I could put the single-IC Ethernet device to work. Of course, I wanted to see the actual development board and check out the single-IC Ethernet device. The discovery process was delayed a bit because I had to disrobe the MC9S12NE64 development board (formally known as the DEMO9S12NE64) by removing it from the precisely machined silk-screened plastic container in which it was mounted (see Photo 1).

After I extracted the MC9S12NE64 development board from its pristine enclosure, I looked for extra PHY ICs.

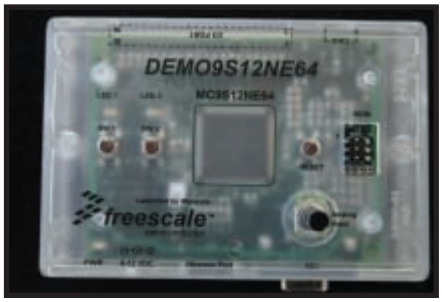


Photo 1—*My friend Mark (the machinist) marveled at the precision and care taken to produce this little case. The only initial confusion I had was which state (1 or 0) meant what to SW3.*

The only ones I came across were a MAXIM RS-232 converter and a standard set of logic gates housed in a 14-pin SOIC package. I was elated! Freescale indeed had delivered a true single-IC Ethernet solution. Photos 2a and 2b show the raw MC9S12NE64 development board before I put it back in its elegant translucent enclosure.

With my curiosity satisfied, I decided to learn more about the MC9S12NE64. The development kit includes a special version of the CodeWarrior C compiler, a trial CMX MC9S12NE64 TCP/IP stack, an open-source TCP/IP stack, and a ton of MC9S12NE64 documentation on the CD-ROM. There's also a demonstration version of Treck's real-time embedded Internet protocols.

The MC9S12NE64, which is based on Freescale's 68HC12 architecture, comes standard with 8 KB of RAM and 64 KB of flash memory. It's a 3.3-V part equipped with all the standard microcontroller amenities. If you call the roll, a couple of serial communications interfaces (SCIs) will answer, as will a four-channel timer, an I²C interface, an SPI interface, and a 10-bit eight-channel A/D converter. Of course, an Ethernet media access controller (EMAC) and a 10/100 Ethernet physical layer (EPHY) are also unique parts of the MC9S12NE64's electrical and logical mix.

The fast MC9S12NE64 can achieve a 40-ns minimum cycle time with a 25-MHz clock. Its on-chip debugging capabilities are designed to eliminate

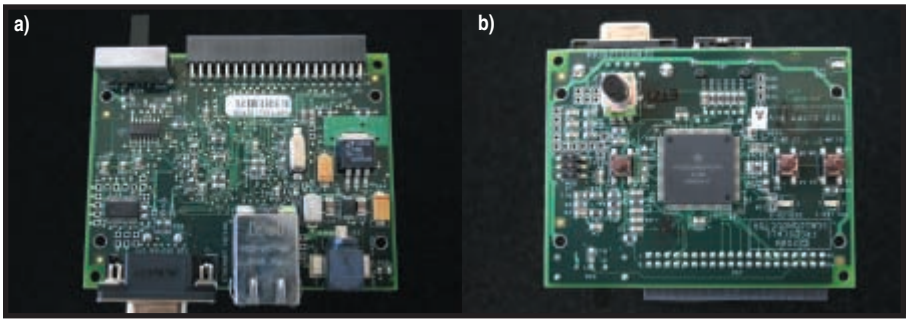


Photo 2a—*Notice that my question about SW3 is answered in the silkscreen legend of SW3. SW3 is a Load/Run switch. The MC9S12NE64 development board looks busy from this angle. However, if you ignore all of the holes and traces, you'll see that there are few components mounted here.* ***b—****Here's a look at the business side of the MC9S12NE64 development board. This is the 112-pin variant. If you decide to use the 80-pin MC9S12NE64, you'll have to add a heatsink pad on your PCB under the Ethernet IC.*

the need for what Freescale calls box emulators. However, the MC9S12NE64 development board is fitted with a six-pin background debug (BDM) pin set. When I saw this, I got my trusty P&E Microcomputer Systems Cyclone Pro and ICD12Z programming and debugging tools ready. Although crawling through hex dumps and source code can be a lot of fun, I was anxious to find out just what I needed to make the MC9S12NE64 go as far as hardware was concerned. Judging from what I'd seen, I didn't think I'd have too much to do on the hardware side to send some packets over the wire.

SUPPORTING HARDWARE

The MC9S12NE64 is a microcontroller internally surrounded by subsystems such as an A/D converter, communications interfaces, and a full-blown Ethernet MAC and PHY in this case. The MC9S12NE64 is a microcontroller, so it needs what all microcontrollers need. Power's a given, and it's supplied to the MC9S12NE64 in the normal manner using a wall-wart/linear fixed-voltage regulator scheme.

The MC9S12NE64 development kit uses a Linear LT1086CM-3.3 fixed-voltage regulator. I'm sure Freescale buys in quantity, so the cost of the LT1086CM-3.3 probably isn't of any real consequence. A less expensive alternative is National Semiconductor's LM1086CS-3.3. It's great if cost is an issue and you aren't buying voltage regulators by the thousands. In either case, you'll get the necessary 3.3 VDC at the voltage regulator's output pin, and that's all that really matters.

The MC9S12NE64 uses an internal 2.5-VDC voltage regulator to supply correct voltages to the CPU core, the PLL subsystem and the Ethernet PHY electronics. The on-chip 2.5-VDC regulator eliminates the need to supply 2.5 VDC externally with an additional power supply. If you're hardheaded, you can bypass the internal 2.5-VDC regulator and supply your own version of CPU core voltage. Freescale frowns on going about supplying the CPU core voltage the hard way.

I'm talking about a microcontroller here, so clocking is a must. To drive the on-chip Ethernet PHY at 100 Mbps, a 25-MHz internal bus clock is required. The crystal-based 25-MHz input clock signal is used by the MC9S12NE64's PLL to synthesize the internal bus clock of 25 MHz and to drive the CPU and the Ethernet PHY. If your application doesn't need 100 Mbps, you can set the internal bus clock as low as 2.5 MHz. The MC9S12NE64 datasheet and application notes suggest running the internal bus clock at 16 MHz for 10-Mbps operation. In either case you'll need the 25-MHz crystal clock to supply clock to the PLL.

I went through the math in the MC9S12NE64 datasheet concerning the PLL loop filter component values. After numerous iterations through the formulas, nothing added up to match the capacitor and resistor values in any of the MC9S12NE64 development board schematics.

In the MC9S12NE64 schematics, the parallel and serial PLL loop filter capacitors were set to 470 and 4,700 pF, respectively (see Figure 1). The PLL filter resistor value varied between 2.2 and 5.1 kΩ. The DEMO9S12NE64 used 5.1 kΩ and a differing MC9S12NE64 development board (EVB9S12NE64) with 2.2 kΩ. All of the MC9S12NE64 schematics showed the 2.2 kΩ value with the same 470- and 4,700-pF capacitor pair.

Thinking that I was missing something, I searched the Freescale web site for clues that would help me make sense of the PLL loop filter calculations. I thought I had struck gold when I came

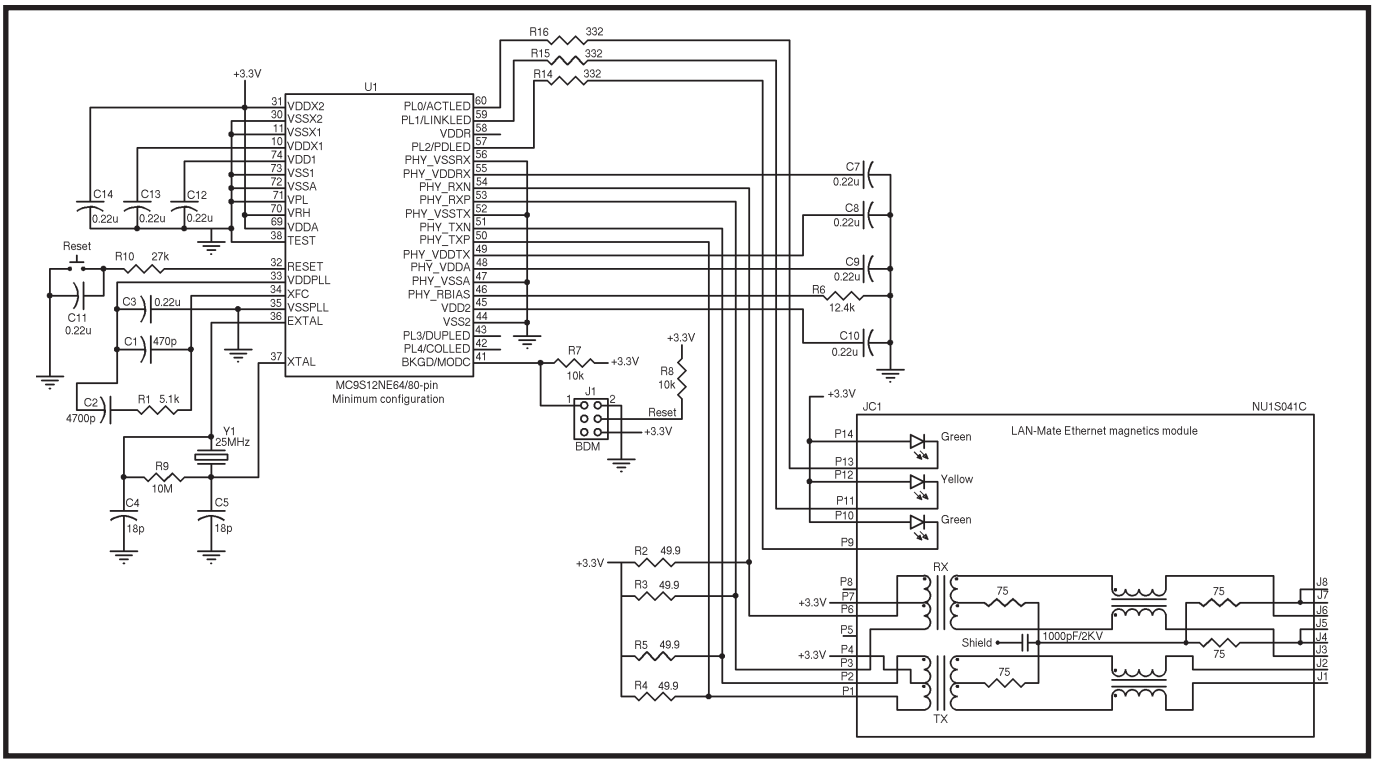


Figure 1—*This is child's play. However, the MC9S12NE64 development kit package comes with some hints about what to do and what not to do when you lay out your MC9S12NE64 PCB.*

across an HCS12 PLL loop filter component calculator. I plugged the SYNR and REFDV values from Listing 1 into the PLL loop filter component calculator and ended up with results that were far from the component values found on my MC9S12NE64 development board. In fact, my manual calculations were extremely close to those offered up by the HCS12 PLL loop filter component calculator. The HCS12 PLL loop filter component calculator consistently set the capacitor values at a factor of four less than those posted in the schematics.

Rather than make any incorrect assumptions, I posed a question to the Freescale technical support staff. As luck would have it, the Freescale web site was going through an upgrade, so I had to wait for an answer. Refer to the *Circuit Cellar* ftp site for more information about PLL formulas.

One of the PLL calculations (f_{PLL}) worked fine. Most of the values you need in order to cipher the PLL frequency (f_{PLL}) can be obtained from the source code in Listing 1. To calculate the f_{PLL}:

$$f_{PLL} = \frac{2 \times (SYNR + 1)}{REFDV + 1} \times OSCCLK$$

If you use the values from Listing 1 to fill in the blanks, everything inside the brackets equates to two. The *OSCCLK* value is 25 MHz (see Figure 1). That gives you an f_{PLL} of 50 MHz. *SYNR* is a 6-bit value that controls the PLL's multiplication factor. To use the *SYNR* value to alter the PLL frequency, set the PLLSEL bit. That's done with `CLKSEL_PLLSEL = 1`. Setting the PLLSEL bit gets the system clocks from the PLL clock.

With the PLLSEL bit set, you can alter the *OSCCLK* value with the numerator portion of the f_{PLL} formula as a multiplier for the OSCCLK signal. The denominator portion of the f_{PLL} formula also depends on the PLLSEL bit being set. *REFDV* is a 4-bit value that defines the granularity of the PLL multiplier steps. The PLL multiplier granularity is determined by dividing the OSCCLK signal by REFDV + 1. With the PLLSEL bit set, the MC9S12NE64 CPU core clock is equal to f_{PLL} and the bus clock is equal to half the core clock. You need a bus clock of 25 MHz to run the MC9S12NE64's Ethernet peripherals at 100 Mbps, and you have it according to the code in Listing 1.

Listing 1—*This little bit of code speaks volumes. As you can see in Photo 3 (page 70), the proof in the pudding is backed up in a debug screen as seen via the HCS12SerialMon serial monitor/debugger.*

```
//System clock initialization
CLKSEL=0;
CLKSEL_PLLSEL = 0;        //Select clock source from XTAL
PLLCTL_PLLON = 0;         //Disable the PLL
SYNR = 15;                //Set the multiplier register
REFDV = 15;               //Set the divider register
PLLCTL = 192;
PLLCTL_PLLON = 1;         //Enable the PLL
while(!CRGFLG_LOCK);      //Wait
CLKSEL_PLLSEL = 1;        //Select clock source from PLL
```

I found another mystery-meat resistor value attached to the MC9S12NE64's PHY R_{BIAS} pin. I searched the MC9S12NE64 documentation over and over, but the references to the R_{BIAS} pin's value were nowhere to be found. Each reference to R_{BIAS} pointed me to the electrical characteristics area of the MC9S12NE64 datasheet. Examining the MC9S12NE64 development board schematics revealed a constant 12.4 kΩ across all of the development boards. It became obvious that 12.4 kΩ was the value of choice, but it would be nice to know how it came about. After a couple of days of searching, I looked for "bias" instead of "Rbias," and found the elusive value of 12.4 kΩ in the EPHY operating conditions (Table A-24).

Even if you decide to do absolutely nothing else with the MC9S12NE64 but use the Ethernet peripherals, you must provide a LAN interface. The MC9S12NE64 development board is fitted with Bel Fuse integrated magnetics identified (part number 0810-1X1T-08). In my experience, integrated Ethernet magnetic parts are somewhat difficult to obtain after the part is found to be carried by a particular distributor. So, I took it upon myself to find the Bel Fuse part. To my surprise, Avnet carries it. The only drawback is that you must order a minimum of 60 sets of magnetics to meet the minimum. However, it isn't a bad deal for approximately $2.

I'm partial to Bothhand Ethernet magnetics because they're easy to find. The Bothhand NU1S041 is electrically identical to the Bel Fuse 0810-1X1T-08 and just a bit more expensive. The major differences between the parts are the way the LEDs are mounted and the footprint. The Bel Fuse isolation magnetics use a set of light pipes to carry the light from LEDs mounted externally to the rear of the magnetic package. The Bothhand LEDs are totally enclosed in the can. They are mounted flush with the front of the magnetics can.

Note that the MC9S12NE64 supports the status LEDs that are contained within the integrated magnetic shell much like any other Ethernet engine would. The status LEDs are optional. However, the MC9S12NE64 allows you to turn them off and save the dedicated status LED port pins for other purposes. The downside to this form of control is that all of the LED status driver pins are either on or off. If only one status signal is needed, you could use a nondedicated MC9S12NE64 I/O pin to drive the status LED via firmware.

The Ethernet isolation magnetics are pretty much a no-brainer. The circuitry and even the range of resistor values used by the MC9S12NE64 development board (49.9 to 51.1 Ω) match those used by the 10/100-Mbps ASIX 88796 L I use in my high-speed Ethernet designs. The MC9S12NE64 isolation circuitry uses a single capacitor while the ASIX 88796 L configuration requires three capacitors. In addition to the Bothhand integrated package, the MC9S12NE64 datasheet notes at least three more manufacturers and their isolation magnetics part numbers that can be used in an MC9S12NE64 design.

Odds are you won't get your initial MC9S12NE64 IC loaded with serial bootloader/debugging code. So, it may be prudent to add a BDM interface just as the MC9S12NE64 development board designers have done. You can reset via the BDM, but adding a simple SPST switch-resistor-capacitor reset circuit to the MC9S12NE64's reset pin is a good idea.

Listing 2—*It's a simple substitution. Now that you know that you'll most likely have to reconfigure a PC to run your MC9S12NE64 development board, doing IP readdressing may become one of your first official tasks.*

```
#include "MOTTYPES.h"

const tU08 hard_addr [6] = { 0x01, 0x23, 0x45, 0x78, 0x89, 0xab };
//Default values
//const tU08 prot_addr [4] = { 192, 168,   2,   3 };
//const tU08 netw_mask [4] = { 255, 255, 255,   0 };
//const tU08 dfgw_addr [4] = { 192, 168,   2,   1 };
//const tU08 brcs_addr [4] = { 192, 168,   2, 255 };

//Florida room modifications
const tU08 prot_addr [4] = { 192, 168,   0,   140 };
const tU08 netw_mask [4] = { 255, 255, 255,   0 };
const tU08 dfgw_addr [4] = { 192, 168,   0,   1 };
const tU08 brcs_addr [4] = { 192, 168,   0, 255 };
```

All that's left is to make sure the MC9S12NE64's power supply pins are properly bypassed. That's it. Freescale has left almost nothing to chance on the firmware side. The MC9S12NE64 package comes with a boatload of example code. Whatever you don't get with the development board can be downloaded from the web site.

FOOLPROOF ETHERNET

The MC9S12NE64 development kit gives you a number of firmware roads to travel to arrive at Ethernet connectivity. Source code is provided for the basic Internet protocols and applications.

The MC9S12NE64 comes preloaded with a serially accessible monitor program. The idea is to allow you to evaluate and debug the development board without having to tool up with box emulators and programmers. The monitor program also includes some code to exercise the switches, buttons, and Ethernet electronics on the development board.

The MC9S12NE64's default IP address didn't jive with the Florida room's LAN. So, I used a laptop to initially check out the MC9S12NE64 development board because I could easily change the laptop's IP address to match the board's default IP address.

The development kit also includes an Ethernet crossover cable for those of you who don't have elaborate networks to attach to. It's also for designers like me who have networks but don't want to change to accommodate the MC9S12NE64 development board.

After I was able to press the correct buttons to get to the Ethernet portion of the code, I pinged the MC9S12NE64 development board. The next step involved altering the default IP address so I could put the development board on the Florida room's LAN and continue my MC9S12NE64 education.

Before I could continue, I had to install the special edition of CodeWarrior and register the package. After my online registration was accepted, I received a 30-day license key via e-mail. The entire load and registration process took less than 15 min. on the 'Net.

I clicked on Startup_Demo.mcp, which contains the CodeWarrior source package that I downloaded and unzipped from the MC9S12NE64 development board's CD-ROM. CodeWarrior started automatically and loaded in the source package. The default IP address set and a dummy MAC address are located in the address.c file.

As you can see in Listing 2, I simply changed the values of the IP address set to fit into the Florida room's test LAN. After recompiling the new image, I simply dropped into Debug mode from the CodeWarrior IDE. The serial monitor/debugger program was coded to erase the MC9S12NE64 and reprogram the incoming application code. Thus, the MC9S12NE64's preload grabbed the new image and stuffed it into the MC9S12NE64's flash memory. I ran the newly loaded and addressed Startup_Demo from the debugger and was able to ping my new IP address of 192.168.0.140.

After I had gotten my feet wet, I decided to blow the entire thing away and to try to recover using the P&E Microcomputer Systems Cyclone PRO programmer/debugger. I connected the Cyclone PRO to the MC9S12NE64's BDM interface and fired up the P&E PROG12Z application.

The first thing the PROG12Z application wants is to acquire the desired programming algorithm. The programming algorithm for the MC9S12NE64 wasn't in the algorithm list. A quick visit to the P&E site provided me with a cross-reference list that specified the programming algorithm 9S12dj64_64k.12P as MC9S12NE64-compatible. After the correct programming algorithm was loaded into the Cyclone PRO, the rest of the programming options were made available. I wiped the MC9S12NE64 and in the meantime wiped all of the MC9S12NE64 development board source code packages from the PC. I then downloaded the HCS12SerialMon serial monitor/debugger and monitor start-up packages anew from the MC9S12NE64 development board CD-ROM. I was starting with a totally clean slate.

I selected the .S19 version of the monitor start-up package and attempted to program the MC9S12NE64 with this file. No go. There were some issues with logical and physical addressing that the programmer didn't want to deal with, and I really wasn't in the mood to try to fix memory maps in the program command files. As it turns out, the same files with the .phy extension were also available. These files were really what the programming algorithm wanted.

After the addressing issues were taken care of, I was able to load and program Monitor.abs.phy into the blank MC9S12NE64. The development board's LEDs that are associated with the push-button switches came back to life, and I was able to run the start-up demo program and ping the MC9S12NE64 box. The start-up monitor and demo program were in place. All of the communications interfaces were active and accessible. However, I can't use the features of the stand-alone HCS12SerialMon serial monitor/debugger currently because it isn't loaded.

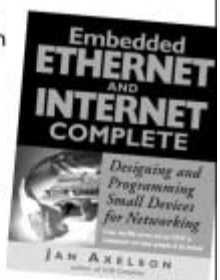
The HCS12SerialMon serial monitor/debugger is an assembler program that's normally stashed away in a protected area of program flash memory (see

PIC-SERVO
Motion Control
Low-Cost Motion Controllers for Brush, Brushless and Stepper Motors
New PIC-SERVO SC Single Chip Servo Controller!

▪ Controller Chips
▪ Controller Boards
▪ Integrated Amplifiers
Visit our web site for a complete selection of controllers and accessories
www.jrkerr.com
JEFFREY KERR, LLC

USB DAQ
LabJack U12
Available now for only ...
$119 qty 1
($99 qty 10+)

USB Data Acquisition & Control
* Built-in screw terminals
* Easy-to-use USB
* Everything included
* Use with C, VB, LabVIEW, etc.
* Windows 98SE/ME/2000/XP
* 12-bit analog inputs (8)
* 10-bit analog outputs (2)
* 20 digital I/O
* 32-bit counter
Made in Colorado, USA, by LabJack Corp.
info@labjack.com, (303) 942-0228
www.labjack.com

Put Your Devices on the Internet
(and in local networks too)
Networking isn't just for PCs. Learn how to design and program small devices that exchange data via TCP/IP, host Web pages, communicate using e-mail and FTP, and much more!


Embedded Ethernet and Internet Complete
by Jan Axelson
ISBN 1-931448-00-0 $49.95
Lakeview Research www.Lvr.com
From the author of USB Complete

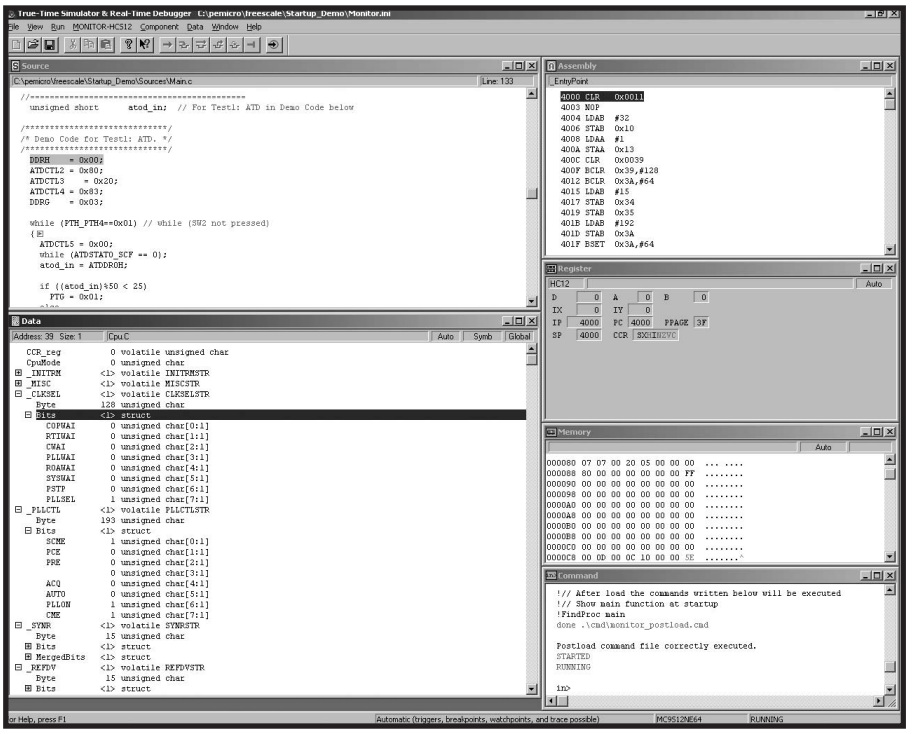


Photo 3—*There's a breakdown of all of the MC9S12NE64 clock stuff in the Data window. In addition, this shot gives you a good idea of what the stand-alone monitor/debugger/simulator can do.*

Photo 3). To get the HCS12SerialMon serial monitor/debugger code programmed back into protected memory, a BDM-equipped device like the Cyclone PRO is necessary. It can be commanded to unprotect and program the protected flash memory block from within the start-up monitor program. A command set to do this without user intervention is included in the HCS12SerialMon code package on the CD-ROM.

All I had to do was load the HCS12SerialMon serial monitor/debugger source assembler file into a CodeWarrior IDE session, enter the Cyclone PRO-based monitor mode via the debug door of the CodeWarrior IDE, and allow the predetermined set of BDM commands to execute against the Cyclone PRO hardware. I was then able to remove the BDM interface from the MC9S12NE64 and access the development board for debugging using only the board's serial interface and the HCS12SerialMon monitor.

ETHERNET NIRVANA

The MC9S12NE64 kit makes no assumptions about how much you know about implementing embedded Ethernet hardware. Freescale gives you every tool necessary to make it happen. As I said, it isn't your father's Motorola. Freescale has proven that single-IC Ethernet isn't complicated; it's embedded.

Fred Eady 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. Fred may be reached at fred@edtp.com.

PROJECT FILES

To download PLL formula information, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/176.

RESOURCE

Freescale Semiconductor, "MC9S12NE64 Datasheet," MC9S12NE64V1, rev. 1.0, 2004.

SOURCES

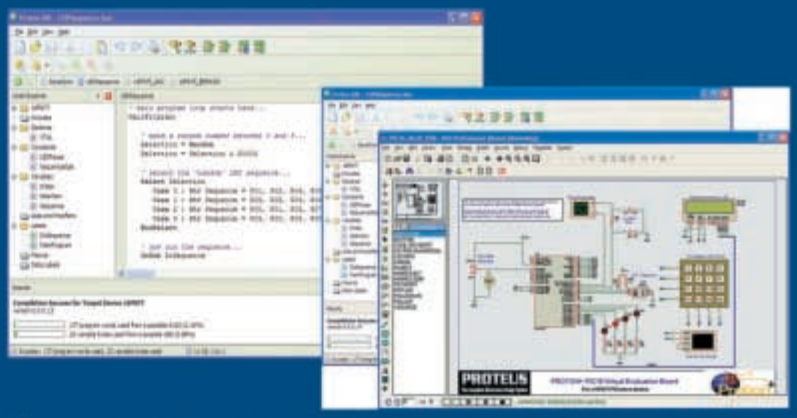
NU1S041 Module
Bothhand Enterprise, Inc.
www.bothhand.com

MC9S12NE64 Microcontroller
Freescale Semiconductor
www.freescale.com

Cyclone PRO and PROG12Z
P&E Microcomputer Systems
www.pemicro.com

PIC Programming Made Easy!

Proton+ PICBASIC Development Suite



Next Generation IDE

Proton IDE is a professional and powerful visual Integrated Development Environment (IDE) which has been designed specifically for the Proton Plus compiler. Proton IDE accelerates product development in a comfortable user environment without compromising performance, flexibility or control.

- *Code Explorer*
- *Compiler Results*
- *Programmer Integration*
- *Integrated Bootloader*
- *Real Time Simulation Support*
- *Serial Communicator*
- *Online Updating*
- *Plugin Architecture*

INTRODUCTORY OFFER! Save $10 LIMITED TIME

SAVE TIME WITH END TO END INTEGRATIONS

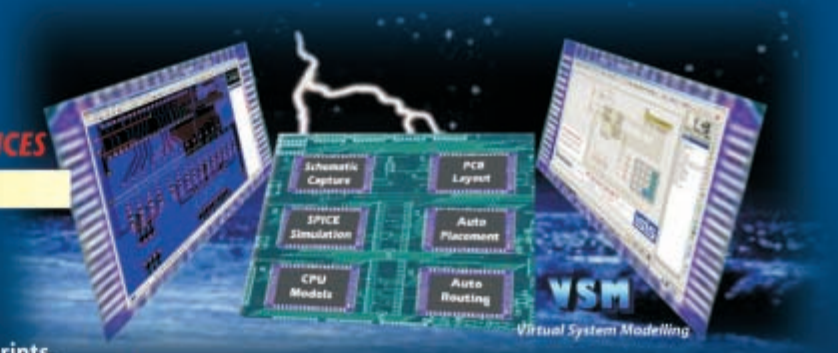
NEW IDE - New IDE makes development using Proton+ even faster and more intuitive!
COMPILER - The popular Proton+ compiler has enhanced support for I²C, SPI, Dallas 1-wire bus, RS232, X10, Compact Flash Memory Cards and USB.
VIRTUAL SIMULATION - Simulate your project in RealTime using the integrate Proteus Virtual PIC Boards.

Visit www.r4systems.com to see our latest projects using **Proton+**

PROTEUS

The Complete Electronics Design System

EASY TO USE CAD TOOLS AT FANTASTIC PRICES



Schematic and PCB Layout

- Powerful and flexible schematic capture.
- Auto-component placement.
- Rip-up and Retry PCB routing.
- Polygonal gridless ground planes.
- Library of over 8000 schematic and 1000 PCB foot prints.
- Bill of materials, DRC reports and more.

Mixed Mode SPICE Circuit Simulation

- Berkeley SPICE3F5 simulator with custom extensions for true mixed mode and interactive simulation.
- Six virtual instruments and 14 graph based analysis types.
- 6,000 models including TTL, CMOS and PLD digital parts.
- Fully compatible with manufacturers' SPICE models.

Proteus VSM - Co-simulation and debugging for popular Micro-Controllers

- Supports PIC16 & PIC12, AVR, 8051, HC11 and ARM micro-controllers. Latest version includes 40 new PIC18's.
- Co-simulate target firmware with your hardware design.
- Includes interactive peripheral models for LED and LCD displays, switches, keypads, virtual terminal and much more.
- Provides source level debugging for popular compilers and assemblers from HiTech PICC, Crownhill, IAR, Keil and others.

PCB AutoRouting

- Proteus PCB design includes an interface to the Electra Gridless autorouter.

"For the cost of the software compared to the productivity gains, I consider Proteus to be pivotal in the commercial viability of my company and by far represents the best value for money of anything Tempus possesses."

ROB YOUNGS, Tempus Consulting

FREE DOWNLOADABLE DEMO

Save Time. Save Money.

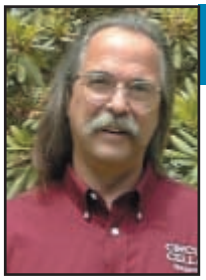
Proteus Starter Kit – $199

Complete Systems from $449

"This is clearly superior in every respect."

R4 SYSTEMS INC.

www.r4systems.com
Tel: 905•898•0665 info@r4systems.com



Joystick Technology

An analog joystick would be a great addition to some designs, but space limitations could present a problem. Fortunately, new devices like the JS1100AQ five-position joystick navigation switch can help you solve the space problem.

If you've watched an adventure movie featuring aircraft, you've probably encountered someone yelling, "Pull up! Pull up! You're gonna crash!" A huge explosion in which one of the good guys dies in a ball of flames usually occurs a few moments later. The mechanical stick aviators hold so dearly in such situations allows them to control the ailerons and elevators used for adjusting a plane's roll and pitch. Control is maintained so long as there isn't a malfunction in the hydraulics or components that connect the control stick to the movable flaps. Ejection seats come into play when a malfunction occurs.

To those of you with your feet planted more permanently on the ground, a joystick may be more familiar as a video game input device. Without ever having to leave the safety of your comfy chair, you can operate a plane, boat, tank, or car. You also can run through the PC's virtual environment of your choice. If you're an RC hobbyist, you can eliminate the virtual environment by using a joystick to maneuver your vehicle through the real world.

In the late '70s, Atari introduced the digital joystick on the 2600 gaming system. I use the term "digital" because the 2600 used five switches: up, down, left, right, and fire. IBM adopted the analog joystick for its PC in 1981. Let's take a look at each one.

JOYSTICK ANATOMY

The joysticks I've described came in two varieties, analog and digital. Both were useless without the right input circuitry and drivers. The digital joystick was nothing more than push buttons. Four switches were arranged around the stick: one above, one below, one to the left, and one to the right. The stick pressed one when pushed up, down, left, or right. Note that 45° stick movements pressed on two buttons at once.

The four button states were decoded into eight directions. The fifth button initiated actions like firing and throwing. Although digital inputs were directly compatible with the digital computer systems for which they provided input, the software had to determine the positioning on the screen by measuring how long a direction button was closed on the joystick and adjusting the reference point on the screen. The digital joystick couldn't indicate a particular position on the screen (or how fast to move); it only indicated movement in one of eight directions from the present location.

IBM's analog joystick used two potentiometers at right angles to each other to measure stick movement. One potentiometer rotated when the stick moved up and down. The second potentiometer rotated when the stick moved left and right. The potentiometer's value indicated direction as well as a proportional change in that direction. The two potentiometers provided this proportional change in a direction in addition to a proportional direction. This added a new dimension to accurate positioning.

Two push buttons completed the analog input device. However, the device required input circuitry for interpretation. Because ADCs were expensive at the time, a simpler interface was used. The resistance of each potentiometer (wired as rheostats) provided the "R" of an RC timing circuit that produced a proportional pulse-width output. The pulse width of the digital signal could be measured by the software and interpreted as position. The analog inputs could now directly access any screen position as an x,y coordinate. This was far more powerful than the information gleaned from a digital joystick.

With today's advances in technology, a microcontroller's on-board ADC could easily read the analog joystick. Although present-day analog joysticks actually

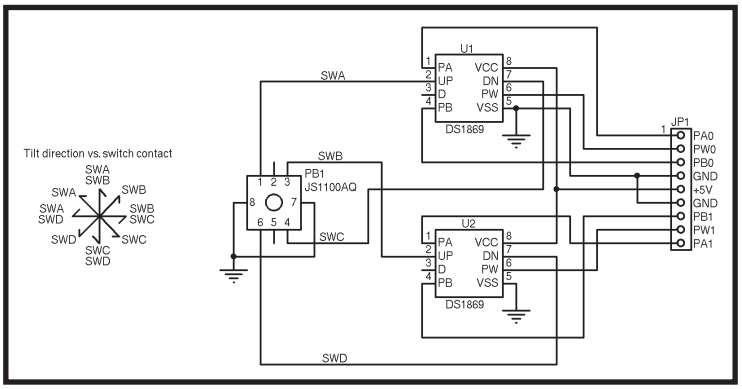


Figure 1—*By connecting a JS1100AQ to multiple DS1869s, the directional switch closures can enable the internal incrementing and decrementing features of each of the digital potentiometers.*

present a current (rheostat or adjustable resistor tied internally to 5 V), they're easily rewired to produce a voltage from the potentiometer's wiper if the ends of each potentiometer are connected to 5 V and ground, respectively.

An analog joystick would be the perfect solution for some designs; however, space limitations sometimes make that impossible. Fortunately, you can combine a few new components to make this happen in the space allotted. Let's take a look at a few possibilities.

NO BRAIN

E-Switch makes the JS1100AQ, which is a five-position joystick navigation switch. It's essentially five switches in one. A vertical press makes one set of contacts. A tilt makes contact with one or two of the four-corner contacts in a similar fashion to Atari's digital joystick.

The JS1100AQ is a surface-mount part, measuring approximately 0.3" square. If you mate the switch to Maxim DS1869 digital potentiometers, tilting the JS1100AQ toward a corner will engage a single switch (see Figure 1). Opposite corner switches are tied to the increment (up) and decrement (down) inputs of the same DS1869. This allows for the independent action of each digital potentiometer or a combined action of adjacent switches by tilting to a side.

The DS1869 has 64 taps over its total resistance (available in 10, 50, and 100 kΩ). It will automatically increment or decrement at a rate of 100 ms if an input is held for at least 1 s. Each of the potentiometer's resistor chains is connected across the 5 V and ground supplies in the circuit (although you could use other voltages as long as they fall within the chip's supply limit of –8 to 8 V). Unfortunately, this is the only manually operating digital potentiometer I've found. If you need more resolution, you must add some smarts.

GOT BRAINS?

Many digital potentiometers are available with serial-type interfaces because of the small number of I/Os required for communication. SPI and I²C are the most popular interfaces. These devices have expanded from the first 32-tap potentiometers to up to 256 taps, which allow the wiper to be positioned in increments of 0.4% of the total resistance, almost an order of magnitude better than their 32-tap predecessors.

Instead of the aforementioned simple interface, these parts now require some smarts so a simple switch input device like the JS1100AQ can control them. A microcontroller with at least eight I/Os can interpret the switch inputs and set the digital potentiometer. Five inputs monitor the JS1100AQ switches. You can use three output bits to communicate with the digital potentiometer (see Figure 2). I ended

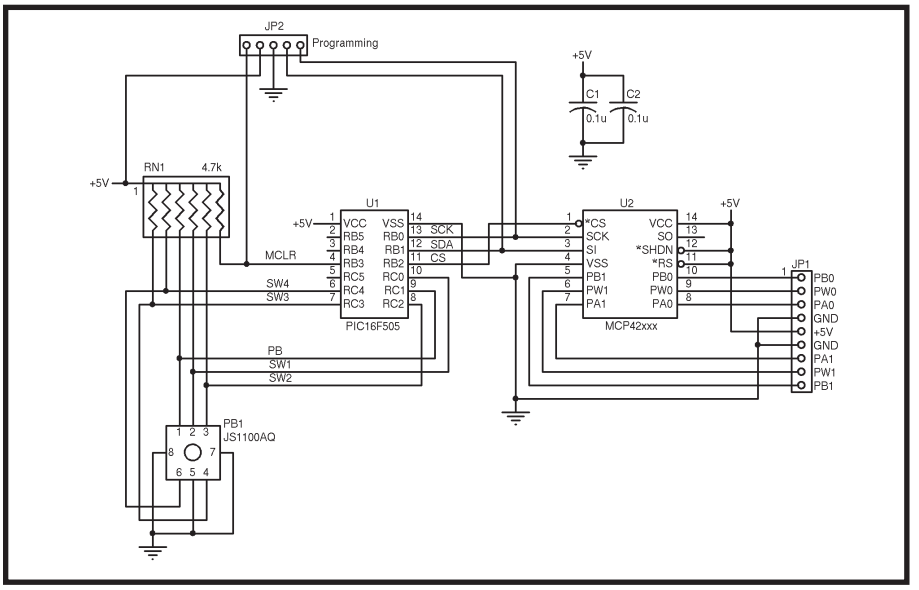
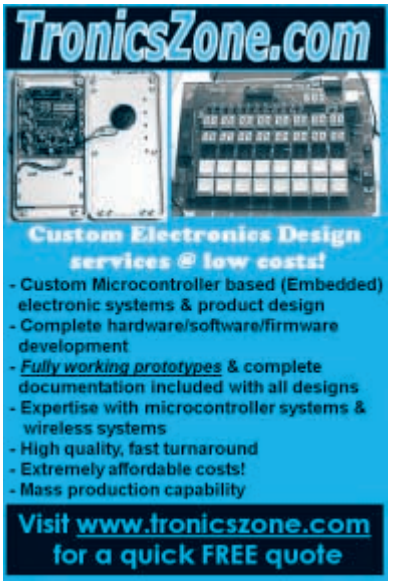


Figure 2—*By choosing a microcontroller with at least eight I/Os, all five switches in the JS1100AQ can be monitored with enough outputs left over for a software SPI interface.*





Profiting from the
1st New Industry
of the 21st Century



Produced by

RoboticsTrends

IDG WORLD EXPO





SECOND ANNUAL

ROBOBusiness
Conference and Exposition

The Nation's Premier Business Development Event for Mobile Robotics and Intelligent Systems



May 10-11, 2005
Hyatt Regency Cambridge
Cambridge, MA

www.roboevent.com



Premier Sponsor

iRobot

Corporate Sponsors

CTG
mBus. machineBus.com
SYSTRONIX Embedded Java Spoken Here
CoroWare
ROGUE ROBOTICS
VALDE Systems, Inc.
FOCUS robotics
ShopBot The Workshop Robot
Frontline ROBOTICS Mobile Security & Response www.Frontline-Robotics.com
SICK

Media Sponsors

ABIresearch
CIRCUIT CELLAR
Entertainment Engineering
NetworkWorld
Nuts & Volts
SERVO

Association Sponsors

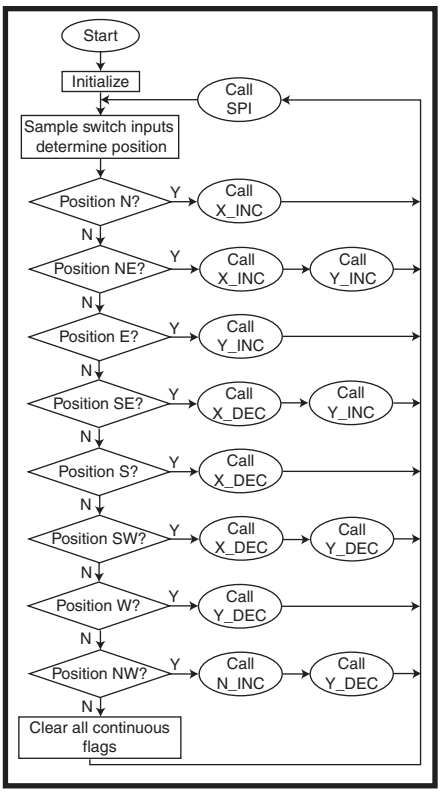


Figure 3—*A simple loop is used to monitor switch inputs, increment and decrement x- and y-axis variables, and communicate the values to an external digital poteniometer via an SPI interface.*

up with nil after a cursory search for small I/O microcontrollers with a serial interface, so the SPI interface had to be a software function. This wasn't a big deal because little else required lengthy time requirements.

The microcontroller's basic functions (besides communicating with the digital potentiometers) are to monitor the JS1100AQ for up, down, left, and right contact closures and to increment or decrement the x and y variables. The algorithm used for this might be as simple as incrementing or decrementing the variable once for each contact closure. You may wish to implement a more complicated scheme to automatically step the variable's value if the contact remains closed.

This will become more complicated if you dynamically change the step size based on time. Much of this would depend on how you'll use the output. If it's used as a simple volume control, then an adjustment response in the seconds range might be suitable. Note that 1-s responses might be required for pointing purposes. Subsecond moves are necessary for gaming because a perceivable time lag makes gaming difficult. Whether position feedback is a bar graph, a cursor, or a paddle, your eye, brain, and finger close the feedback loop by automatically compensating for errors.

Figure 3 shows a simple program loop that looks at the JS1100AQ input switches and determines which way the JS1100AQ is being tilted—position 1 (north) to position 8 (northwest). This allows you to assign a position based on any combination of switches, and you can compensate for a JS1100AQ rotation or the swap of an axis. Depending on the position, subroutines are called to adjust `XVal` or `YVal`. These values are used to set the taps on digital potentiometer 1 and potentiometer 2 every time through the main loop during the call to the SPI subroutine. The SPI subroutine requires 500 µs, which is the longest length of time to execute. It's executed every time to act as a timer. Therefore, timers aren't necessary.

The `X_INC`, `X_DEC`, `Y_INC`, and `Y_DEC` subroutines are all similar. Figure 4 shows the X_INC subroutine. A flag for each switch is cleared unless the switch is held in the closed position. The `XSkip` and `YSkip` variables begin with a value of 255. These variables count down each time an `X` or `Y` subroutine is executed (potentially each time through the main loop), allowing a subroutine to be skipped until the variable reaches zero. If the subroutine is skipped 256 times, then a change to `XVal` or `YVal` can happen only every 128 ms (256 × 500 µs), which is the automatic repeat rate.

In this simplest form, 256 increments would take 33 s (256 × 128 ms). So, even though the time for a single increment seems short (128 ms), waiting for a maximum change takes a long time (33 s). By substituting different algorithms in the `Calculate New XInit` in the `X INC/DEC` subroutines, you can affect the number loaded into the `XSkip` variable (reloaded from `XInit`). The simplest algorithm is to decrement `XInit` each time. This would reduce the automatic repeat rate from 128 ms by 500 µs for each change in `Xval` in automatic Repeat mode. For a maximum change, this cuts the total time in half to approximately 16 s.

You can see that a linear change might not be the best approach. If `XInit` is reduced by 12.5% of itself each time `XInit` is divided by eight (three rotations), the total time is reduced to less than 2 s. This has a side effect of limiting the minimum step time because 12.5% of anything less than eight equals zero, and `XInit` is no longer reduced. With this algorithm, the step time is reduced for the first 32 taps in 1 s, while the remaining 223 taps are stepped at 4 ms each over the next 1 s.

HIGHER RESOLUTION

Most digital potentiometer manufacturers don't produce devices with more than 256 taps. If you are using a digital potentiometer's output as a voltage divider rather than its resistive properties, then you can substitute a serial DAC for it.

Serial DACs such as the MCP4922 have a voltage output. They are available in single and double devices similar to digital potentiometers. In fact, the same SPI routine can be used as the one communicating with the digital potentiometer. Both formats require 16 bits of data to be sent to control a single device. Packages with dual devices require two 16-bit writes.

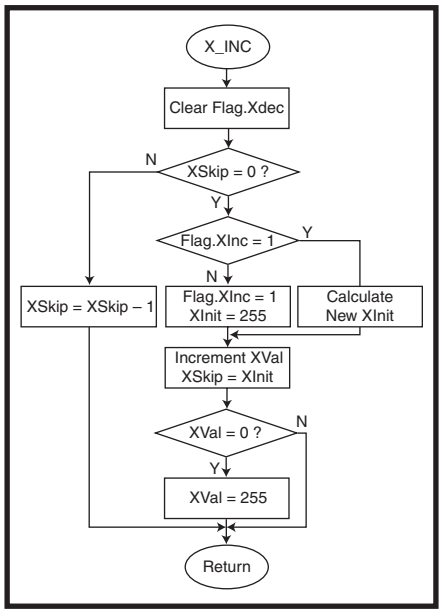


Figure 4—*This routine is one of four used to adjust the `XVal` and `YVal` variables. The rate that `XVal` is incremented depends on the `XInit` variable. A simple algorithm (XInit = XInit – 1) will change the rate in a linear fashion. Tune for the best picture according to the application.*

MCP4922 Dual 12-bit DAC

Bit 15															Bit 0
*A/B	BUF	*GA	*SHDN	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

BUF	
0	No buffer used
1	Buffer used

*SHDN Input control	
0	Output buffer disable
1	Power down

*A/B	
0	Write to DACA
1	Write to DACB

*GA	
0	VOUT = 2 × VREF × D/4096
1	VOUT = VREF × D/4096

MCP42*xxx* Dual 256-tap digital potentiometer

Bit 15															Bit 0
X	X	C1	C0	X	X	P1	P0	D7	D6	D5	D4	D3	D2	D1	D0

C1	C0	
0	0	Do nothing
0	1	Write data
1	0	Shut down
1	1	Do nothing

P1	P0	
0	0	Do nothing
0	1	POTA
1	0	POTB
1	1	POTA and POTB

Figure 5—*Although both devices use a 16-bit SPI format, the commands within the data format are different. Of the 16 bits, the DAC has 12 bits of data while the digital potentiometer uses only 8 data bits.*

Figure 5 shows the communication differences between the digital potentiometer and the DAC. The DAC has independent reference inputs meant for 4.096- or 2.048-V references. A 2× gain is available through the *GA configuration bit. This will allow a 2.049-V reference to actually produce a 0- to 4.096-V output. It provides an easy conversion between the data byte value and voltage output because 1 bit equals 1 mV.

MINIMUM I/O

Now that 8-pin microcontrollers are no longer the laughing stock of the industry, Microchip has reduced the pin count again. Are three I/Os and one input useful? Isn't this getting a bit whacky?

I designed the next circuit using a PIC10F20*x* series microcontroller that comes in an SOT-6 package (see Photo 1). With two output pins defined as SCK and SDA to control a digital potentiometer or DAC, there was little left for monitoring the JS1100AQ switches. I connected JS1100AQ's independent switch to the microcontroller's input. I then needed to monitor the other four directional switches with the single pin that was left. Not an easy job for a digital input.

I decided to monitor the capacitor of an RC circuit with the last I/O pin. The direction switches (JS1100AQ) connect different resistor values between 5 V and the capacitor, which has its other lead grounded. By using resistor values that are half the previous value, I could get eight different RC time constants charging the capacitor by using one switch or combinations of two switches. As an input the microcontroller could count how long the digital input took to go from a logic 0 to a logic 1. This was dependent on the V_{IH} of the port input, but it was consistent and dependent on V_{CC}. By configuring the same port pin as an output (outputting a logic low), the capacitor could be discharged.

The main loop of execution program monitors the JS1100AQ's independent switch for closure. This reconfigures the RC monitor pin to an input and starts a software loop (12 µs) monitoring the input pin and incrementing a 16-bit counter variable. Depending on the tilt direction of the JS1100AQ, one or more of the corner switches will be closed. Each combination of switch closures creates a different current flowing into the capacitor. Thus, the capacitor will charge at eight different rates. Execution breaks out of the counter loop after the monitor input is read as a logic 1. The monitor pin is reconfigured to an output (discharging the capacitor). The timer's count is compared to a table of values. The closest selection from the table returns the position of the JS1100AQ (N, NE, E, SE, S, SW, W, NW). Branching to appropriate routines can adjust the `XVal` and `YVal` variables used to update the SPI device.

Just when I thought I had it licked, I remembered a little note that explained how SPI values would be internally transferred after the raising of the device's CS input. Whoa! I wasn't using the CS and had it tied low (enable). Did I not use all of the I/Os? No, I used all four. What to do?

I was able to tie the monitor pin to the SPI device's CS input (see Figure 6). Timing the capacitor and sending SPI commands were independent operations, so the monitoring pin could do triple duty by monitoring the voltage input, shorting the capacitor to ground, and providing a CS signal.

I chose an RC combination that would be approximately 100 ms. (The value limits the maximum step speed.) The three other values became one-half, one-quarter, and one-eighth of the first value. I needed to find values for the LUT for each combination of switch

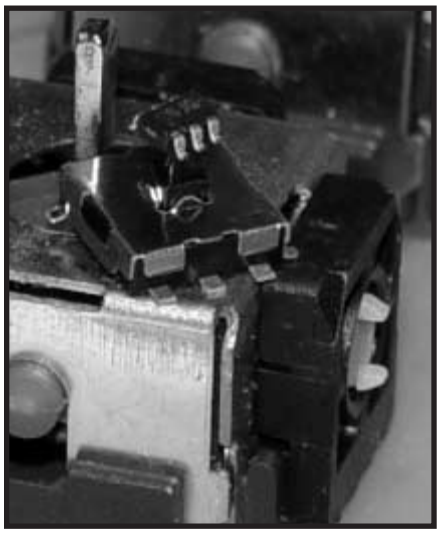


Photo 1—*When looking at this analog joystick, notice the size difference between it and the JS1100AQ multiposition switch balanced on top of it. Look closely at the JS1100AQ, and you'll also see the SOT-6 PIC10F20x.*

positions. By altering the program to output a pulse on the SCK pin based on the monitor input, I could measure the pulse width in microseconds based on what the circuit actually saw at the monitor input. I placed the measured values in the LUT to let the program determine which switches the JS1100AQ was closing. If I'd had 1% resistor values available, I might have been able to choose table values based on a single measurement of the highest resistor or perhaps a self-calibration routine during power-up.

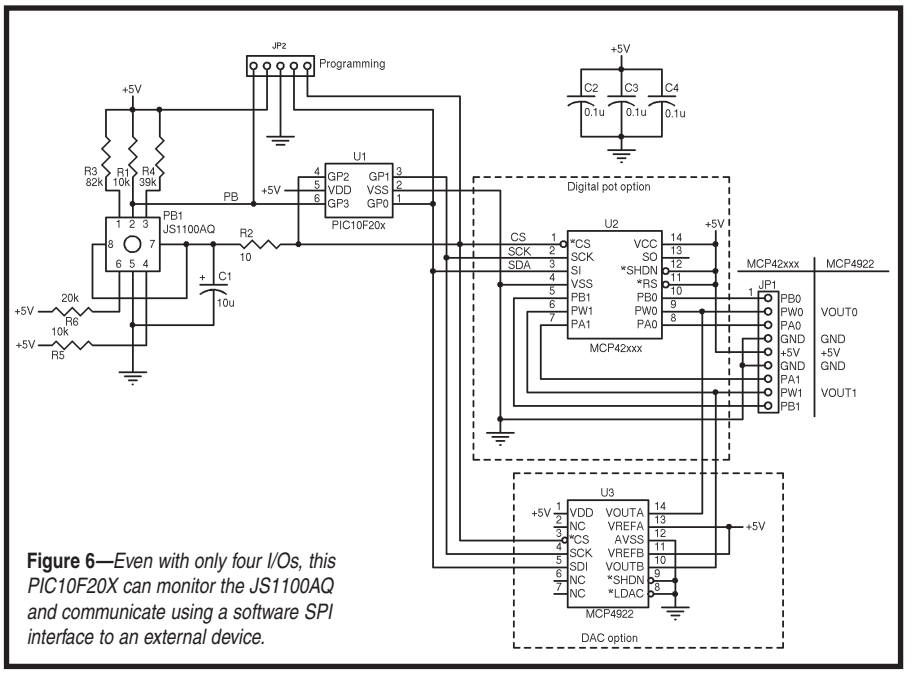


Figure 6—*Even with only four I/Os, this PIC10F20X can monitor the JS1100AQ and communicate using a software SPI interface to an external device.*

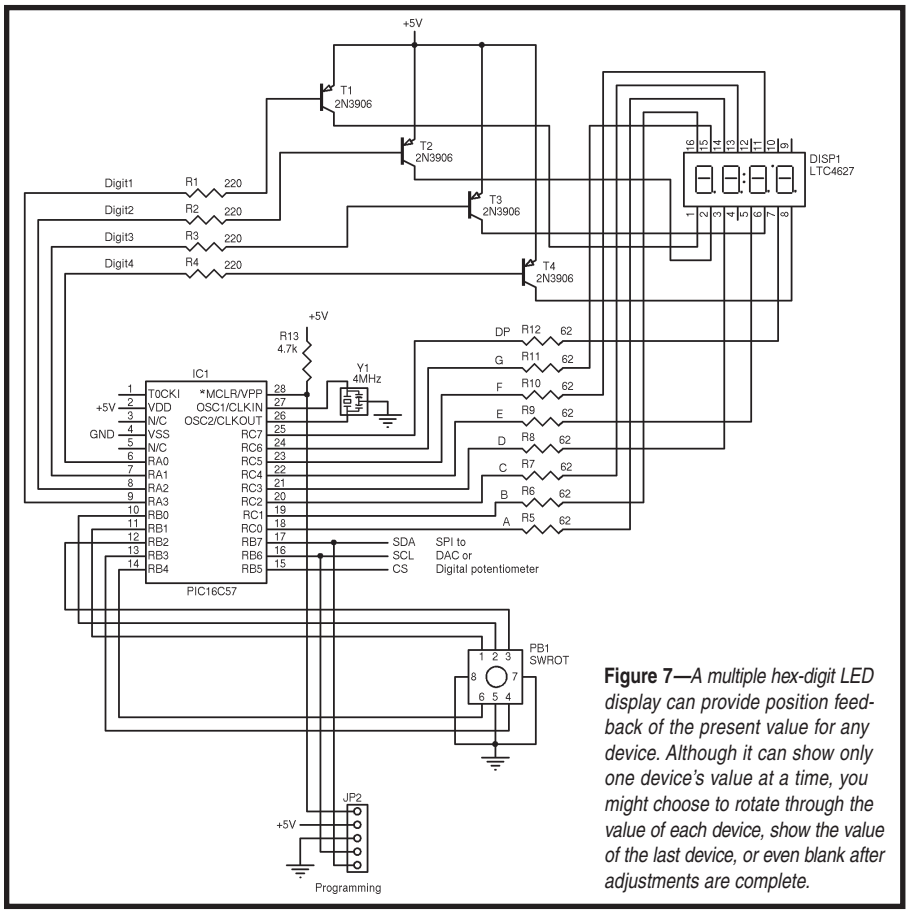


Figure 7—*A multiple hex-digit LED display can provide position feedback of the present value for any device. Although it can show only one device's value at a time, you might choose to rotate through the value of each device, show the value of the last device, or even blank after adjustments are complete.*

VISUAL FEEDBACK

I have to admit that relying on an application to provide position feedback isn't the user-friendliest option. So what kind of options might be useful with your interface?

The obvious choice is to hang an LED/LCD panel meter on the output of each DAC. But this is an expensive solution and doesn't work well with the digital potentiometer solutions. A reasonable alternative would be a multi-digit LED display (an LTC4627 costs $2.40 from Digi-Key).

Substituting a microcontroller with more I/Os isn't a big deal either. For the most part, the peripherals can boost microcontroller costs, not just the number of I/Os. The PIC16F505 costs $1.75 versus the PIC16F57, which costs $2.73.

With a circuit similar to the one in Figure 7, the position (`xval`) can be displayed while an adjustment is being made. The display could alternate display values for multiple axes. Center off values can be presented by displaying –127 to 128.

Creativity reigns when designing user interfaces. If you've had a chance to play with an iPod, you know what I mean. It isn't what you have to work with, but how you use it.

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.*

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2005/176.

SOURCES

JS1100AQ Joystick navigation switch
E-Switch Inc.
www.e-switch.com

DS1869 Rheostat
Maxim
www.maxim-ic.com

MCP42xxx Dual digital potentiometer, MCP4922 DAC, PIC10F20x, PIC16F505, and PIC16F57
Microchip Technology
www.microchip.com



’51 Favorites

The ’51 has been around for more than two decades, but today’s ’51s are vastly superior to their ancestors. This month, Tom looks at where ’51 technology has been and where it’s going.

To those of you who recall my “’51 Flavors” column, I apologize that this column’s title is so derivative (*Circuit Cellar*, 163, February 2004). The ’51 has been around for 25 years or so, so it’s a little hard to come up with something original. But that’s the point after all. It’s the fact that the ’51 market is crowded with derivatives from dozens of suppliers big and small that sets it apart from all the rest. It’s ironic, given that Intel isn’t known for being loose with their intellectual property, that the ’51 has morphed into a virtual peoples’ micro.

It’s the fact that the new versions, like the title of this column, aren’t too original that keeps the bandwagon going. The capabilities of the modern ’51s are vastly superior, but successful derivatives are the ones that don’t stray too far from the essence of the original. Volkswagen New Beetle, *sí*. New Coke, *no*.

Is it my fault the wizards come up with new ’51 chips faster than I can come up with clever titles? Better get used to it because the ’51 is here to stay.

SWISS ARMY SoC

Add STMicroelectronics to the already long list of ’51 suppliers. Yes, they’re a bit late to the party, but you have to admit their uPSD33*xx* makes quite an entrance. It’s arguably one of the highest performance and integration ’51s on the street (see Figure 1). With a fast processor (10 MIPS peak at 40 MHz), up to 288 KB of flash memory, 32 KB of SRAM, and plenty of I/O, it’s a big-ticket System-on-a-Chip with a single-digit price tag ($4 to $8 at 10,000) and no NRE.

STMicroelectronics arrived in the ’51 market via an admittedly circular route with their acquisition back in September 2000 of an outfit called Waferscale Integration and its programmable system device (PSD) chip. I wrote the PSD up years ago in an article reflecting the chip’s multifaceted (memory, I/O, and programmable logic) support role (“Swiss Army Chip,” *Circuit Cellar*, 120, July 2000).

The ’33*xx* concept is simple enough. Just take yesteryear’s typical two-chip ’51 MCU and PSD setup and shrink it to a single chip. Actually, under the hood, it’s both a little more complicated and literally as simple as that. One huge upgrade, and virtually mandatory at this late stage of the game, is that the STMicroelectronics ’51 core is turbocharged for performance far beyond the 1 MIPS or so of the original.

To refresh your memory, an original 1970s-era ’51 would execute most instructions in one or two machine cycles, with a few (e.g., MUL) taking up to four machine cycles. That sounds good, except for the fact that a machine cycle required 12—yes, count ’em, 12—clock cycles. Ouch.

So many clocks per machine cycle is architectural low-hanging fruit that the ’33*xx* harvests by cutting the number of clock cycles per machine to four, expediently boosting performance to three times the original at a given clock rate. Note that while the number of clocks is reduced, the number of machine cycles required to exe-

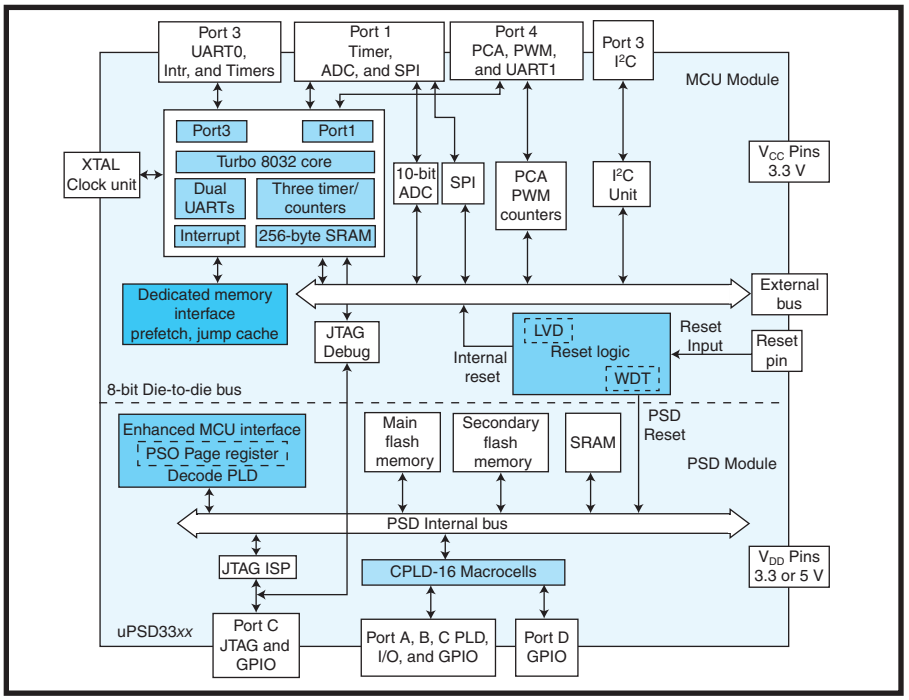


Figure 1—*It’s two, two, two chips in one. STMicroelectronics pushes the integration envelope by packing a hopped-up ’51 core and a PSD chip in a single package.*

picc enterprise edition



300+ supported chips
3 reliable compilers
1 simple solution

HI-TECH Software now brings you three award-winning compilers in one with the new PICC Enterprise Edition.

upgrade offers available

HI-TECH SOFTWARE

HI-TECH Software's **PICC Enterprise Edition** is the only compiler package available with comprehensive support for all 10/12/14/16/17/18 series PICs and also the new dsPIC processors. PICC Enterprise Edition is the complete package that unites our successful PICC, PICC-18 and dsPICC compilers.

With the choice of over 300 different PIC processors, deciding which one to use is a challenge. Even when you do decide, changing project requirements during development often call for a larger or more sophisticated chip. A change in processor can also mean buying a new compiler...until now!

With HI-TECH Software's PICC Enterprise Edition, there is finally a single product that supports all of Microchip's PICmicro processors. Not only does this simplify which compiler to choose, it also eases migration between processors by having a familiar tool-set.

PICC

Our most popular genuine ANSI C compiler, PICC is a high-performance C compiler for the Microchip PICmicro 10/12/14/16/17 series of microcontrollers. PICC supports all standard data types including 24 and 32 bit IEEE standard floating point.

This industrial-strength compiler makes full use of specific PIC features and using an intelligent optimizer, can generate high-quality code easily rivaling hand-written assembler.

HI-TECH Software's PICC compiler can be fully integrated with MPLAB, or can be used directly from a Makefile or command line.

PICC-18

HI-TECH Software's PICC-18 compiler is a proven C compiler for the Microchip PICmicro PIC18 family of microcontrollers. Delivering reliable efficient code that's as good as or better than hand-written assembler code.

Tightly tuned to the PIC18 architecture, PICC-18 allows firmware development in a fraction of the time required for conventional assembly language.

HI-TECH Software's PICC-18 implements full ISO/ANSI C (with the exception of recursion) and integrates with the HI-TIDE IDE or with MPLAB.

ds PICC

HI-TECH Software's dsPICC is the outstanding choice for developers on Microchip's newest family of PICmicro devices.

Built with the same technology as our highly successful PICC and PICC-18 compilers, dsPICC promises the same reliability and code efficiency to which you have become accustomed.

Full ISO/ANSI C is supported, including recursive function calls.

HI-TECH Software's dsPICC integrates seamlessly with MPLAB and HI-TIDE, or can be run directly from the command line.

Recognized Microchip Third-Party Tool Provider

HI-TECH Software proudly supports the Microchip brand with industrial-strength software development tools and C compilers

HI-TECH Software LLC 6600 Silacci Way Gilroy, CA 95020 USA
Ph: 800 735 5715 **Web:** http://www.htsoft.com/

cute a particular instruction remains unchanged, easing migration of timing-dependent code.

The good news is that nearly half the ′51 opcodes are just 1 byte long and execute in a single machine cycle. The bad news is that a stock ′51 fetches 2 bytes from program memory every machine cycle, even if the instruction is 1 byte (i.e., the second byte is ignored). Surely, there's a better way to make use of the available bus bandwidth than these dumb dummy fetches.

STMicroelectronics does just that, taking advantage of the freed up bus to further boost performance with a prefetch queue that gathers up to 6 bytes of opcodes ahead of time. The queue helps insulate the processor core from the latencies induced by slow (i.e., multimachine cycle) instructions and also the wait states required by the flash memory at higher clock rates.

Of course, the problem with prefetching is the dreaded, but oh so frequent, branch. It's easy to imagine real-world scenarios (i.e., tight nested loops) that bring a prefetch scheme to its knees, with branches flushing the queue so often that you end up back in a de facto dummy fetch situation.

Enter the second upgrade, a branch cache with four entries, each of which stores 6 bytes worth of instructions. If a branch hits in the cache (i.e., branch target address matches one of the four cache tags), the 6 bytes of instructions in the cache are broadsided into the prefetch queue (see Figure 2).

Both the prefetch queue and branch cache can cloud determinism, possibly introducing a bit of timing uncertainty. In other words, a chunk of code may take slightly more or fewer clock cycles to execute, depending on the particular state of the queue and cache at a given point in time. The resulting jitter may not always cause a problem,

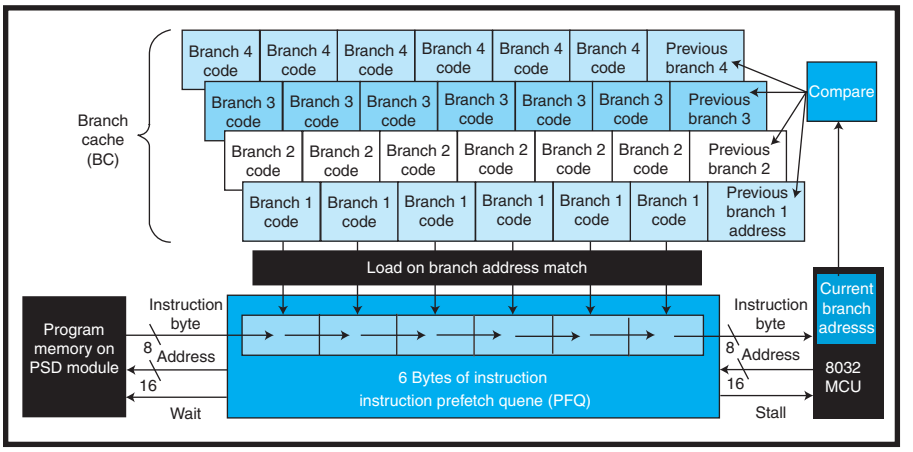
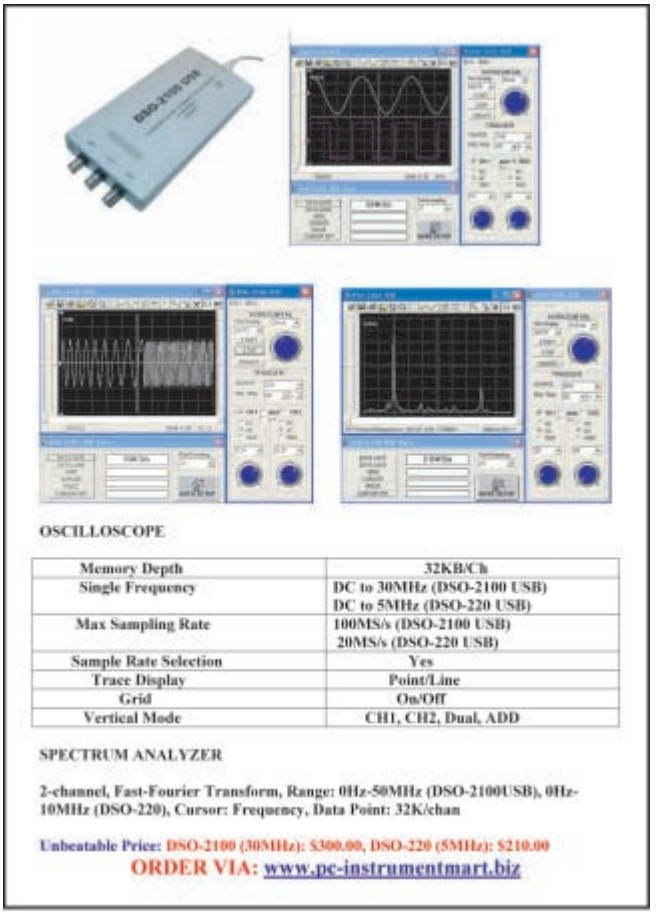


Figure 2—*One key to the '51s longevity is the fact that virtually all modern versions boost performance over the original. For the STMicroelectronics uPSD33xx, upgrades include a six-entry prefetch queue, and a four-entry branch target cache that help cut the number of clocks per machine cycle.*





but if it does, it can be a real showstopper. Fortunately, the '33*xx* allows you to dynamically enable or disable the prefetch queue and branch cache under software control.

Device type	V_{CC} for MCU module	V_{DD} for PSD module	Ports 1, 3, and 4 MCU module	Ports A, B, C, and D on PSD module
5 V: uPSD33XX	3.3 V	5.0 V	3.3 V (5-V tolerant)	5 V
3.3 V:uPSD33XXV	3.3 V	3.3 V	3.3 V (5-V tolerant)	3.3 V (not 5-V tolerant)

Table 1—*The headlines for the stacked die approach are all about integration. But don't overlook the benefits of being able to mix and match multiple voltage devices in a single package.*

Other than the speed-up, the '33*xx* core includes the standard '51 peripherals (i.e., two UARTs and three timer/counters) plus a nice complement of add-ons, including an eight-channel, 10-bit ADC, six-channel programmable counter array with capture/compare and PWM, SPI, and I²C.

Turning to the PSD side of the part, you find the memory, which is comprised of a main block of flash memory, a secondary block of flash memory, and SRAM. Thanks to built-in programmable decode logic, the various memories can be flexibly allocated within the MCU's code and data spaces. For example, the main flash memory block, which is typically allocated as program space, can be temporarily switched to data space for the purpose of in-system programming (ISP).

In addition, the PSD includes extra I/O lines (3 × 8 bits), which are fronted by a small (16 macrocell) PLD or can be accessed directly by software as general-purpose parallel I/O. The PLD is a nice touch for handling simple glue and interface logic chores in hardware at high speed, possibly eliminating the need to add an extra chip. Finally, a JTAG port provides both direct access to the flash memory for programming and serves as the debug channel for the MCU core.

I said earlier the '33*xx* was a conceptually simple combination of an MCU and PSD. It turns out the concept has been carried into practice. Under the hood, the '33*xx* is actually composed of stacked MCU and PSD die (i.e., it's truly two separate chips in one package).

The stacked die concept is a pretty big deal that's really taking off. I must confess that I wondered early on about the reliability and robustness of the scheme (e.g., thermal and mechanical shock), but so far I haven't heard of any problems.

On the other hand, there are clear advantages. Notably, it allows STMicroelectronics to offer a dozen different variations by mixing and matching relatively few die with two package and two voltage options. That's much less costly at the front end and far more streamlined in manufacturing than actually tooling up and running with a dozen different wafers. Presumably, STMicroelectronics passes along some of the savings to you.

Besides the price tag, the stacked die approach delivers a unique, and possibly pivotal, benefit in terms of I/O flexibility. Even 8-bit parts these days are migrating to advanced processes that run at lower voltages.





Something "fishy" is going on in Test and Measurement



USBInstruments™



STINGRAY Dual Channel Oscilloscope, Signal Generator & Multi-Function Instrument

STINGRAY is the value-for-money dual-channel PC oscilloscope with signal generator, data logger, spectrum analyzer, voltmeter and frequency meter capabilities. Despite its low cost, **STINGRAY** offers a wealth of features including 1M S/s sampling with 12-bit conversion, advanced digital trigger modes, AC/DC coupling and a built-in signal generator with 10 bit resolution.

STINGRAY
only $220



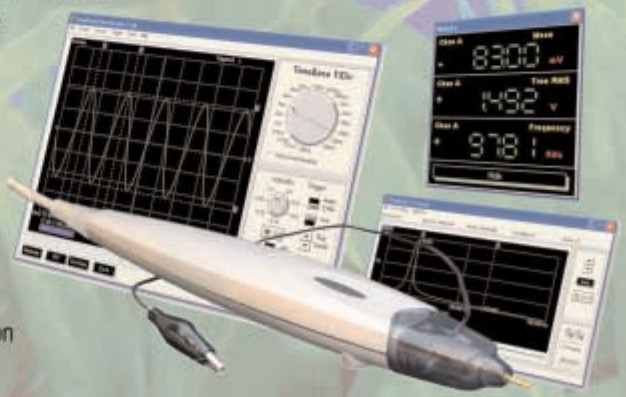
STINGRAY Features

- Dual-channel standard BNC input connectors
- 12-bit simultaneous ADC sampling on both channels
- 20M S/s sampling rate (repetitive) 1M S/s native
- Signal Generator Output/External Trigger Input
- Maximum input voltage +/- 50V
- AC/DC Coupling
- Edge, min/max pulse width and delayed trigger modes
- Analog Bandwidth 200KHz
- Self-Powered USB Interface - no external PSU required
- 3rd Party application software support provided
- Hardware upgradeable over USB
- Dimensions: W: 116mm x H: 30mm x D: 100mm
- Weight: 5oz without cable

SWORDFISH Hand Held USB Oscilloscope

A unique hand-held device that combines the functions of an oscilloscope, data logger, spectrum analyzer, voltmeter and frequency meter in a single instrument. **SWORDFISH** features a user-replaceable precision spring-loaded probe tip which can be used to probe even small SMD components. The probe cap can be removed to allow **SWORDFISH** to connect to standard oscilloscope probes or BNC cables if required. **SWORDFISH** comes complete with oscilloscope and data logging software. Supplied Windows DLL's allow 3rd party applications use. Sample code in several popular programming languages are provided. Windows CE and Linux drivers are also available upon request.

SWORDFISH
only $275



SWORDFISH Features

- 10 Bit ADC Resolution
- 1G S/s sampling rate (repetitive) 40M S/s native
- Maximum input voltage +/- 50V
- AC/DC Coupling
- Edge, min/max pulse width and delayed trigger modes
- Analog Bandwidth 5MHz
- Self Powered USB Interface - no external PSU required
- Precision spring-loaded probe tip or standard BNC connection
- 3rd Party application software support provided
- Hardware upgradeable over USB
- Dimensions: W: 34mm x H: 24mm x D: 240mm excluding USB cable
- Weight: 5oz with cable

Detailed specifications, information and downloads at www.usb-instruments.com

Saelig Company Inc

1160-D2 Pittsford-Victor Road Pittsford NY 14534 USA T: 585-385-1750 F: 585-385-1768

info@saelig.com www.saelig.com/ad/cctestfish.htm

NEW!

The bad news is that a lot of embedded applications still call for 5-V I/O to interface with standard I/O devices and sensors.

To serve 5-V or mixed 3.3-V/5-V applications, STMicro exploits the stacked die scheme by offering versions of the '33*xx* that combine the 3.3-V MCU core with either a 3.3- or 5-V PSD chip (see Table 1, page 81). Thus, a mixed-voltage system can use the MCU I/O ports for 3.3-V I/O and rely on a 5-V PSD for true 5-V I/O. Yes, you have to provide both voltages (maybe they'll throw in a regulator someday), but you'd have to do that in any mixed-voltage design anyway. At least the ST approach enables true multilevel I/O capability in a single package without the need to add external level-translator.

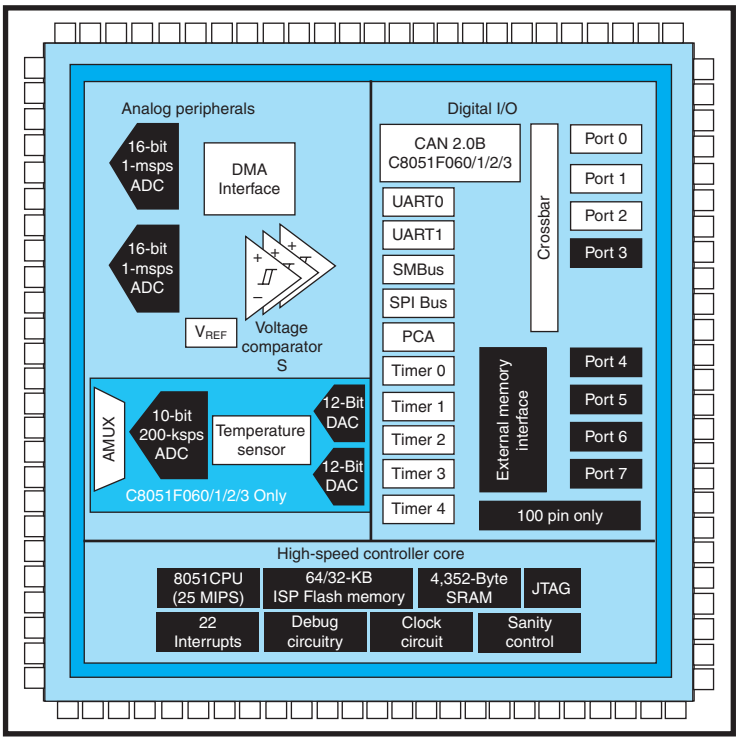


Figure 3—*The Silicon Labs C8051F06x is all about analog with a capital "A."*

SIGNAL CENTRIC

As the price for ones and zeros (i.e., digital) heads toward the latter, attention is turning to analog as the differentiating factor. Silicon Labs's (formerly Cygnal) C8051F06*x* lineup is a good example of the new mixed-signal breed (see Figure 3).

For sure, the 'F06*x* has all the digital bases covered, starting with the mandatory processor core makeover. By virtue of a rather aggressive pipelined implementation that executes most instructions in just one or two clocks, the 'F06*x* delivers 25 peak MIPS (at 25 MHz), which should help finally lay to rest the pokey legacy of 12-clock '51s.

Other upgrades include a 22-source two-level priority interrupt controller and a novel crossbar switch that allows the arbitrary mapping of I/O functions to pins. The crossbar is a nice touch that goes a long way toward

Parts List Software
for Engineers and Designers
■ Easily create and manage multi-level parts lists for products in development...and after.
■ Track sources for items with multiple price breaks.
■ Calculate product costs at any quantity.
■ Launch CAD, viewer or browser from any item.
■ Automatically generate RFQs or POs.
New Version 5.0
■ New Report Layout Editor customizes reports/labels.
■ New Connection to QuickBooks 2002/2003 Pro simplifies accounting (US version only).
■ New Multi-currency for foreign suppliers eases exchange rate calculations.
Parts & Vendors™
For Windows 98/NT/Me/2K/XP 3 Editions, starting at $99 + s/h
Visit www.trilogydesign.com and download our FREE DEMO.
Trilogy DESIGN™
Or, Call 800-280-5176
530-273-1985 Fax 530-477-9106
P.O. Box 2270, Grass Valley, CA 95945

Micro-Server
Internet Appliance Engine
SOM-5282EM
• Coldfire 66/80 Mhz CPU
• 10/100BaseT Fast Ethernet
• 16 GP I/Os & 8 Channel A/D
• Programmable in Java™ or C
• Telnet, FTP, and HTTP Servers
• 3 Serial Ports, CAN 2.0B & SPI
• uClinux with Real Time support
• 4.5 MB Flash & up to 16 MB RAM
• Typical Power Consumption < 2 Watts
• Real Time Clock & Nonvolatile Memory
• Carrier/Socket Board & Power Supply Available
• Small, 144 pin SoDIMM form factor (2.66" x 1.5")
The SoM-5282EM is a System on a Module, based on the Freescale MCF5282 Processor. This 32-Bit processor runs uClinux making it extremely easy to create smart Network/Internet capable devices, with Data Acquisition and Control properties. If Real-Time processing is required we can optionally provide RTAI Real-Time extensions. Write sophisticated network applications in days instead of months using standard GNU tools. Single Unit pricing starts at $150. Optional Carrier/Socket board, Enclosure, and Power Supply also available. For additional information visit, www.emacinc.com/som/.
Since 1985 OVER 20 YEARS OF SINGLE BOARD SOLUTIONS
EMAC, inc.
EQUIPMENT MONITOR AND CONTROL
Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: www.emacinc.com



Photo 1—*Dude, where's my serial port? Featuring two USB ports, the Silicon Laboratories C8051F06x evaluation board brings the '51 into the modern age of development tools.*

eliminating the conflict that sometimes occurs with fixed-pinout chips when both of the functions you need happen to be assigned to the same pin.

It's no surprise to find a healthy complement of memory (32 or 64 KB of flash memory and 4 KB of SRAM) and all the usual digital interfaces, including parallel I/O, UARTs, SMBus (I^2C-compatible), SPI, plenty of counter/timers, and even a controller area network (CAN 2.0B) port on some models. A built-in oscillator calibrated at the factory is precise enough (24.5 MHz ±2%) to eliminate the need for an external crystal in many applications.

But what really sets the 'F06*x* apart is a full quiver of analog arrows, starting with a pair of fast (1 million samples per second) and precise (less than 1 LSB nonlinearity, no missing codes) high-resolution 16-bit ADCs. The converters can be configured as two single-ended inputs or paired for differential operation. When operated as a pair of single-ended converters, true simultaneous conversion is possible to support applications (e.g., motor control and power factor correction) that require measuring two inputs (e.g., voltage and current) at exactly the same time.

Even the fast 'F06*x* processor core might have a hard time keeping up with the speedy ADCs in software. Thus, the 'F06*x* includes a built-in DMA controller that automatically handles the transfer between the converters and internal or external RAM. Conversions can be triggered in software, continuously triggered by a timer, or externally triggered via the conversion start (CNVSTR) pin. Impressive, but that's just the start. The list of analog goodies goes on with three comparators, a precision voltage reference, and a built-in brownout detector. Put all the pieces together and there are seemingly few analog challenges the 'F06*x* isn't up to. But in case you happen to come across one, just move up to the 'F060, 'F061, 'F062, or 'F063 variants. To bring the signal chain full circle, these parts throw in two 12-bit voltage output DACs, an additional eight-channel (multiplexed) 10-bit ADC, and even an on-chip temperature sensor.

Shop around for all that analog stuff in separate chips, and there's no doubt the 'F06X is a steal at less than $10 at 10,000. It wasn't that long ago you'd pay that much just for the 16-bit ADC, never mind all the rest.

The '51 heritage may be old school, but the development tools are up to date. For example, note how both ports (data and debug) on the 'F064 evaluation board in Photo 1 are USB-enabled.

LITTLEST PROCESSOR THAT COULD

Last month I wrote that the time has come for 32-bit MCUs to move into the mainstream, notably riding the bandwagon of ARM-based chips. But don't take that to mean the party is over for 8-bit chips. Indeed, the MCU market forecasts generally predict that the volume for 8-bit MCUs will continue to grow. It isn't a zero sum game, and the 32-bit MCU of tomorrow will happily share ever-expanding shelf space with the '51s of today.

In the normal world of business and commerce, products have life cycles during which they are born, grow, plateau, and ultimately decline. But not so in the world of chips, at least as long as that fountain of youth known as Moore's law continues to hold sway. Smaller, faster, and cheaper is the ticket for continued growth.

There's no better example than the newest members of the LPC family from Philips (see Figure 4). When I first wrote about the then new parts in "Little Processor that Could" (*Circuit Cellar*, 112, November 1999), I speculated on just what the LPC moniker stood for: Low pin count? Low-power controller? Low-price chip? Judging from the new parts, the answer is all three.

In particular, the '9102 and '9103 stand out by virtue of their ultra-small 10-pin package at a mere 3 mm × 3 mm (see Photo 2). For those of you who are metric-challenged, that's less than 0.125″ on a side. You could fit 50 of these little puppies in the footprint of the original '51's 40-pin DIP!

With so little real estate to play with, the function of each pin becomes critical. An on-chip clock can eliminate the cost and pins required to support an external clock. Although it isn't by any means a new concept, Philips gets brownie points for factory calibrating their on-chip RC oscillator to ±1% accuracy at room temperature (25°C). Even though the clock frequency may drift as much as 5% across the full temperature range (–40° to 85°C), it can be dynamically tuned if a good reference is available. The on-chip watchdog timer also has it's own low-frequency (400-kHz nominal) clock, which can be used to drive the CPU as a power-saving option.

Another pin-saving fea-

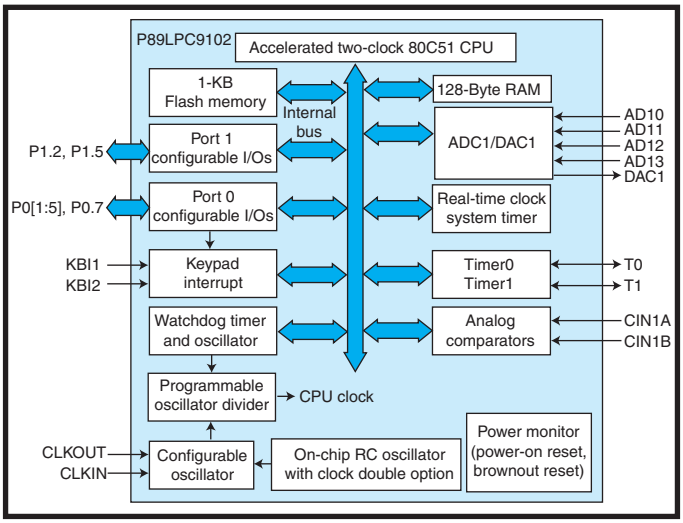


Figure 4—*Every pin counts when you've only got 10 ('9012, '9012) or 14 ('9017) to play with. Philips rises to the challenge with on-chip clocks (CPU and watchdog) and reset circuits.*

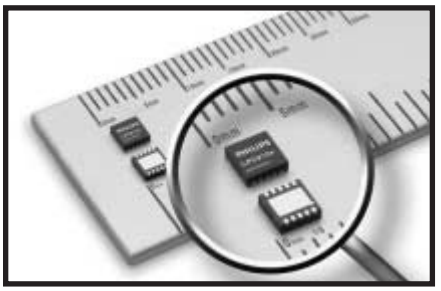


Photo 2—*It's a small world after all. And it's getting even smaller thanks to the incredible shrinking '51s from Philips.*

ture is inclusion of on-chip reset logic for reliable power-up timing and glitch suppression without any external components. The bottom line is that many LPC designs will be able to get off the drawing board with only power and ground connections required.

With only 1 KB of flash memory and 128 bytes of RAM, these new LPCs aren't the parts for software-intensive applications. On the other hand, when it comes to living up (or rather down) to low-power (less than 1-µA standby) and low-price ($0.70 at 10,000) aspirations, small is indeed beautiful.

There may not be room for much software, but the code that does fit will run fast. Like the STMicroelectronics and Silicon Labs parts, the LPC core is accelerated significantly with practically all instructions executing in two or four clocks. Using the RC oscillator (nominal 7.3728 MHz) in clock doubling mode, that translates to a minimum instruction cycle time of 136 ns, delivering nearly an 8-MIPS peak throughput.

UNTIL NEXT TIME

So, there you have it: three more '51s covering the spectrum from *tall* to *venti* in Starbucks-speak. In fact, the Starbucks analogy isn't far off the mark. After all, the '51 is a veritable franchise at this point. And the plethora of outlets means you get a big menu to choose from, everything from a plain cup of Joe to a decaf-mocha-low-fat cappuccino.

The '51, and the coffee, may be commonplace, but like all franchises, at least it's reassuring to know what you're getting into. And judging by the market stats, both caffeine and '51s are very, very addictive. What's next, '51s Anonymous support groups for the hopelessly strung out?

The truly amazing thing is that you can get a powerful MCU for just about the price of a cup of coffee from Starbucks. The only difference is that next year the '51 folks will give you more for less, just like they always have. And that's why the '51—and 8-bit MCUs in general—are here to stay.

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

SOURCES

LPC910*x* MCU
Philips Semiconductors
www.semiconductors.philips.com

C8051F06*x* MCUs
Silicon Laboratories
www.silabs.com

uPSD33*xx* MCUs
STMicroelectronics
www.st.com



IDEA BOX

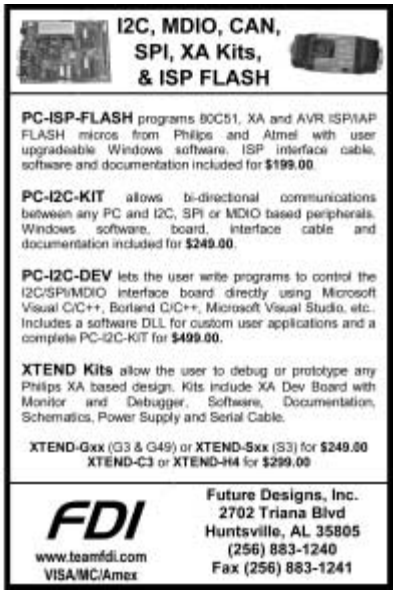
THE DIRECTORY OF PRODUCTS AND SERVICES

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″× 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 kc@circuitcellar.com. For more information call Sean Donnelly at (860) 872-3064.

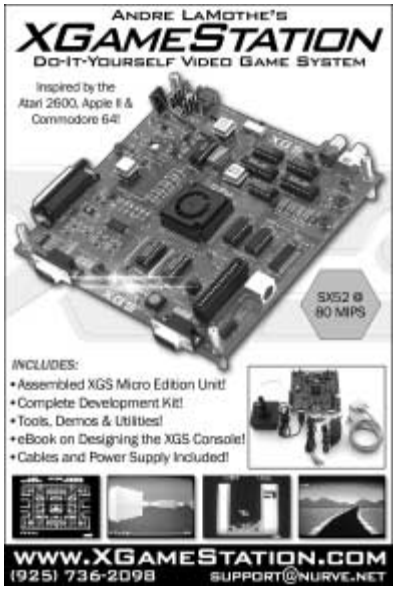
The Suppliers Directory at www.circuitcellar.com/suppliers_dir/
is your guide to a variety of engineering products and services.

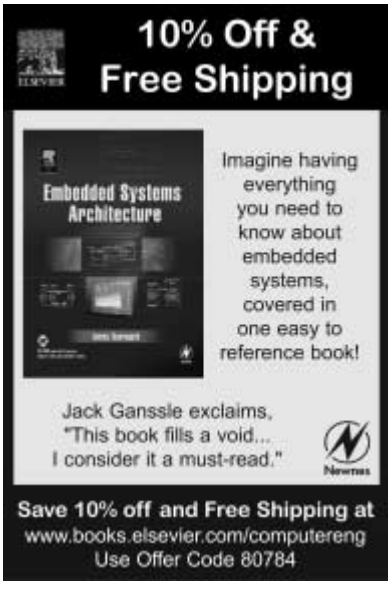






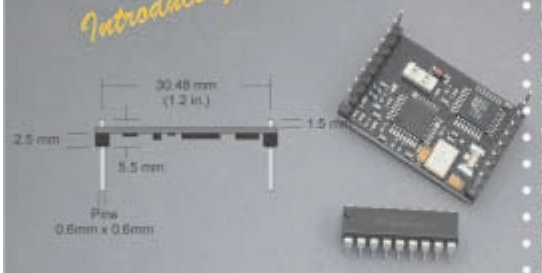






RF MODULES .....

Introducing the AT-XTR-903



- Multi-Channel Intelligent Wireless Transceiver
- 3V operation
- 9600 to 38.4Kbps
- Virtual Wire Operation
- Wafer Thin, Tiny Footprint
- Low Current
- 433, 868 & 900MHz versions
- Standard serial I/O
- Programmable TX Power
- Programmable Frequency
- User Customizable Features

ABACOM Technologies

www.abacom-tech.com

Tel: (416)236-3858 Fax: (416)236-8866

IDE, Compilers, ICD, Simulators, Programmers...

Emulators



8051 PIC® AVR®
80196 XEMICS
MSP-430 SENSORY

Phyton 718.259.3191 www.phyton.com

NEW!

Armed to the Teeth

BL2600 Wolf Single-Board Computer



$237 qty.100

BL2600 Wolf shown with optional 16MB SF1000 Serial Flash Expansion Board

Our BL2600 Wolf features include:

- Rabbit 3000® @ 44.2 MHz
- 10/100Base-T Ethernet
- 512K Flash, 512K SRAM, and 256K Data SRAM
- Dual entry IDC through-hole sockets and friction-lock connectors
- 36 digital I/O
- Eight 11-bit analog inputs
- Four 12-bit analog outputs
- 5 Serial Ports
- RabbitNet compatible for variety of I/O expansion

KIT PRICE $399* Reg. $488

Limited Time Only

Order Online!

* Special discount on purchase of BL2600 Starter Package (board + development tools). Use promotional code **9984**.

www.armedsbc.com

Z-WORLD 888.362.3387 530.757.3737

ElectronicKits.com



Usb/Serial Port PIC Programmer

Educational Electronic Kit $34.95

* Connect to computer via USB or Serial Port
* Free Software and Updates
* Windows 9x/Nt/2000 XP Compatible
* ICSP (In Circuit Serial Programming)
* Quality PCB and Silkscreen

See complete details at www.ElectronicKits.com

USB and USB Flash Pic Programmers Available

Carl's Electronics Inc. Offering hundreds of Educational Hobby Kits

Visit our website at www.ElectronicKits.com

NanoCore12™ for best price & performance: 9S12C microcontroller modules from $45!



NanoCore12™ Features:

- on-board RS232C interface
- up to 33 I/O lines, with multi-property programmability (e.g. direction, pull-up/pull-down, reduced drive, invert polarity, etc.)
- up to 8 key wake-up interrupt inputs, with digital filtering
- SCI • SPI • CAN
- 8-channel 10-bit ADCs
- multiple PWM channels
- 4-channel timers, supporting input capture/output compare, event counting, gated time accumulation, and simple PWM
- 32K multi-sector Flash • 2K RAM
- operates up to 48MHz via PLL
- 3.3 V or 5 V operation
- advanced CISC architecture
- on-chip Serial Monitor
- supports BDM debugging

1-877-963-8996 • www.technologicalarts.com

Toll-free in USA and Canada

™NanoCore12 is a trademark of Technological Arts, Inc.

EmbedDisk -40 to +85°C Extreme Temp



IDE Flash Industrial Grade Storage

MSTI Media Stream Technologies Inc.
www.embeddisk.com
info@embeddisk.com

iCOP ICOP Technology Inc.
www.icoptech.com
info@icoptech.com

SHIPPED NEXT BUSINESS DAY
FILE IN before 1PM EASTERN
$85 US
w/ solder mask
no drill size restriction
5" x 6"
2-layer - 2 pc
PROTOTYPE
Printed Circuit Board
Also available
Small Run Production
5 days turn
1pcb.com

USB Serial & Digital
USB to Serial
1, 2, 4, and 8-port models
RS-232, RS-422, and RS-485 interfaces
Data rates to 921.6K bps
USB to Digital
48/96 TTL (SSR rack compatible)
Reed and Form C relay outputs
Optically isolated inputs
Daisy chain multiple units
SEALEVEL
Sealevel Systems, Inc. 864.843.4343 TEL
sales@sealevel.com
www.sealevel.com

okwelectronics.com
RF Remote Control Systems
FM 315 MHz
'Mix-n-match' Encoders & Decoders
102 SERIES HAND-HELD ENCODERS
• Transmitter encoders for OEM systems
• Secure KEELOQ protocol
• FM 315 MHz frequency
• 1,2 or 4 push button switches
• LED transmission indicator
• Range up to 660ft
• Auto shut-off
• ABS enclosure
• 9V Battery supplied
• FCC compliant
008 SERIES FM DECODER
• FM remote Rx/decoder unit
• 'Easy learn' Tx encoder feature
• Secure KEELOQ protocol
• FM 315 MHz frequency
• Four relay outputs
• Range up to 500ft
More info. @ www.okwelectronics.com
OKW Electronics, Inc.
Tel: (800) 965 9872 Fax: (412) 220 9247
300 Old Pond Road, #203, Bridgeville, PA 15017
Email: sales@okwelectronics.com

okwelectronics.com
In-Circuit Emulator
ICEPIC 3
Next Generation Emulation!
All new universal In-circuit emulator for the PIC Microcontroller. Exceptional performance and value.
FEATURES
• Operates within MPLAB
• Hardware support for single steps in 'C'
• Fast USB connection
• Microchip assembler and 'C' compatible
• Source level debug in assembler or 'C'
• Full speed 40MHz emulation
From only $900*
• 2.8/5V Emulation
• Plug-in daughterboards ordered separately to configure system to the desired PIC
*Daughterboard not included in price.
SEE OUR WEBSITE FOR PIC COMPATABILTY!
More info. @ www.okwelectronics.com
OKW Electronics, Inc.
Tel: (800) 965 9872 Fax: (412) 220 9247
300 Old Pond Road, #203, Bridgeville, PA 15017
Email: sales@okwelectronics.com

Everything you need to build your own...
MOBILE ROBOT
SONAR UNITS
VISION SYSTEMS
MOTOR DRIVERS
OPTICS
MICROPROCESSORS
ARITIFICIAL INTELLIGENCE
WEB CONTROLS
Zagros Robotics
PO Box 460342, St. Louis, MO 63146
(314) 768-1328 · info@zagrosrobotics.com
www.zagrosrobotics.com

PROTOCESSOR
Add Industrial Automation Protocols to your Device - Fast
Modbus/TCP
EtherNet/IP
Modbus/RTU
SNMP
HTTP
Webserver
BACnet
Ethernet, RS-232 & RS-485
Metasys
Ensure connectivity using interchangeable, low cost OEM Modules with a common PCB footprint and software interface. Development kit with "C" source code available.
www.ProtoCessor.com
800-317-8319

Advertising Calendar
May Issue #178
Communications
Space Close: Mar. 10
Material Close: Mar. 17
June Issue #179
Measurement & Sensors
Space Close: Apr. 11
Material Close: Apr. 18
CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS
Contact: Sean
(860) 872-3064 Fax: (860) 871-0411

Circuit Cellar
back issues are available as
searchable archives
on
CD-ROM.
Get the ones you missed!
CD-ROM #9 2004 Issues 162-173
CD-ROM #8 2003 Issues 150-161
CD-ROM #7 2002 Issues 138-149
CD-ROM #6 2001 Issues 126-137
Order online
www.circuitcellar.com
or call (860) 875-2199



ANSI C software development tools for:
IMAGEcraft
Check it out:
Version 7 Tools
Full-featured 45-day
demos on our website!
CypressM PSoC
AVR & tinyAVR
HC08,11,12,16
TI MSP430
ARM
hardware
development kits:
Elektronikladen - Motorola
HC08, HCS12 / HC12 reference
design & starter kits
CANDIP - AVR/CAN on a chip!
Ethernut - 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!

DEVICE PROGRAMMER, USB ROM Emulator, EPROM ERASER
TopMax
ChipMax
$995
$345
TopMax : Programs over 7000 devices, 48 Pin ZIF Socket ,Tests TTL/CMOS Logic IC, Low voltage, On-board processor and built-in Universal AC, 9X/NT/2000/XP,CSA, UL, Made in USA, Lifetime free software, 2 year warranty, Free software update for life time

TM-8G - DP	8-gang (32-pin DIP) adapter for E(E)PROM	$325
TM-8G - PL	8-gang (32-pin PLCC) adapter for E(E)PROM	$495

ChipMax : Reliable and low price ,Programs over 4000 devices, EPROM, FLASH Memory, Microcontrollers, PIC, Serial EPROM, EPLD, GAL, PALCE, 40 Pin ZIF Socket, Parallel, 9X/NT/2000/XP, Made in USA ,1 year warranty, Lifetime free software, 110 or 240 AC Adapter. CM-4G: 4-gang adapter 32-DIP $115

EeRom-8U
$495
"USB Interface"
EeRom-8U : 8Mbit-45ns EPROM/Flash Emulator, High speedy USB port interface (Plug & Play)
• Fast download (8Mbit per second), Lower Voltage, 8bit, 16bit, and 32bit data split
• High Speed access time (built-in 8M SRAM 15ns)
• No external power is needed, 9x/XP/NT/2000
• Accepts all standard Hex formats and binary files
• Jumper less Emulation size configuration

LA6T-110
$115
LA6T-110 : UV ERASER for Personal user, Fast and efficient, Molded Plastic Case, Hold 9 of 28-pin or 6 of 40-pin EPROM. Timer(15-40 min).
Programming Adapters(low price) : 8-SOIC($70), 32-PLCC($70), 44-PLCC($115), 48-TSOP($135), 44-QFP($65), 52-PLCC($165), 68-PLCC($165) ,
EPROM Emulator Adapter, 32/28-pin($95) 44-pin($115) DIP-to-PLCC

EE TOOLS
E.E. Tools, Inc.
540 Weddell Drive
Sunnyvale, CA 94089
www.eetools.com
Tel. 408-734-8184
Fax. 408-734-8185

I²C™/SMBus
• Bus Monitors
• Protocol Analyzers
• Host Adapters
• Multiplexers
• Battery Applications
• Software Tools
MCC
Micro Computer Control
I²C is a trademark of Philips Corporation
www.mcc-us.com

SmartTFT™
Ultra Bright, Wide Viewing Angle, Impressive Color Images on TFT
$99 OEM Boards
• 6.5 x 4.5 inches, Program in C/C++
• Relays, Opto-couplers, Solenoid, Quad. Decoders
• 16-bit ADCs and DACs, 4-20 mA
• Supports QVGA TFT, Touch sreen
• 10 BaseT Ethernet with TCP/IP
• CompactFlash with file system
• 160 field Removable industrial screw terminals
50+ Low Cost Controllers with ADC, DAC, solenoid drivers, relays, PC-104, CompactFlash, 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 95616 USA
Tel: 530-758-0180 • Fax: 530-758-0181
www.tern.com • sales@tern.com

AVAILABLE NOW
R109
HYPERCORE II
Microprocessor Core Module
The Ideal Controller Project Development Platform - 50 MIPS SOC Microprocessor, 128K Flash, 128K SRAM, 12 bit Analog, 30+ Digital I/O. Program in Basic or C...Your project is almost done! Or, contact us for a custom design!
R2 Controls
866-797-9229 or
www.r2-controls.com

$199



Tiny, yet Linux capable

16MB fast SDRAM	10/100 Ethernet
64MHz Coldfire MCU	3 serial ports (RS232)
1 - 4 MB flash memory	1 CAN port
SD card socket (to 1GB)	LCD/KPD port
Eclipse/CDT dev. env.	serial debugger incl.

55 I/O pins & capabilities to burn...

www.steroidmicros.com

DEVELOPMENT BOARDS

NEW USB Development Board



ONLY $32.90

- AT89C5131 Microcontroller
- 32kBytes Flash Memory
- Real Time Clock
- External EEPROM
- Easy USB Interface and Control
- In-Circuit USB Programming

Also Available: Opto-Isolated Relay Output Board, Web Control Board and Heaps More

www.futurlec.com

China Connection...

PCB Production, Assembly, Turnkey

Prototype ~ Production

D/S PCB ~ 12¢ per in^2

4-L PCB ~ 20¢ per in^2

3 weeks Delivery w/solder Mask
NRE not included
UL Approved

MYLYDIA INC.

1-800-Mylydia
sales@mylydia.com

www.mylydia.com

PROBLEM SOLVED!

Custom Hardware & Software Design



- Rapid Prototypes
- Cost-effective Design
- Free Consultation and Quotes

MJS CONSULTING INC.
Mark J. Samarin M.S.E.E.
http://www.mjsconsulting.com
253-858-6844

It writes your USB code!

NO Need to be a USB expert!

HIDmaker ($399) – creates ready to compile PC & PIC programs that talk to each other over USB.

Choose your favorite languages!
PIC: Pic Basic Pro, CCS C, Hi-Tech C, MPASM. **PC:** VB6, Delphi, C++ Builder.
Single chip solution: PIC with built-in USB

HIDmaker Test Suite ($149)

USBWatch – shows your device's USB traffic, even during 'enumeration', without expensive equipment.

AnyHID – Test any USB HID device. See what data it sends, even what the data is used for.

Trace SYSTEMS, Inc.

301-262-0300
WWW.TraceSystemsInc.com

Serial, Ethernet, Web, Email
Simple, DTE, DCE, No SW Changes
$99.95 Qty 1



Grid Connect Inc.
+1 800 975-GRID
www.serial-to-ethernet.com

www.can232.com

The original (Controller Area Network) CAN to RS232 serial converter.

Powered with an Atmel® AVR® RISC microcontroller and a Philips SJA1000 CAN controller makes a unique combination of high speed transfers possible even at higher CAN bus speeds. CAN232 has also a built in receive FIFO of 32 messages which makes it possible to receive small bursts of messages without losing any frames.

Does it sound interesting?
Read more at **www.can232.com**

* Free sample programs in Delphi, VB, Linux, C & C++
* 8-15VDC supply via CAN connector (CiA DS102)
* Small size 68mm by 30mm (2.7" by 1.2")
* Timestamp in mS on CAN frames
* Operating system independent
* Both 11 & 29 bit ID support
* ASCII or Binary protocol
* Firmware upgradable
* Custom design
* CE approved



Philips is a registered trademark of Koninklijke Philips N.V.
Atmel® and AVR® are registered trademarks of Atmel Corporation

"Your Source for Design & Mfg. Services"

- ✓Schematic Capture, PCB Design
- ✓Embedded Microcontroller Design
- ✓Hardware, Firmware Design
- ✓Rapid Prototypes, Fab and Assembly
- ✓RT-OS • C • C++ • ASM • Basic
- ✓PowerPCB • OrCAD • Allegro

Reliable ♦ Fast Work ♦ Free Quotes ♦ Guarantee!

VNISOURCE DESIGN
www.vnisource.com
Tel: (469) 675-0267
Fax: (469) 675-0346

PC KEYBOARD EMULATION

Interface Keypads, Switches, or RS-232 to your PC Keyboard Input

MODEL KE24 ONLY $99.95



• Up to 12 x 12 matrix • Programmable • RS-232 Port • Macro Capability

The KE24 is the ultimate in flexibility. Inputs or serial data can emulate any of the 101 keys from a standard keyboard.

MODEL KE18 ONLY $44.95



• Up to 9 x 9 matrix • 2.5" x 3.0" Size • PC Keyboard Port • PCXT, AT Compatible

The KE18 combines a multitude of features with small size at an economical price. Custom units available.

HAGSTROM ELECTRONICS
11 Fiddlers Green
Lansing, NY 14882

Toll Free: 888-690-9080
Phone: 607-533-4441 Fax: 607-533-4443
www.hagstromelectronics.com

VISA MasterCard

ALL ELECTRONICS CORPORATION

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
Free 96 page catalog
1-800-826-5432

PLCs for OEMs

Program like a PLC (Ladder + BASIC)
Low Cost & Flexible like a Micro-p

16 to 192 Digital i/o
4 to 8 Analog i/o
10A PWMs @ 24V
LCD Port
Keypad/Expansion
Real Time Clock
Stepper pulsers
Interrupts
Pulse Measures
MODBUS Ports
RS232 + RS485



$199 Qty 1

Incredibly Easy to Program!
Our software is used by many colleges for teaching PLCs!

Get Free Ladder Logic Simulator:
www.tri-plc.com/cci.htm

Toll Free: 1-877-TRI-PLCS (1-877-874-7527)

TRi Triangle Research

Auto485 - RS232 Converter



Auto -turnaround of Tx/Rx control
Turn your RS485 into extra RS232

http://www.tri-plc.com /auto485.htm

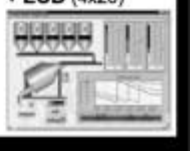


Ethernet Xserver with Virtual HMI



MODBUS Protocol
Master or Slave
- Easy SCADA/HMI

$119 Low Cost KEYPAD + LCD (4x20)



IDA Pro Disassembler





From Atmel to Zilog, hundreds of µProcessors and µControllers supported. Older Unlimited Freeware version available for 80x86 and z80. Free Evaluation Version. The PC de-facto standard.

PC Magazine 2001 Technical Excellence Awards Finalist

www.datarescue.com

QScreen™ - Low Cost Versatile Instrument Controller



$359/100s

- **Touchscreen Operated GUI**
- **128x240 CCFL- Backlit Graphic Display**
- **Programmable in C or Forth**
- **Hundreds of Screens, Buttons, & Menus**
- **A/D and Two RS232/485 ports**
- **8 Timer-Controlled Digital I/O Lines**
- **Up to 1 MB Flash, 512KB RAM**
- **Wide Selection of Plug-in I/O**

Mosaic Industries Inc.
tel: 510-790-1255 fax: 510-790-0925
www.mosaic-industries.com

ValueCAN

The High Value Tool
For Controller Area Network

- USB to CAN
- Simple software analyzer included
- DLL with examples for custom apps
- PC isolated from CAN
- 100% bandwidth at 500Kb

Only **$295**

ICS RADDAQ

www.intrepidcs.com/vcan

THE PORTLAND AREA ROBOTICS SOCIETY, PORTLAND STATE UNIVERSITY IEEE ROBOTICS AND AUTOMATION SOCIETY AND LEMON STUDIOS PRESENTS...

PDXBOT-04 ...the DVD

Filmed at one of the West Coast's largest robotics expositions and contests. Brought to you from the Great Pacific Northwest, by the pioneers who brought the explosion of Sumo Robotics competition to the USA...

WATCH EVERY EXCITING MOMENT OF THE DAY'S EVENTS PLUS...

→ Rules and insider hints for building competitive robots

→ *Backstage Interviews with Vendors, Sponsors, Competitors, Sumo Pioneers* AND MUCH MORE!

BUY YOUR COPY AT
WWW.LEMONSTUDIOS.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

JK microsystems

WWW.CONITEC.NET

$333 COMPLETE KIT

GALEP-4 UNIVERSAL PROGRAMMER

Introducing a pocket programmer with true **Universal Output**



Latest generation Galep-4 uses ASIC universal pin technology for each pin of 40 pin ZIF-socket. 5200+ device library / lifetime free updates. Programs 8/16-bit EPROM'S, EEPROM's, 0-Power 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) Providing virtually matching performance at only 1/3-1/5 the price. Info, orders, softw : 619-702-4420

ISO 9001 CERTIFIED PROFESSIONAL TOOLS 9001

info, downloads: www.conitec.net

CONITEC DATASYSTEMS - 1951 4TH AVE, SUITE 301 - SAN DIEGO, CA 92101 - TEL: 619-702-4420 FAX: 619-702-4419

PROTOTYPE CIRCUIT BOARDS

$85 US for two 5" x 6" 2-layer boards

Shipped NEXT BUSINESS DAY if data is received by 1:00 pm EASTERN

www.apcircuits.com

AP Circuits

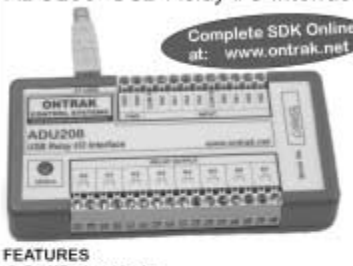
(403) 250-3406

staff@apcircuits.com

USB Data Acquisition

ADU208 -USB Relay I/O Interface

Complete SDK Online at: www.ontrak.net



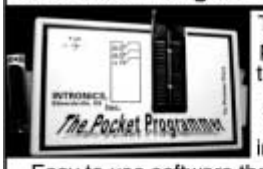
FEATURES
-8, 5-AMP relay outputs
-8, ISOLATED digital inputs
-Port Powered, Aux 5VDC output $189.00 QTY 1

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

The Pocket Programmer $149.95



The portable programmer that uses the printer port instead of a internal card.

Easy to use software that programs E(E)prom, Flash & Dallas Ram. 25/27/28 & 29 series from 16K to 8 Megabit. Adapters available for 874X, 875X MCU's, Pic, Serial Eprom's, 40-Pin X 16, Proms and More.

Same Name, Address & Phone # for 19 Years... Isn't it Amazing ?

Intronics, Inc.
Box 13723 / 612 Newton St.
Edwardsville, KS 66113 Add $7.00 COD
Tel. (913) 422-2094 Add $6.00 Shipping

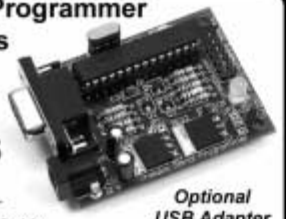
WWW.IN-KS.COM Visa / MC/Amex/Disc

Order online at: www.melabs.com

microEngineering Labs, Inc.
Development Tools for PIC Microcontrollers

Phone: (719) 520-5323
Fax: (719) 520-1867
Box 60039
Colorado Springs, CO 80960

Serial Programmer for PICs



$119.95
Includes:
Programmer
9-pin Serial Cable
AC Power Adapter
ZIF Adapter for 8 to 40-pin DIP
Software for Windows 98/Me/NT/2K/XP

Optional USB Adapter $39.95



EPIC Parallel Port Programmer starting at $59.95

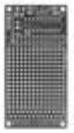
LAB-X Experimenter Boards



Pre-Assembled Board
Available for 8, 14, 18, 28, and 40-pin PIC MCUs
2-line, 20-char LCD Module
9-pin Serial Port
Sample Programs
Full Schematic Diagram

Pricing from $69.95 to $349.95

PICProto Prototyping Boards



Double-Sided with Plate-Thru Holes
Circuitry for Power Supply and Clock
Large Prototype Area
Boards Available for most PIC MCUs
Documentation and Schematic

Pricing from $8.95 to $19.95

BASIC Compilers for PIC MCUs



Easy-To-Use BASIC Commands
Windows 9x/Me/2K/XP Interface

PicBasic Compiler $99.95
BASIC Stamp 1 Compatible
Supports most 14-bit Core PICs
Built-In Serial Comm Commands

PicBasic Pro Compiler $249.95
Supports all PICmicro MCUs
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

Front Panels?

Unrivaled in price and quality for small orders

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

NEW

RS232 to TCP/IP

- *TCP/com*™ v2.0, RS232 to TCP/IP software. Plus TCP/IP to RS232.
- *WinWedge*™ RS232 or TCP/IP data direct into any Windows app. - Excel, Access, etc.

TALtech

Free 30 day evals at
www.taltech.com

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

PRINTED CIRCUIT BOARDS

QUALITY PRODUCT
FAST DELIVERY
COMPETITIVE PRICING

PROTOTYPE through PRODUCTION

- UL approved
- Single & Double sided
- SMOBC
- LPI mask
- Through hole or SMT
- Nickel & Gold Plating
- Routing or scoring
- Electrical Testing
- Artwork or CAD data
- Fast Quotes
- Flex Circuits

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 • Modem 847.233.0014 • www.pulsar-inc.com • yogii@flash.net

Weeder Technologies

www.weedtech.com DATA SHEETS

RS-232 Stackable

1710-B Brighton Cove, Ft Walton Beach, FL 32547 850-863-5723



Digital I/O • Analog Input • Analog Output • Solid State Relays • Stepper Motor Controllers • Pulse Counters • Pulse Timers • Frequency Measurement • RPM Measurement • Duty Cycle Measurement • Period Measurement • Projectile Velocity Measurement • Third-Party Peripheral Interface • Data Logging

Plug end-to-end
Stack 32 modules on the same RS-232 cable.

Download Windows software

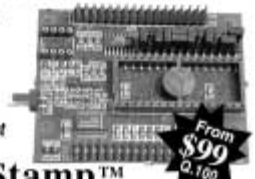
Data Acquisition & Control

...simple as a laptop.

NOW YOU CAN HAVE IT ALL!

- *Easy to Use*
- *Low Cost*
- *Low Power*
- *Tiny Size*
- *High Speed*
- *DOS Environment*

with the **DOS Stamp™**



From $99 Q.100

Easy Software Development: Use your C/C++ or Basic compiler to produce DOS EXE. Download EXE to flash disk via serial port. EXE runs on power up.

Standard Features: BIOS & DOS-ROM, 128K flash disk, 512K SRAM, 40 MHz CPU, 16 digital I/O (opto rack interface), 2 RS-232, 2 timer/counters, simple bus interface, real-time clock

Options: 8-ch 12-bit ADC, flash disks up to 288 MB, 1 RS-485, TCP/IP

Tiny Size, Low Power: 2"x2.6", 5V @ 200 mA at full speed, 40 mA at lowest speed.

Visit http://www.bagotronix.com for info, prices, and FAQs

BAGOTRONIX
2900-1 Crescent Drive
Tallahassee, FL 32301
850-942-7905 phone & fax

INDEX OF ADVERTISERS

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

Page		Page		Page		Page	
87	Abacom Technologies	88	FieldServer Technologies	89	MCC (Micro Computer Control)	33	Renesas
91	All Electronics Corp.	93	Front Panel Express	57	Microchip	74	RoboBusiness Conf & Expo
92	AP Circuits	90	Futurlec	95	Micromint	82	Saelig Company Inc.
43	Arcom	90	Grid Connect	92	microEngineering Labs, Inc.	3	Scott Edwards Electronics, Inc.
7	Atmel	91	Hagstrom Electronics	90	MJS Consulting	88	Sealevel Systems
93	Bagotronix, Inc.	79	HI-TECH Software, LLC	91	Mosaic Industries, Inc.	21	Silicon Laboratories, Inc.
37	Beta Layout Ltd.	51	Holmate/Holtek Semiconductor, Inc.	5	Mouser Electronics	88	SMTH Circuits
49	Bitscope Designs	46, 87	ICOP Technology, Inc.	90	Mylydia, Inc.	10	SPLat Controls Pty Ltd
89	Brightan Systems	89	IMAGEcraft	C2	NetBurner	93	TAL Technologies
24	CadSoft Computer, Inc.	47	Imagine Tools	86	Nurve Networks LLC	C3	Tech Tools
87	Carl's Electronics	90	Intec Automation, Inc.	88	OKW Electronics, Inc.	40, 41	Technologic Systems
73	CCS, Inc.	91	Intrepid Control Systems	92	Ontrak Control Systems	87	Technological Arts
92	Conitec	92	Intronics, Inc.	10	P & E Microcomputer Systems Inc.	89	Tern, Inc.
65	CWAV	55	Jameco	80	PC-Instrument Mart	90	Trace Systems, Inc.
1	Cypress MicroSystems	81, 92	JK microsystems, Inc.	16	PCBexpress	91	Triangle Research Int'l, Inc.
91	DataRescue	70	JR Kerr Automation & Engineering	C4	Parallax, Inc.	83	Trilogy Design
86	Decade Engineering	15	Keil Software	27	Philips	81	Trinity Robot Contest
86	DLP Design	70	LabJack Corp.	86	Phytec America LLC	73	TronicsZone
31	Dynon Inst.	70	Lakeview Research	87	Phyton, Inc.	90	VNISource Design
80	Earth Computer Technologies	90	Lawicel HB	93	Picofab, Inc.	93	Weeder Technologies
89	EE Tools (Electronic Engineering Tools)	91	Lemon Studios	93	Pulsar, Inc.	88	Zagros Robotics
86	Elsevier, Inc.	17	Lemos International	89	R2 Controls	59	Zilog, Inc.
83	EMAC, Inc.	2	Link Instruments	71	R4 Systems, Inc.	87	Z-World
25	ExpressPCB	85	Linx Technologies	23	R.E. Smith		
32	FaabMedia, Inc.	63	MaxStream	11, 35	Rabbit Semiconductor		
86	FDI-Future Designs, Inc.	9	M16C Design Contest	63	Reach Technology, Inc.		

Preview of April Issue 177
Theme: Robotics

AVRcam: A Low-Cost Embedded Vision System

Multiple-Axis Stepper Motor Controller (Part 1): Design Basics

Digital RC Servo Controller (Part 1): 32-Channel Design

Automatic Gate Control

Practical Application for TDD (Part 2): Automated Test-Driven Environment

Embedded Security Design (Part 2): Circuit Board

USB ARM DAQ: Simple Data Acquisition

ABOVE THE GROUND PLANE **Foolish LED Tricks**

APPLIED PCs **Test-Driving the Micro64**

FROM THE BENCH **Stay in Touch:** Sensor Material for Robotics Applications

SILICON UPDATE **USB Easy Riders**

ATTENTION ADVERTISERS

May Issue 178 Deadlines

Space Close: March 10
Material Close: March 17

Theme: Communications

Call Sean Donnelly now to reserve your space!
860.872.3064
e-mail: sean@circuitcellar.com

Embedded Solutions.

Dozens of Embedded Controllers with Countless Configurations to fit ANY Industry.

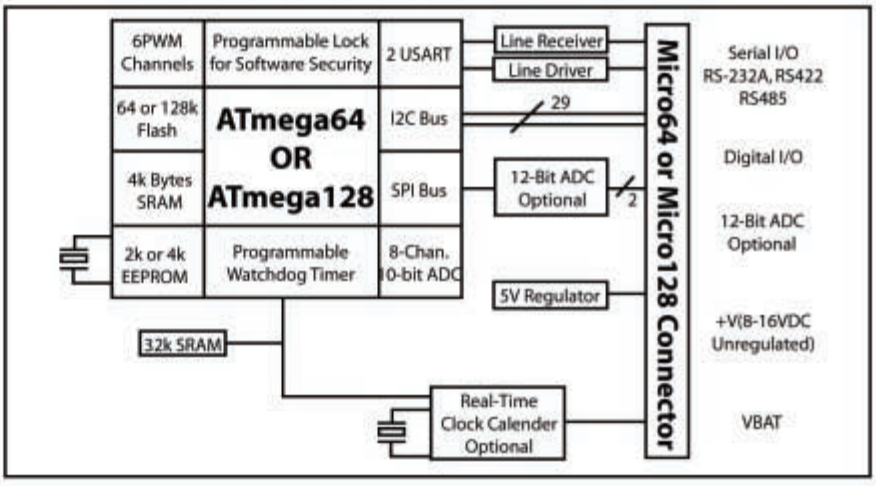
ATMEL AVR® Based Controller



- Small size - 1.5" x 2.1" x 0.52
- Dual powered - operates on +5V or 6.5-20V at 55 mA (typical)
- Program and Data Memories
  - 64k or 128k Bytes of In-System Reprogrammable Flash with 10,000 Write/Erase Cycles
  - 2 or 4 K Bytes EEPROM with 100,000 Write/Erase Cycles
  - 36K Bytes SRAM
  - Programming lock for Software Security
- Peripheral Features

Starting at $119 - Single Qty

- Optional 2-channel 12-bit ADC
- 8-channel 10-bit ADC
- 8 Single-ended Channels
- 7 Differential Channels
- 2 Differential Channels with Programmable Gain (1x, 10x, 200x)
- Byte-oriented Two-wire Serial Interface (I2C)
- Dual Programmable Serial USARTs
- 1 TTL
- 1 RS-232A, RS-422, or RS485
- Master/Slave SPI Serial Interface
- Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
- Two Expanded 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Two 8-bit PWM Channels 6 PWM Channels with Programmable Resolution from 1 to 16 Bits
- Programmable Watchdog Timer with On-chip Oscillator
- 29 Digital I/O that can sink or source 20 mA



Data sheets, Online Ordering and a Complete Catalog of Products at...

www.micromint.com

Delivering Embedded Controllers for 24 Years

Reduce Your Time to Market.

Minimize your time from conception to production by utilizing one of Micromint's market-proven controllers. Whether your concerns are digital or analog, inputs or outputs, Micromint has a product to fit your needs. Order quantities of one to thousands. Custom design and configurations are available.

With over 500,000 controllers in the marketplace, Micromint has been providing innovative, turn-key solutions to the OEM market for 24 years—from design through production, as well as packaging and shipping the final product. Our broad line of embedded controllers and turn-key solutions can turn your imagination into reality.

Visit our website @ www.micromint.com to see our complete line of OEM Solutions.

Micromint

115 Timberlachen Circle | Lake Mary, FL 32746 | 800-635-3355 | 407-262-0066 | Fax 407-262-0069



PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

Anniversaries

Every technical magazine I pick up lately seems to be editorializing about the 30-year anniversary of personal computing. Because all of us in the embedded-control community use the same technology, I suppose I have an obligation to chime in with my opinion and congratulations too. Well, up to a point, that is.

All these journalists keep pointing to one event the— *Popular Electronics* article about the Altair 8800 in January 1975— as the start of the revolution. Of course, the fact that Ziff-Davis owned *Popular Electronics* certainly doesn't reduce the number of editorials with that historical perspective. Anyone who was actually there at the time knows there was a little more to the timeline. Yes, this article was important, but "personal computers" really started with the 8008 a couple years earlier. (While the 4004 was technically a microprocessor, in my opinion its limited addressing range made it useless as a "personal microcomputer.") The first real microcomputers were the Micral (built in France) and the Scelbi 8H (built in the U.S.) in 1973. Both used the 8008.

This sensitivity to the history of the PC is completely personal. Ninety-nine percent of early computing initiatives were West Coast-driven, but the other 1% were East Coast-driven— in fact, here in Connecticut. When I heard about Scelbi, I made a beeline to where they were located in Milford, Connecticut (about 60 miles south of where I live). Consequently, my first computer, and the basis of my earliest articles, was a Scelbi 8B. To clarify the history, let me point out one other thing about Scelbi. Nat Wadsworth stopped making Scelbi computers just about the time when the Altair appeared. I remember him being upset that MITS (Altair's manufacturer) could sell a whole computer for what Intel was quoting to him as the price of 8080 chips. While Scelbi's immediate transition from hardware to book publishing looked to the world like he had failed as a hardware manufacturer, it wasn't from lack of talent or initiative. Few people know it, but Nat had a history of poor health and had to choose his business pursuits accordingly.

Finally, if all these columnists are using a magazine article as the starting point of the timeline, then they are missing an earlier article of equal significance (to us computer geeks, anyway). In July 1974, six months before the *Popular Electronics* article appeared, *Radio-Electronics* presented Jonathan Titus's Mark-8 8008-based computer. Two months later, Hal Singer started the *Mark-8 Newsletter*, one of the very first computer publications, and, coincidentally, the first place you would have seen a construction article by yours truly. ;-)

The starting point of a revolution is always a matter of opinion. I think it was the 8008, while most people in the media point to the 8080. Similarly, the 6800 or 6502 crowd acknowledges Intel, but the first real computer for them was very brand-specific. The business community thinks more in terms of a 20-year than a 30-year anniversary. For them the first real computer was a Pentium, not the 8088. The important thing to realize is that no single event (other than Intel's first microcomputer chip) is a fixed milestone, so anniversaries are somewhat gray. In fact, by the end of 1975, there were about 50 different processors from the likes of Texas Instruments, RCA, National Semiconductor, Motorola, and others. I hesitate to point to specific dates, but I guess we can all safely agree that it started about then.

Having firsthand experience of all these events does have its downside. I chuckle every time I shake hands with someone at a convention and they say, "I've been reading you since *BYTE*." Of course they don't realize that they've also just implied, "you old fossil <grin>." I laugh and typically answer them by saying, "That only means you've been doing this as long as I have, and we're both getting old."

The good news is that the last 30 years have been very good to me, and I owe my success primarily to the loyalty of my readers. I've lamented in a few past editorials that I was sitting at the table when some big computer company deals were being negotiated, and I chose to stay the course doing the things I enjoyed. Quite a few other editors and columnists went off and joined these ventures with various long-term results. I guess I just enjoyed writing and inventing too much back then to risk changing it all to reach for the brass ring.

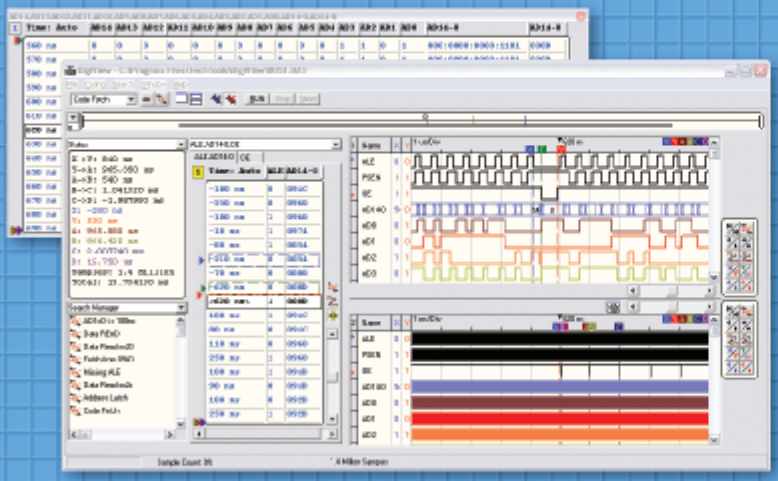
While most of the tech companies, magazines, and computer brands that were common 25–30 years ago are gone, *Circuit Cellar* is still here. And I am encouraged daily about our future by your enthusiasm and interest. Soon we will be approaching the length of time that *BYTE* was a viable publication, and I have every intention of passing that mark by a wide margin. Of course, if someone comes up to me at a convention 30 years from now and says, "I've been reading you since *BYTE*," someone shoot us both!



steve.ciarcia@circuitcellar.com

DigiView

100 MHz, 18 Channel, Portable Logic Analyzer!

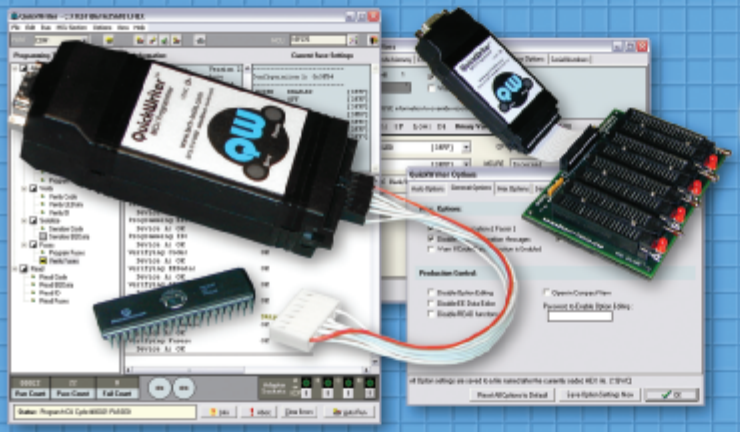


Auto Hardware Compression Captures from 128K to 30 Billion Samples @10ns resolution!

only $499.00

- Save up to 5 minutes of Data @ 10ns.
- Edge and Pattern Triggers.
- Real-time Hardware Compression.
- USB Powered for Portability.
- Includes Capture & Analysis Software.
- Pattern Searches with Match, Duration & Animation.
- Drag & Place and Drag & Snap Markers.
- Waveform Next/Previous/Nearest Edge snapping.
- Multi-Signal Data Tables with Link Groups.
- Split Waveform Views with Link Groups.

Download and try the Software!



QuickWriter

Multi-Function PICmicro® MCU Programmer In-Circuit & Gang Operation

only $199.00

- Supports 12, 14, 16 & 18 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.

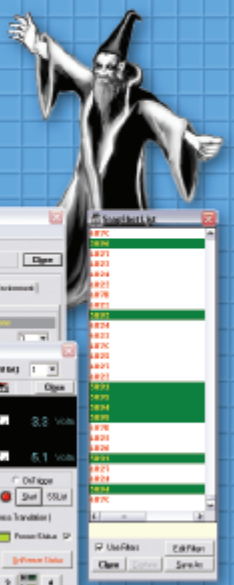
See Our Other Embedded Development Tools



EconoROM III & FlexROM III

EPROM & FLASH Emulation at its best!

from $179.00



TechTools

www.tech-tools.com

(972) 272-9392 • sales@tech-tools.com

Copyright © 2005 TechTools • DigiView, FlexROM, EconoROM and QuickWriter are trademarks of TechTools • PICmicro is a registered trademark of Microchip Technology Inc.

ping)



$24.95
#28015



Parallax's new Ping ultrasonic sensor provides a very low-cost, easy method of distance measurement.

- Distance measurement up to 3 meters +/- 1 cm
- Supply Voltage 5V +/-10% (Absolute: Minimum 4.5V, Maximum 6V)
- 3-pin interface (power, ground, signal)
- Measures 4.6 cm x 2.1 cm
- 20 mA power consumption
- Narrow acceptance angle
- Simple pulse in / pulse out communication
- Indicator LED shows measurement in progress

The Parallax Ping is available in single quantities for $24.95, or get a five-pack for $100!

Order online at www.parallax.com or call the Sales Department toll-free in the US at 888-512-1024 (Mon-Fri, 7am-5pm, PT).

www.parallax.com

Parallax, and the Parallax logo are trademarks of Parallax, Inc. ©2005 by Parallax, Inc. All Rights Reserved

PARALLAX INC