#204 July 2007

www.circuitcellar.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

INTERNET & CONNECTIVITY

MiniEmail Client

Internet Password Manager

Motion Tracking & Analysis

Are You Ready for 16 Bits?

07>
7 25274 75349 9
$4.95 U.S. ($5.95 Canada)

Announcing a complete hardware and software solution from NetBurner

The NetBurner MOD5234

ETHERNET CORE MODULE with eTPU

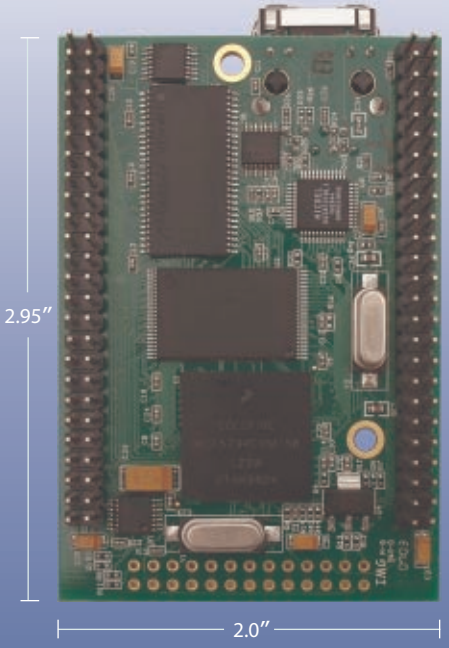
Only $99
Qty. 100

NetBurner MOD5234 Ethernet Core Module Features

INDUSTRIAL TEMPERATURE RANGE
| –40°C to +85°C

PERFORMANCE AND MEMORY
| 32-Bit CPU | Freescale ColdFire MCF5234 147 Mhz
| 2MB Flash Memory | 8MB SDRAM

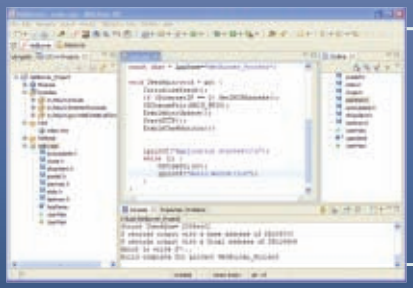
DEVICE CONNECTIVITY
| 10/100 Ethernet | 3 UARTs | 16-channel eTPU | I²C | SPI | CAN
| 47 Digital I/O | 16-bit Data Bus | SD/MMC Flash Card Support



Customize with NetBurner's Royalty-free Software Suite

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

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



All hardware and software is included with the

NetBurner MOD5234 Development Kit for only $299!

The Development Kit features NetBurner's Eclipse, an enterprise level professional IDE that provides editing, downloading and debugging in one environment.

Order the MOD5234 Development Kit: Product No. NNDK-MOD5234-KIT

NetBurner
Networking in One Day!

Product No. | MOD5234-100IR
Information and Sales | sales@netburner.com
Web | www.netburner.com
Telephone | 1-800-695-6828

freescale™
Alliance Member

Real Mixed-Signal Programmability.

Get PSoC®. Because change happens.

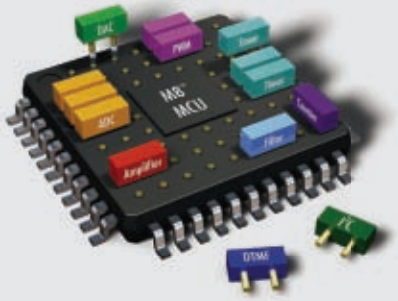
PSoC flexibility enables changes anytime: at concept, through production, in the field. Specifications change constantly. Yet pressures to differentiate, minimize costs, and speed time-to-market remain the same. To stay ahead of the curve, you need flexibility, programmability, and scalability. PSoC's unique programmable architecture delivers this and more. Futureproof your design; make PSoC your agent of change.

PSoC delivers:

- The configurability of an FPGA, the mixed-signal integration of an ASIC, and the familiarity of an MCU.
- Reusable IP, compatible device families and variable resource options ensure you can optimize design efforts and accommodate changes.
- The industry's first visual embedded design tool, PSoC Express™, speeds design time, enabling you to generate a complete design without writing a single line of code.

GET STARTED WITH PSoC NOW.

- *Download our "Change Happens" White Paper and get 50% off a PSoC development kit:* **www.cypress.com/changepaper**
- *Download free PSoC Express™ visual embedded software:* **www.cypress.com/changesoft**
- *Request free PSoC device samples:* **www.cypress.com/changechip**
- *Free online training:* **www.cypress.com/changetrain**
- *Purchase PSoC development tools:* **www.cypress.com/changetools**



PSoC includes programmable analog and digital blocks, a fast MCU, flash and SRAM memory, all in a compact package (as small as 3mm x 3mm).

CYPRESS PERFORM

Link Instruments

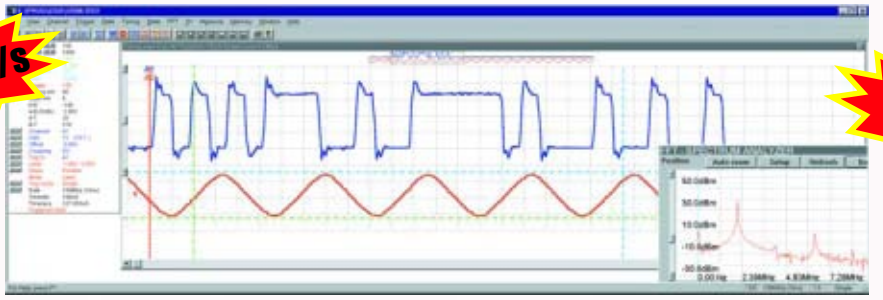
PC-Based Test Equipment

Digital Oscilloscopes

- 2 Channel Digital Oscilloscope
- **500 MSa/s** max single shot rate
- 1Mpt sample memory
  250 MSa/S (Dual channel) 512 Kpts
  500 MSa/S (Single channel) 1 Mpts
- Only 9 oz and 7" x 3.5" x 1.5"
- Portable and Battery powered
- USB 2.0
- FFT Spectrum Analyzer

DSO-8202 (200MSa,128K) **$799**
DSO-8502 (500MSa,1MPt) **$950**

500MSa/s

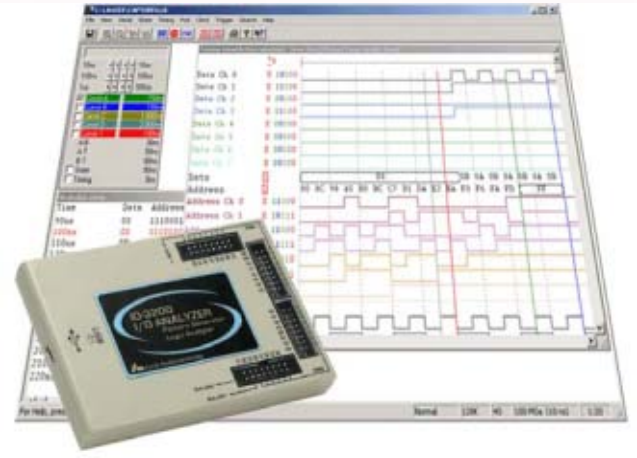


1Mpts

Logic Analyzer & Pattern Generator

NEW!

Portable
3.2" x 3" x 0.65"
USB 2.0 Powered



- Logic Analyzer (32 channels)
- Pattern Generator (up to 32 channels)
- up to 400 MSa/s
- Variable Threshold
- 2 External Clocks
- SPI output and disassembly
- I^2C output and disassembly
- up to 2Msamples/ch

IO-3208A	**$750**
IO-3232A	**$899**
IO-3232B	**$1399**

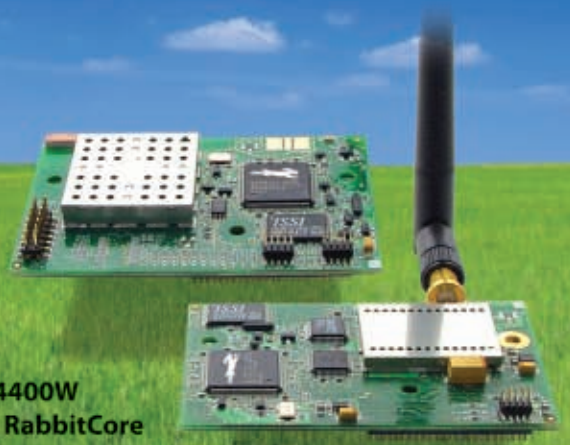
Link Instruments (973) 808-8990

www.Linkins4.com

New! Next Generation Rabbit 4000

Go Wireless
Go Rabbit®

Check out our new Wi-Fi and ZigBee® core modules — the latest additions to our pin-compatible family.



RCM4510W
ZigBee RabbitCore®
$72 (qty. 100)

Wireless Development Kits
Only $199 (reg. $299)

RCM4400W
Wi-Fi RabbitCore
$99 (qty. 100)

Applications

- Industrial Control
- Remote Terminal Unit (RTU)
- Building Automation
- Data Acquisition

RCM4500W Family

- ZigBee/802.15.4
- 49 GPIO
- 6 Serial Ports

RCM4400W Family

- Wi-Fi/802.11
- 35 GPIO
- 6 Serial Ports

RABBIT Semiconductor

2900 Spafford Street, Davis, CA 95618 Tel **530.757.8400**

Order Your Kit Today

Development kits include everything to start your development, Dynamic C® software, wireless RabbitCore module, development board, programming cables and accessories.

www.rabbitkits-wireless.com

07115

TASK MANAGER

A Look Back and Forward in Tech Time

When did you start surfing the 'Net? I began using it years ago when I first started college. At the time, I was living in an old building and sharing a room with two other students who didn't have much experience on the 'Net either. The building was pretty low-tech in comparison to many of today's 21st-century setups. Our study room had one light, one phone jack, no cable, and our door actually had a lock and key. (I didn't have one of those fancy access cards that students now carry on their key chains.) To keep sane during the first week of the semester, we took turns playing video games on a small TV. But, as you can imagine, that got old quickly. That's when we began using the 'Net for entertainment during our late-night study breaks. We'd look for sports scores, the day's news, and info about our favorite musicians—but that was about it.

When I went to London in 1998, I found myself shot back a few years in "tech time." The room I was living in didn't have phone or Internet access. To send an e-mail or browse the 'Net, I had to walk across the street to a hole-in-the-wall Internet café and pay about £1 for 30 minutes of online time. If I was in the library, I had to wait in line in the PC room for a workstation to free up. It wasn't until a few years later when I was back in the U.S. that my daily online experience began to resemble what it is today.

In this issue, we present you with two interesting articles about Internet-related design projects. Both projects prove that with a little effort, you can use your design skills to create innovative solutions to common problems.

Ever need to check your e-mail when your PC was off? In "MiniEmail" (p. 14), Alexander Mann describes how he built a compact MCU-based mail client. The quiet system—which features an Atmel ATmega32 microcontroller and a Microchip Technology ENC28J60 Ethernet controller—continually checks for messages and enables you to read them as soon as they arrive.

On page 22, Carlos Cossio describes a handy password management system. You can use the system to enter, display, and securely store all of the passwords and usernames that you use for various web sites. An added bonus is that you can program the system to automatically complete username and password fields for you when you click on your favorite security-protected web site.

Finally, note that this issue is the perfect antidote for anyone who has design contest fever. On page 55, we provide photos and descriptions of the projects that placed in the Luminary Micro DesignStellaris2006 contest. Check out the descriptions of the winning projects and then go to www.circuitcellar.com/designstellaris2006/winners/winners.html to view the contest files and more. And if you're gearing up to enter a project in the Microchip 16-Bit Embedded Control Design Contest, be sure to read through Jeff Bachiochi's contest primer on page 76. Jeff covers everything you need to know about Microchip's 16-bit microcontroller and DSC families.

It's time to put your design skills to the test. Good luck! As usual, keep me posted on your progress.

cj@circuitcellar.com

C. Abate

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

MANAGING EDITOR
C.J. Abate

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Jeff Bachiochi
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Steve Bedford
Ken Davidson
David Tweed

ASSOCIATE EDITOR
Jesse Smolin

CHIEF FINANCIAL OFFICER
Jeannette Ciarcia

MEDIA CONSULTANT
Dan Rodrigues

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNER
Mary (Turek) Sobuta

STAFF ENGINEER
John Gorsky

ADVERTISING

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

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

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

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

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

PRINTED IN THE UNITED STATES

CONTACTS

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

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

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

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

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

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

SuperH Flash Microcontroller Reaches Speeds up to 160MHz

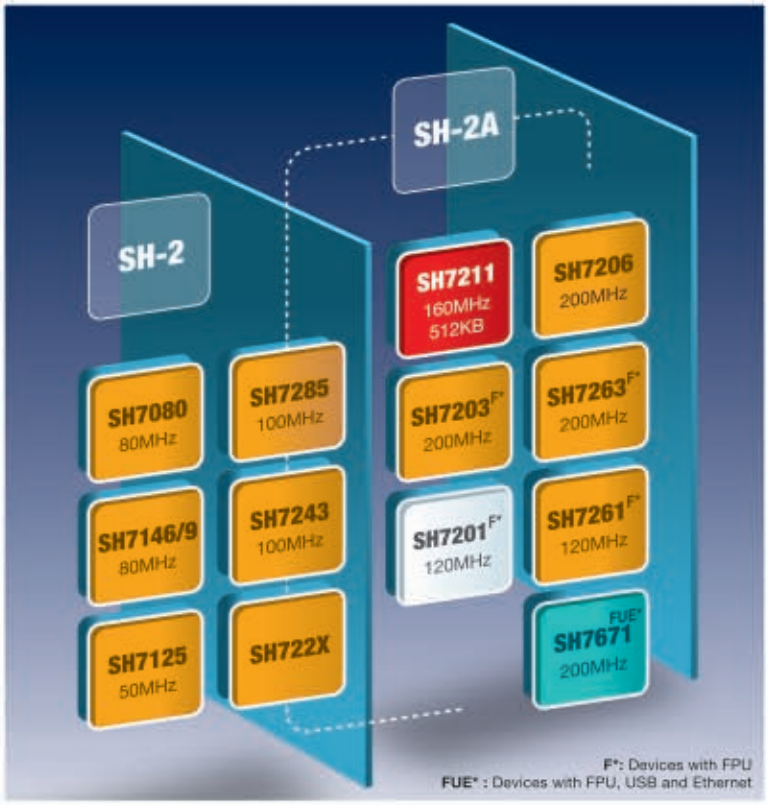
Superscalar performance,
high-speed on-chip flash memory access, and much more

Renesas Technology

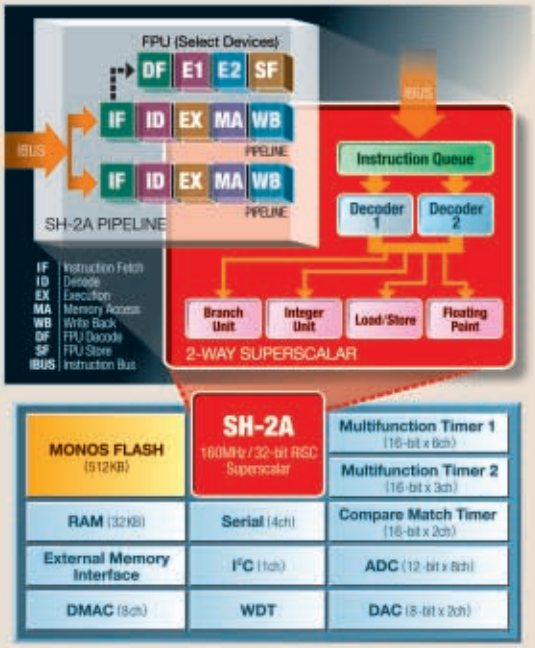
No.1* supplier of microcontrollers in the world

proudly introduces the SuperH family of superscalar flash devices. With 160MHz superscalar performance, high-speed access to on-chip flash memory and up to 200MHz CPU operation from SH-2A devices, the SuperH family allows you to incorporate all of your brilliant ideas into super high performance applications. The SuperH RISC engine and the SH-2A core are establishing new performance standards in the industry. The SH-2A core is geared towards systems that demand real-time, high-precision control and require a combination of high performance CPU and high-speed flash.

SuperH MCU Roadmap



HOT Products **SH7211F**
(R5F72115D160FPV)



Top Reasons to Select SuperH

480 Dhrystone MIPS at **200** MHz

- **Superscalar Performance**
  - Two instructions are executed per cycle at 160MHz
  - World's fastest embedded FLASH with 12.5ns read access time.
- **Fast Real-Time Control**
  - Register-Banks enable 37.5nsec (6 cycles) interrupt latency time.
- **High Integration**
  - Large on-chip flash and on-chip RAM
  - Advanced 16-bit PWM timers to drive two motors simultaneously
  - 1.25μsec 12-bit A/D conversion with 3 sample & hold circuits

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

#1 MCU REACH FURTHER

Get Started Today -

Register to qualify for a SuperH MCU Kit

www.america.renesas.com/ReachSH/d

Starter Kit for SH7211F:
R0K572115S000BE
USB Emulator:
HS0005KCU11H

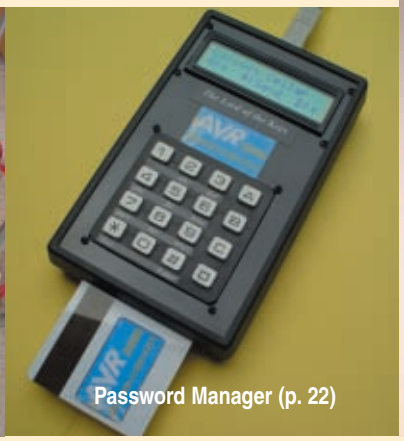
Renesas Technology Corp.

Everywhere you imagine. RENESAS

July 2007: Internet & Connectivity

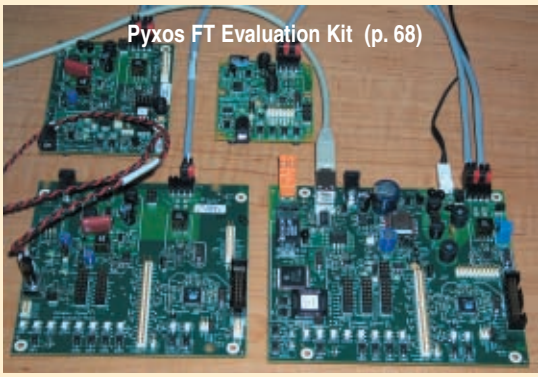
FEATURES


MiniEmail (p. 14)


Password Manager (p. 22)


Add a Flash Drive (p. 59)

COLUMNS


Pyxos FT Evaluation Kit (p. 68)

DEPARTMENTS

More MIPS - Less mA - get AVR32!

AVR32 outperforms industry-standard architectures in both performance and code density!

AVR32 is designed for low power applications demanding high performance.

The Harvard architecture and multiple high-speed system buses guarantee exceptional processing performance whilst various sleeping modes and Dynamic Frequency Scaling ensures low power consumption. Both UC3 and AP7 sub-families are supported by Atmel's large range of development tools, and the free AVR32 Studio makes it easy to start developing C/C++ code today!

AVR32 UC3 Family
32-bit Flash Microcontrollers

- Up to 512 KB in-system programmable Flash and 64 KB RAM on-chip
- DSP instruction set
- Low latency interrupts
- Large number of I/O pins
- Communication interfaces: Ethernet MAC, USB device & OTG
- System extension capability through SRAM/SDRAM interface

80 Dhrystone MIPS at 66 MHz
40 mA at 66 MHz, **3.3V**

AVR32 AP7 Family
32-bit Application Processors

- DSP and SIMD instruction sets
- Low latency interrupts
- Large number of I/O pins
- Communication interfaces: Ethernet MAC, Hi-Speed USB
- Man-Machine interfaces: LCD, Camera, Audio
- Linux supported by Atmel

210 Dhrystone MIPS at 150 MHz
75 mA at 150 MHz, **1.8V**

Note: for more details about the test conditions for performance and consumption, please refer to datasheets.

Get more at: **www.atmel.com/AVRman**

NEW PRODUCT NEWS

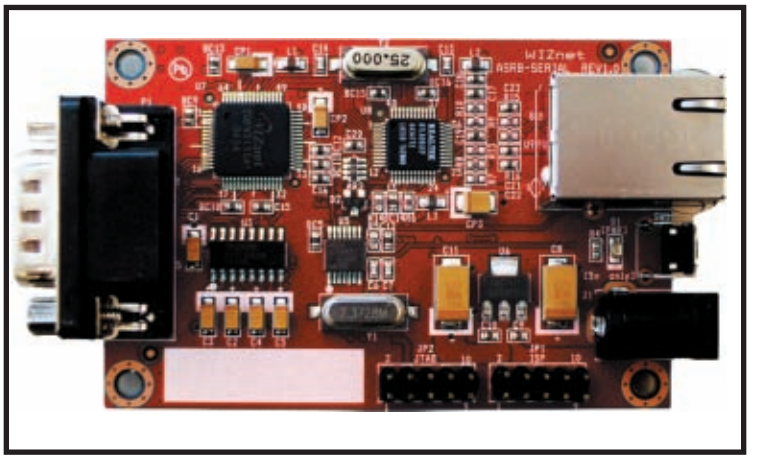
Edited by John Gorsky

NEW ETHERNET ADAPTERS MEET MANY NEEDS

WIZnet is now offering several new adapters to meet your Ethernet connectivity needs. The **WIZ100SR** and **WIZ110SR** serial-to-Ethernet modules are upgraded versions for use in access control, medical devices, and other industrial and consumer products. The newly released W5100, which includes embedded hardware TCP/IP, Ethernet MAC, and PHY is applied in these modules to guarantee better performance, stability, and a lower cost. Both modules support a maximum serial speed of 230 kbps, 10/100 Mbps Ethernet, and autoMDIX. The WIZ100SR is a plug-in module with a 12-pin header. The WIZ110SR has DE-9 and RJ-45 interfaces. A configuration program is available that allows users to configure all the parameters of the modules and control them through the Ethernet connection.

The WIZ200USB is a USB-to-Ethernet module for use with various USB devices, such as mass storage, CDC, and USB printers. It supports both USB 2.0 full-speed and OTG and can act as a host or device.

The WIZ800MJ is an all-in-one module that can provide a one-stop solution for Internet connectivity to existing embedded devices. In this module, all Internet-enabling components, such as an Ethernet controller, a transformer, a MagJack, and TCP/IP stacks are included. It supports a memory bus and an SPI for microcontroller interfacing. Anyone can implement Ethernet functionality by adding the WIZ800MJ without changing the existing microcontroller in an embedded system.



The WIZ100SR and WIZ110SR are less than **$20**. The WIZ200USB costs less than **$25**. The WIZ800MJ costs less than **$10** in 1,000-piece quantities.

WIZnet, Inc.
www.wiznet.co.kr

USB HUB WITH LOCKING CONNECTORS

The **SeaI/O-270U** is a seven-port optically isolated USB 1.1/2.0-compatible hub, which protects computers from damaging power surges, spikes, and ground loops commonly found in industrial and OEM applications. The SeaI/O-270U offers system designers an effective tool for interfacing USB peripherals in virtually any environment. In addition to protection from electrical damage, the SeaI/O-270U includes the patent-pending SeaLATCH locking USB connector design to prevent the accidental cable disconnection of both upstream and downstream USB ports.

The hub provides up to 1,500-VAC protection from harmful voltages injected on USB peripheral cabling. Status LEDs on the hub indicate external power, connection to the host, and fault conditions. The hub is housed in a rugged metal enclosure and each of the seven USB downstream connections provides a full 500 mA of power suitable for all USB peripherals.

SeaI/O-270U hubs are supported in Windows, Linux, and all other operating systems that recognize USB. Simply provide power to the hub using the included power supply, plug the hub into an available USB port with the included cable, and the hub is automatically recognized.

Standard SeaI/O-270U hubs operate between 0° and 70°C. Designers can special-order extended temperature range (–40° to 85°C) hubs. SeaI/O-270U hubs are available now. The SeaI/O-270U costs **$379** and up.

Sealevel Systems, Inc.
www.sealevel.com

NEW PRODUCT NEWS

16-PORT T1/E1/J1 TRANSCEIVER

The **DS26519** is the industry's first 16-port T1/E1/J1 long- and short-haul transceiver. At 16 ports, this single-chip transceiver (SCT) offers the highest integration and lowest cost per port in the industry. The high integration of the DS26519, along with its low 1.8-V/3.3-V power operation, makes it ideal for CSUs, DSUs, muxes, switches, routers, channel banks, and test equipment.

The DS26519 saves space and cost by eliminating the need for the mechanical relays typically required by redundancy protection implementations. The device uses hitless protection switching (HPS) to provide reliable redundancy protection by presenting high impedance at transmit outputs and receive inputs when power is not applied. Additional space and cost savings are provided by its software-selectable, internal receive-and-transmit-side termination for 75-Ω E1 coaxial, 100-Ω T1 twisted-pair, 110-Ω J1 twisted-pair, or 120-Ω E1 twisted-pair applications.

The DS26519 provides comprehensive functions for monitoring performance and communicating diagnostic information. The diagnostic suite includes, but is not limited to, a bit-error-rate tester, loss-of-signal indication, analog, local, remote and dual loopbacks, large error counters for BPV, CV, CRC-6, CRC-4 and E-bit, in-band repeating-pattern generation/detection, and JTAG. Also included is an internal clock-rate adapter and an HDLC controller with a 64-byte buffer in both the transmit and receive paths.

This device is also available in a software-compatible eight-port version, the DS26518. Evaluation kits with supporting software, driver source code, and reference designs are available for both devices. Prices start at **$69.96** for the DS26519 and **$40.60** for the DS26518 in 5,000-piece quantities.

Maxim Integrated Products, Inc.
www.maxim-ic.com

VARISTORS PROVIDE HIGHER SURGE CURRENT CAPABILITY

The new **LS41** and **LS42** series of varistors offer a higher surge current capability and are designed for surge voltage protection in industrial and consumer electronics. Both series have retained the compact dimensions of the proven LS40 series (37.5 × 46 mm).

The surge current capability of the LS41 was increased by 25% and is now 50 kA at a pulse form of 0.4 µs. These types are designed for rated voltages of 130 to 460 VAC.

The surge current capability of the LS42 was increased by as much as 62% to 65 kA. This series is designed for rated voltages between 250 and 460 VAC. Approval to UL 1,449 and CSA 22.2 has been granted for both series.

The varistors cost between **$7.50** and **$10** each, depending on the type.

EPCOS, Inc.
www.epcos.com

NEW PRODUCT NEWS

LOW-COST TRANSCEIVER-BASED FPGA

The new **Arria GX FPGAs** are optimized to support PCI Express (PCIe), Gigabit Ethernet (GbE), and Serial RapidIOTM (SRIO) standards up to 2.5 Gbps. These standards are rapidly emerging as mainstream protocols in a wide variety of markets and applications. Features of the Arria GX family include the proven Stratix II GX transceiver technology, flip-chip packages for superior signal integrity, software tools, and verified IP cores.

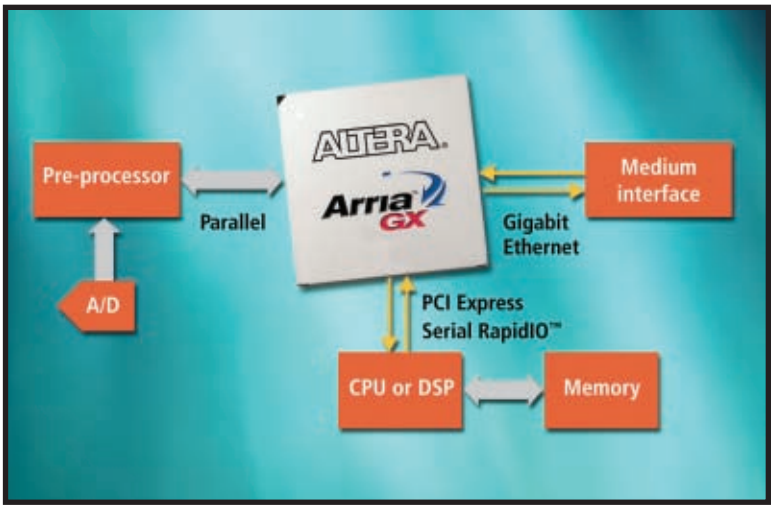
The Arria GX family comprises five devices ranging in density from 21,580 to 90,220 logic elements (LEs), up to 4.5 Mb of embedded memory, and up to 176 multipliers. It is built on TSMC's established 90-nm process. The family addresses the rapidly growing need for low-cost FPGAs with transceivers in the communications, computer, storage, and industrial markets.

The FPGAs provide up to 12 full-duplex transceiver channels optimized to implement the PCIe, GbE, and SRIO protocols. The devices use flip-chip packaging technology, which provides a significant signal integrity advantage over wire-bond packages when combining transceivers with advanced memory interfaces.

Arria GX FPGAs offer a unique design environment that includes software tools, complete characterization reports, reference designs, and verified IP cores that meet compliance and interoperability. A protocol-specific development kit with support for PCIe x1 and x4, SRIO, and GbE is also available. Designers can begin their Arria GX designs today by downloading and installing the easy-to-use Quartus II development software version 7.1.

The EP1AGX50CF484C6 costs **$50** in 25,000-piece quantities.

Altera Corp.
www.altera.com



OSCILLOSCOPES PACK PERFORMANCE IN A LUNCHBOX-SIZED INSTRUMENT

Six new oscilloscopes that break the paradigm of portable instrumentation are now available. With 1 Mpts deep memory, the new **5000 Series oscilloscopes** merge high-resolution and lab-quality waveform acquisition and measurement capability with a lightweight, portable frame. For the first time, high-resolution measurements, typically taken with larger, more expensive instrumentation, can be taken wherever they are needed, either in the field or at the bench.

The oscilloscopes allow engineers to discover the root of debug issues much faster than those using previous-generation portable oscilloscopes. Built to achieve a 100,000 waveforms per second acquisition rate, more than 25 times faster than the next closest competitor, the 5000 Series offers industry-leading glitch capture capability in its class. A high-definition XGA display system, which displays signals in 256 levels of intensity grading, highlights varying degrees of signal activity, which provides an unrivaled ability to view signal anomalies as they occur.

To further enhance the customer experience, the 5000 Series oscilloscopes also integrate a full suite of connectivity standards, including front- and back-panel USB ports, GPIB, LAN, and XGA-out.

The 5000 Series oscilloscopes are available now. They start at **$4,050** for the DSO5012A.

Agilent Technologies, Inc.
www.agilent.com

<P>DESIGN. CONNECT. REALIZE WITH EZ WEB LYNX.</P>

<P>Simply programmable in just HTML!</P>

A simple embedded Ethernet integration device to get your products online fast.

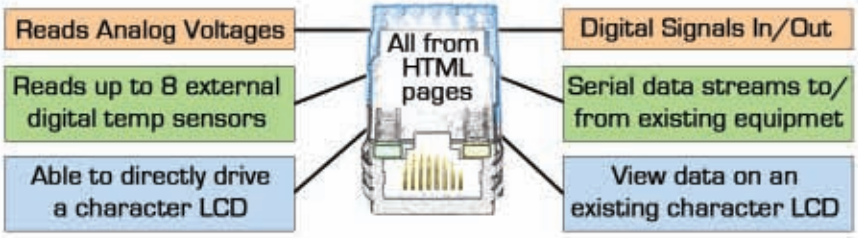
- Reduce development and engineering time
- Extended HTML tags allow for dynamic web content
- No other technical knowledge needed for programming device
- Easily added to existing electronics to gain ethernet capability

- Flash Memory for custom web pages
- Send email alerts
- Development Kit Includes:
  > Two EZ Web Lynx
  > Docking station for programming/testing
  > Power supply and all cables
  > HTML editor and upload utility
  > Printed manual

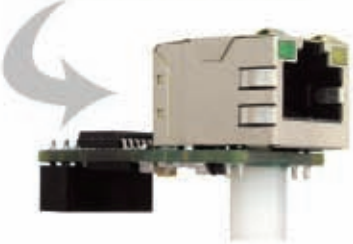
Per Unit	3.3V	5V
100	$22	$27
26-99	$30	$35
1-25	$49	$54
Development Kit	$149	

[Call for 1000+]

3.3 V with 17 I/O
or 5V with 11 I/O



ACTUAL SIZE



www.EZWEBLYNX.com • 262.522.6500 x45

NEW PRODUCT NEWS

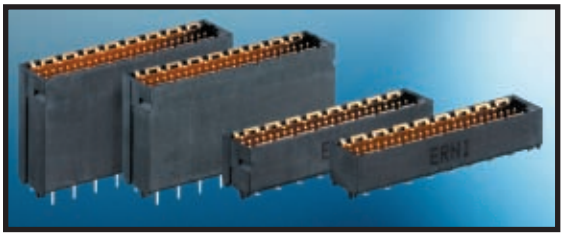
CONNECTOR FAMILY ENABLES FAST MEZZANINE CONNECTIONS

The new **MicroSpeed connectors**, available in a variety of different heights, are ideal for high-speed data transfer in telecom and datacom applications. The 1-mm modular shielded SMT connector system consists of two contact rows and two outwardly arranged shield plates.

With the availability of different height versions of the male (1, 2, 9, and 10 mm) and female (4, 6, 8, and 10 mm) connectors, it is possible to create 16 different stacking heights from 5 mm to 20 mm. A major advantage is the multiple-connector mating capability based on the high-mating tolerance that allows the mating of more than one connector pair at the same time.

For impedance purposes, the longitudinal pitch of MicroSpeed connectors is 1 mm and the transverse pitch is 1.5 mm. The differential signal pairs can be arranged horizontally or vertically. Optimized crosstalk behavior is achieved with a vertical (transverse to longitudinal direction) configuration of the signal pairs, and the pairing arrangement of signal and shield contact pairs.

While the signal contacts are made in SMT, the user can select between two terminal options for the shielding, depending on the application: SMT or THR connection. The shielding pins also assure robust strain relief of the connector. The length of the connector modules is 27 mm with 50 signal contacts and two shield plates. They can be easily lined up with virtually no loss of board space.



Large-volume prices start at **$9.50** for a mated connector pair.

ERNI Electronics
www.erni.com

SERIALIZER/DESERIALIZER FOR PORTABLE ELECTRONICS WITH CAMERAS

The **FIN212AC** is the first µSerDes device specifically designed for serializing high-speed signals in multimegapixel resolution CMOS and CCD image sensors. Using proprietary CTL I/O for serializing up to 12 bits at speeds up to 40 MHz, the FIN212AC can be configured as a serializer or deserializer and is implemented in pairs. Patented CTL technology operates well in noisy RF environments by eliminating additional shielding required by similar high-speed signals while offering the industry's lowest EMI (–110 dBM). The FIN212AC is especially beneficial when used in portable electronics with clamshell and slider form factors, where minimizing the number of signals across a hinge reduces board space and improves reliability.

The FIN212AC offers selectable LVCMOS edge rates and pulse widths to increase design flexibility while minimizing EMI. These specialized features allow the designer to tune the FIN212AC to a particular frequency range and enable implementations for both RGB and microcontroller interfaces without requiring software modifications. Although it primarily targets 8- to 12-bit camera interfaces in cell phones, the FIN212AC is also well suited for applications with similar parallel interfaces with up to 12 bits, such as laptops with integrated web cams, VOIP phones, and security cameras. Featuring ultra-low, Power-down mode (about 0.1 µA) when not transmitting data, the FIN212AC is an extremely low-power solution for conserving battery life in portable applications.

The FIN212AC costs **$1.49** (BGA) and **$1.30** (MLP).

Fairchild Semiconductor Corp.
www.fairchildsemi.com

NEW PRODUCT NEWS

Visit www.circuitcellar.com/npn for more New Product News.

TILT SWITCH BOOSTS SENSITIVITY TO SMALL CHANGES

The **DSA01** is a new combination tilt/sensor switch. The DSA01, which is triggered when tilted beyond ±10° of the horizontal, brings unique sensitivity to small angle changes.

Engineered to replace mercury and pendulum switches, the DSA01 is environmentally friendly and contains no mercury. In addition, the switch features a sealed body construction that ensures high-contact reliability.

The switch is cylindrical, measuring 29.3 mm in length and with a diameter of 11.5 mm. An optional PCB adaptor is available to simplify the assembly. The PCB adaptor eases mounting by securely housing the DSA01.

The DSA01 tilt switch is a single-pole single-throw maintained circuit. It has an On angle range of 10° to 170° and an Off range of 190° to 350°, allowing 500 ms of settling time between states. The resistive load is 0.1 A at 12 VDC. The mechanical life and electrical life are rated at 100,000 operations minimum.

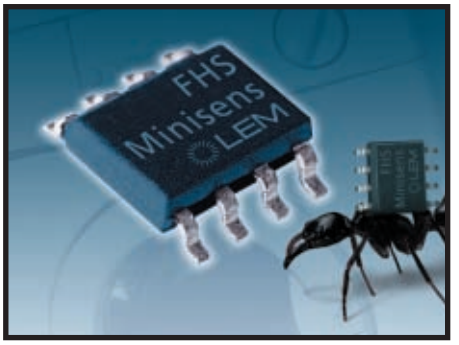
The DSA01 housing is constructed of PBT material. Rubber rings are nitrile butadiene rubber. The contact balls and terminals are brass with nickel plating. The operating temperature range is –10° to 70°C. The switch is rated to withstand 90% humidity for 96 h at 40°C.

NKK Switches
www.nkkswitches.com

ISOLATED CURRENT MEASUREMENT TRANSDUCER

The **Minisens** is a miniature integrated circuit transducer for AC and DC isolated current measurement up to 100 kHz. This new component offers full isolation (no optocouplers required) and high sensitivity (from 20 to 200 mV per amp of primary current) with no insertion losses. To reduce manufacturing costs, it is mounted directly onto a PCB as an SMD.

Minisens integrates, in one mixed-signal ASIC, Hall-effect sensors with a magnetic concentrator to allow direct-current measurement without the need for an additional magnetic core. The noncontact measurement enables an almost unlimited current level, eliminating the need for current flow through the device. The only limiting factor is the thermal capacities of the primary conductor. The current can be carried either by a track (or tracks) located on a PCB underneath the Minisens or by a cable or bus bar under or above the IC. The unlimited design possibilities for the primary conductor allow current measurement of 70 A or higher.



Many parameters of the ASIC can be configured by on-chip nonvolatile memory: adjustment of the transducer's gain, offset, polarity, temperature drift, and gain algorithm (proportional to, or independent of VDD). Two outputs are available: one filtered to limit the noise bandwidth and one unfiltered to ensure a response time of less than 3 µs. The LEM Minisens costs $1.96 each for 1,000 pieces.

LEM
www.lem.com

MiniEmail

A Compact MCU-Based Mail Client

Alexander's mail client system uses an Atmel ATmega32 microcontroller and a Microchip Technology ENC28J60 Ethernet controller to continually check for e-mail. When mail arrives, you can immediately read it on the system's LCD and respond with a standard keyboard.

E-mail is rightly regarded as one of the "killer applications" of our age. There's no doubt that having the ability to send messages to anyone in the world has made a positive impact on our daily lives. With that said, it's clear that even the fastest e-mail service is futile if you check your mailbox only once or twice per day. If you want to read your mail as soon as it arrives, the only feasible solution has been to keep your PC running all day so you can check for new mail on a regular basis. This is not the smartest solution. Not only can it get expensive, it can be annoying to listen to your PC's fans run all day and night.

In this article, I will present a new solution. My MiniEmail system is a compact microcontroller-based mail client (see Photo 1). The silent, easy-to-use system doesn't require a lot of power and it is immune to mail worms. Another advantage is the system's short start-up time. If you want to write a quick e-mail but your PC is off, you can simply switch on the miniature e-mail client and start writing without having to wait for your PC to boot up and load the necessary applications. All you need is an Ethernet connection and the MiniEmail system.

HARDWARE

The hardware for the MiniEmail system is inexpensive. It cost me about $50. The LCD is the most expensive part.

To keep things simple, I left the system's power supply, 5- to 3.3-V conversion, crystals, and latch out of Figure 1. The main components are an Atmel ATmega32 microcontroller and a Microchip Technology ENC28J60 Ethernet controller.

Because a mail client is a piece of complex software, you need a fast microcontroller that has a considerable amount of program space. The MiniEmail system uses almost all of the ATmega32's features, including the SPI, internal EEPROM and SRAM, counters, USART interface, sleep modes, all 32 I/O lines, and most of the 32 KB of program memory.

The ENC28J60 is a stand-alone Ethernet controller that provides basic functionality for transmitting frames over an Ethernet connection. It has 8 KB of built-in SRAM, which can be divided into transmit and receive buffers as desired, and it provides several interrupt sources (e.g., when new packets have arrived). The ATmega32 also has 128 KB of external SRAM connected as well as an LCD, which is a standard module with a resolution of 128 × 64 pixels.

Take a look at the ATmega32's pin connections in Figure 2. Ports A and C are used as 8-bit-wide general I/O ports, one of which is latched using an

Photo 1—The complete MiniEmail system includes an LCD, a keyboard, and several connections.

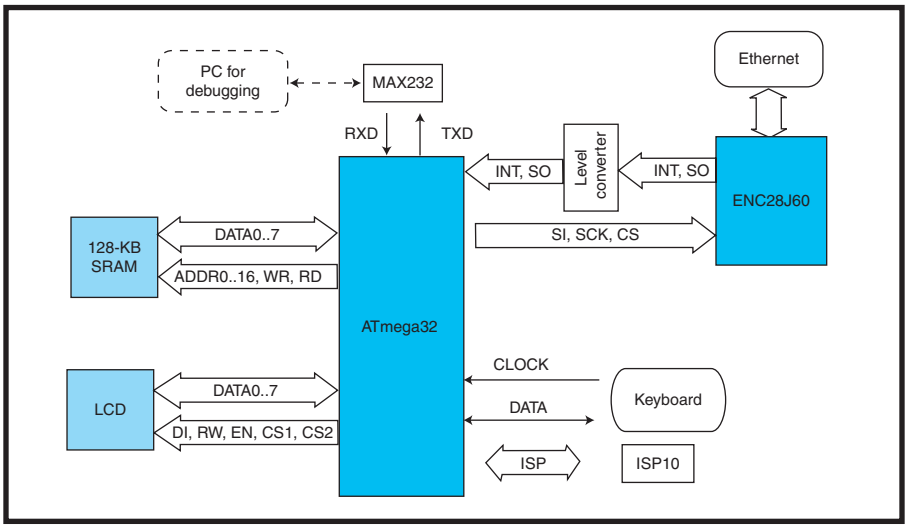


Figure 1—This is a block diagram of MiniEmail's hardware. The arrows indicate the directions of data flow between the devices. (The rounded boxes indicate parts that do not sit on the circuit board.)

NXP Semiconductors 74HC573. The two ports provide data connections to the LCD and SRAM (U3). For the SRAM, you need three additional wires: write (*RAM_WR), read strobe (*RAM_RD), and the seventeenth bit of the address (ADDR16). The LCD connector (CON1) uses five additional wires (for the signals CS1, CS2, DI, EN, and RW). CS1 and CS2 are taken from the general I/O port A (DATA6 and DATA7) and determine which of the two halves of the LCD is selected (i.e., the two controllers on the LCD module you are talking to). RW (where you can use ADDR16 again) sets the direction of the LCD access (read or write). DI describes the type of instruction sent to the LCD. EN is the enable signal for read and write cycles.

For the keyboard, you need only two pins: KEY_DATA and KEY_CLOCK. The clock signal must be connected to an external interrupt pin, INT1. One additional wire is needed to switch the latch (LE).

You are left with eight I/O pins on the ATmega32's ports B and D. RXD and TXD are connected to a MAX232, an RS-232 level converter that also provides the negative supply voltage needed for the LCD (LCD_VOUT in Figure 2). The ATmega32's USART functionality is used as a debugging interface. It isn't needed for normal operation.

The ATmega32 and ENC28J60 communicate via their SPIs. Because you need the Atmega32's SPI pins MOSI, MISO, and SCK for in-system programming, they are multiplexed with a 74HC4053. As long as reset is active, pins five to seven on port B of the Atmega32 are connected to the ISP10 connector via the 74HC4053; otherwise, they are connected to the ENC28J60's SPI. The ENC28J60 needs a supply voltage of 3.3 V provided by a 3.3-V voltage regulator (U2), but it is 5-V compatible, so you can connect it directly to the ATmega32's outputs. In the opposite direction, you need a level conversion, which is done by a 74HCT08 (U1) for the interrupt signal, which is connected to the ATmega32's INT0 INPUT pin. The rest of the circuit concerning the ENC28J60 is taken from its datasheet.

The PWM pin (PB3 on the ATmega32) is used to adjust the brightness of the LCD's backlight. This feature is used to switch off the backlight in Standby mode and flash the backlight to alert you when new e-mail arrives.

Photo 2 shows the circuit board without the cables, which are normally connected to it. The ATmega32 is in the middle. The ENC28J60, together with its peripherals, is in the top-right corner of the circuit board. Moving from left to right on top, there are two wires for the 5-V supply voltage, the PS/2 connector for the keyboard, the nine-pin connector for the USART interface, and the RJ-45 Ethernet connector in the top-right corner. An ISP10-compatible connector (on the bottom right) is used for the in-system programming of the ATmega32 microcontroller. The LCD is connected to the circuit board via a 16-pin connector. (You need two additional wires from the pin header next to the ATmega32. This is actually a design error and not recommended for reproduction. You should use an 18-pin connector.)

There are three LED status indicators. The red one next to the ISP10 connector is illuminated when the ISP interface is active. The green and the red LEDs next to the ENC28J60 blink when packets are received (red) and sent (green).

SOFTWARE

The firmware for this project is posted on the *Circuit Cellar* FTP site. I wrote the firmware in C language with a few small parts of inline assembler. I used the open-source software suite WinAVR, which includes the GNU GCC compiler with special libraries for AVR devices and avrdude, a tool for the in-system programming of AVR microcontrollers.

The firmware must be compiled using the `-Os` option, which opti-

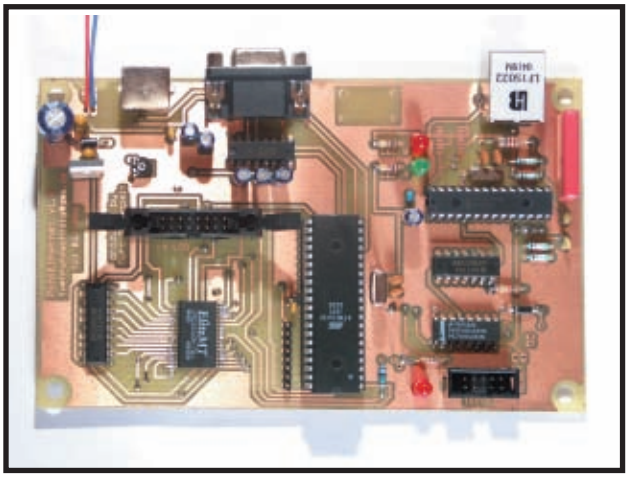


Photo 2—Take a look at the circuit board. To provide a better overview, I removed the cables normally connected to it.

mizes the code for size. This is important. Otherwise, the compiled code would be about 50% larger—too large to fit in the program memory. At the moment, the code (optimized) is 30 KB.

USER INTERFACE

The user interface consists of three control elements: menus, edit fields, and an elaborate text editor. A special screen (the Mail Menu) enables you to quickly browse through your mailbox.

After power-up, the system displays a greeting message. After a short

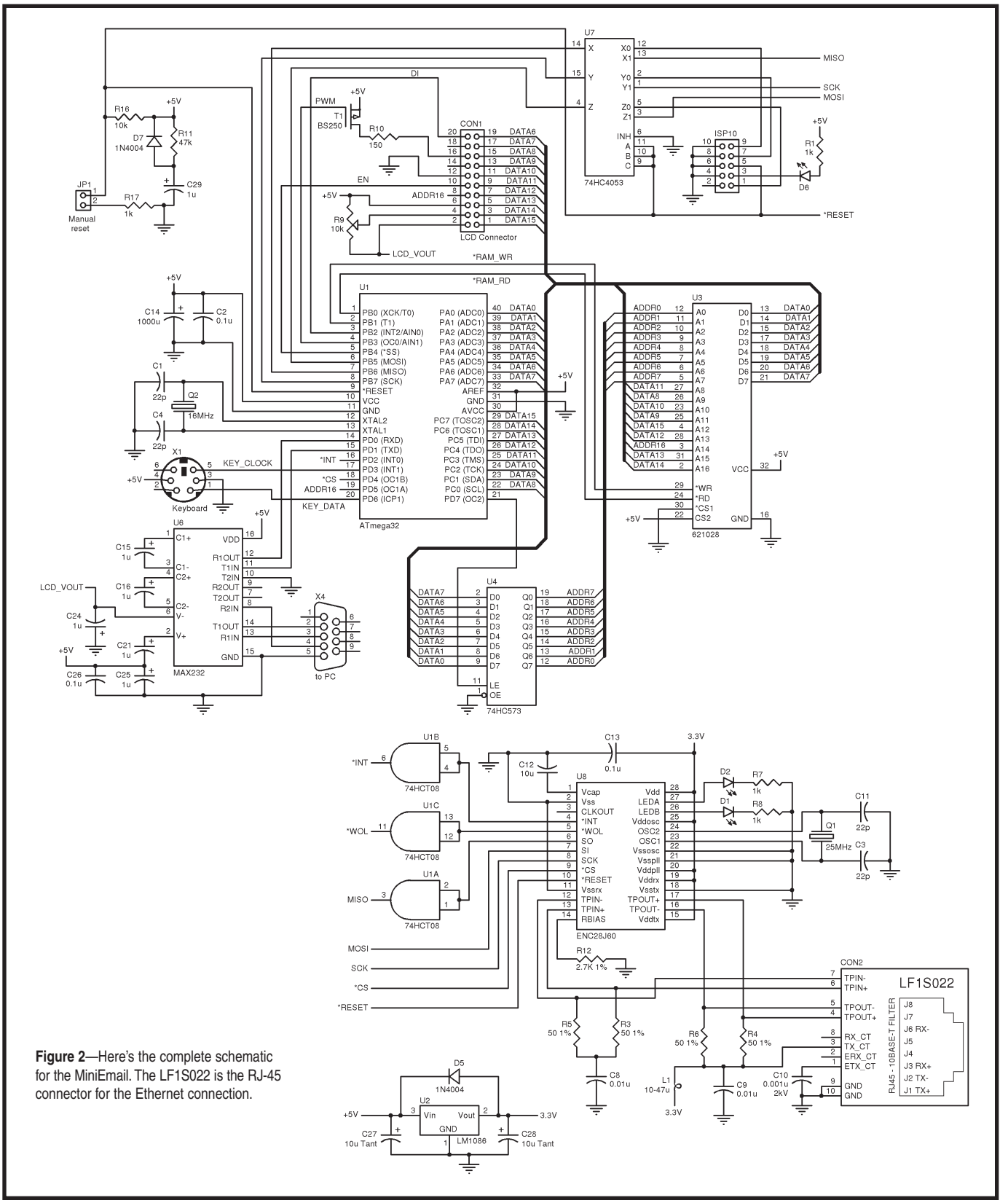
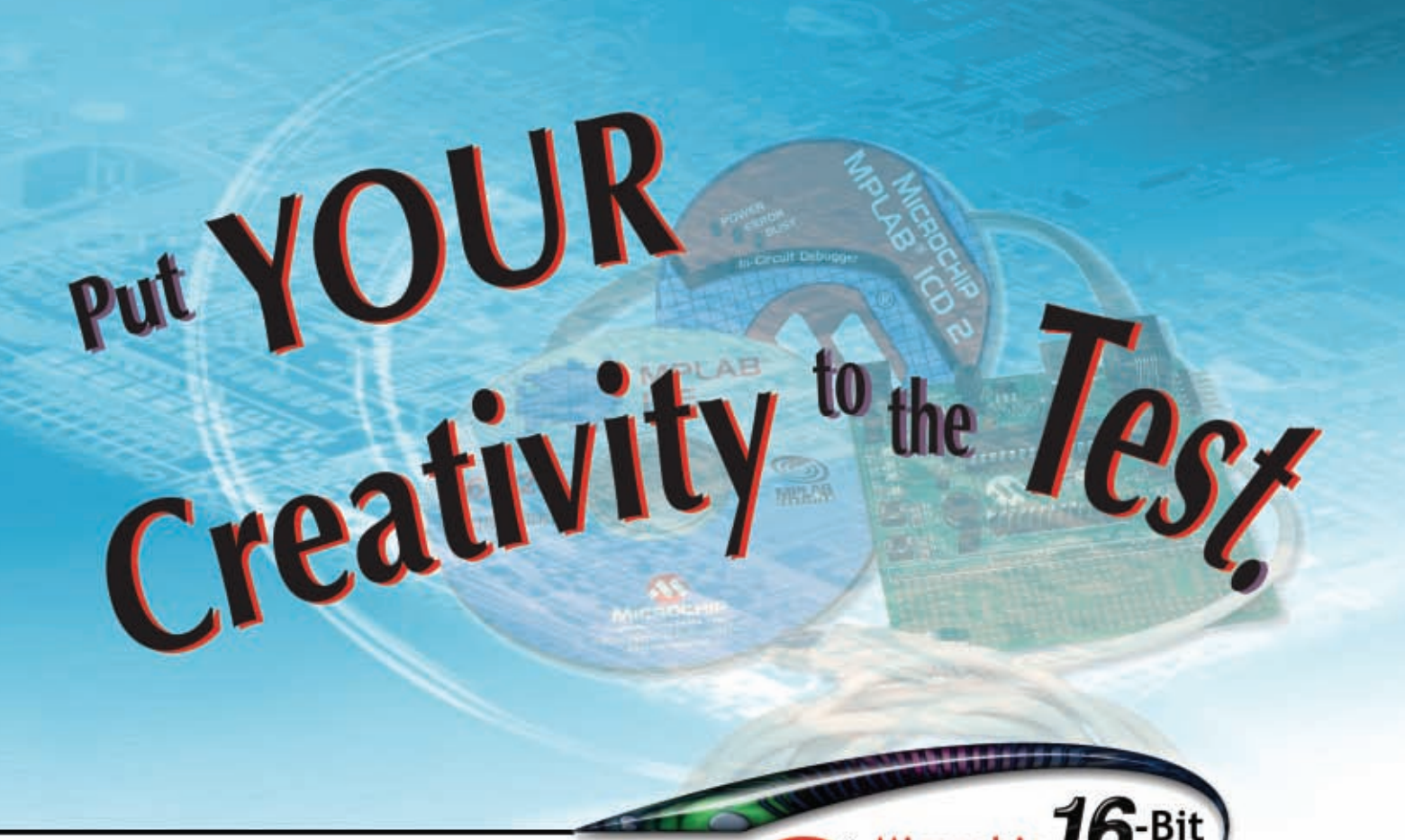


Figure 2—Here's the complete schematic for the MiniEmail. The LF1S022 is the RJ-45 connector for the Ethernet connection.



Circuit Cellar magazine and Microchip are pleased to bring you the Microchip 16-bit Embedded Control Design Contest.

Microchip 16-Bit Embedded Control Design Contest

This is the perfect opportunity to learn about Microchip's 16-bit Microcontroller and Digital Signal Controller (DSC) families. These powerful workhorses offer a broad range of products while preserving the compatibility that helps save you time and money on subsequent designs. Put your creativity to the test. Your project could win you a share of $15,000 in cash prizes and give you the international recognition you deserve.

Learn about unique contest sample opportunities, special offers from Microchip, and how you can be the next Circuit Cellar design contest success story.

For details, visit www.circuitcellar.com/microchip2007

CIRCUIT CELLAR®

MICROCHIP

Photo 3—This is a screenshot of MiniEmail's main menu. In the upper-right corner, a clock shows the current time, which is retrieved from the Internet. An arrow to the left of the menu items indicates the selected item.

while, the Main menu appears (see Photo 3). The Compose Mail, Check Mailbox, and Configuration submenus form a hierarchical menu structure (see Figure 3). When the other items listed beneath the respective menu titles in the diagram are activated (e.g., start the text editor), they enable you to input data, such as a username and password, or retrieve mail from the mail server. "Standby" is the only action that is accessible directly from the main menu. All other actions are grouped by function in the submenus.

The text editor is the part of the software that you will probably spend most of your time with. It is used for writing mail and reading it in Read-only mode, where no alteration of the text is possible. Therefore, it has to be as easy to use as possible. I implemented all the basic features you would normally take for granted in an ordinary text editor, but when you try to write such routines for the first time, it is much more involved than you'd expect.

The text editor supports the dynamic word-wrapping and scrolling of the text using the arrow keys. You can also use the Page Up and Page Down buttons if you want to browse through the text quickly. A scrollbar on the right border indicates the cursor's position in the text. You can choose between Overwrite and Insert modes, and there is a simple method to cut and paste text blocks.

During text editing, the entire LCD is used to display the text, so up to 168 characters can be shown at a time. That isn't much. One of the first items on my to-do list is adding a larger display.

WRITING MAIL

With respect to the firmware, sending mail is much easier than reading it, so let's first focus on the Compose Mail menu. The first item in the menu starts the text editor so you can enter the body of your letter. You then enter the recipient's mailing address and the subject of your e-mail, just like you would do when sending e-mail from your PC.

Additional fields, such as CC or BCC are not included, but since this requires only one more line in the header of the mail, it is not difficult. Your e-mail also needs a reply address, so the recipient knows who sent the mail. The reply address is normally the same for all of the messages you write. The text you enter in this edit field is stored in the ATmega32's EEPROM, so you don't have to type it every time you write a letter. After you select the last menu item, "Send" initiates the dispatch of the mail and displays a message that indicates whether or not it was successful.

CHECKING FOR MAIL

What makes this part more sophisticated is the ability to handle not only one e-mail at a time, but also fetch mail from the server. The system can determine which messages are new and which messages have been read. It can also extract data such as the sender, subject, or sent date from the header of the mail and then display the information.

The amount of mail the firmware can handle is limited by the size of the external SRAM. The maximum number of e-mails is currently 1,024. (If you've got more mail, you will be so busy answering it that you won't have time to build your own MiniEmail client—or you should delete some old mail). Note that 1,024 is the number of unique identifiers that the system can remember. The server assigns a unique identifier to each piece of mail. The system uses the identifiers to keep track of which letters are new on the server, which have already been read, and which have been marked for deletion.

All of the header data for all of the 1,024 messages cannot be held in SRAM at once; only the most recent (about 50) mail headers are held. When you want to browse through older e-mails, the firmware automatically reconnects to the server and fetches the headers of the next 50 e-mails.

When you select Check Mailbox in the main menu, you get to a submenu where you can retrieve and read mail. Before you can collect your mail, you must enter your username and password, which can be stored in EEPROM for your convenience. The firmware then retrieves the headers and displays the Mail Menu, where you can browse through your e-mail (see Photo 4). Apart from the size and the date, the first 42 characters of the subject and the mail sender are shown. In the first row, additional icons indicate (from left to right) whether a message is new, has been marked for deletion, or has been read. You can view the content of the selected message by pressing Return. When the mail is fetched from the server, it is prepared for viewing. The header and HTML tags, as well as long runs of the same character, are stripped from the mail and base64 decoding (used to encode 8-bit characters) is performed, so the content of the message is as readable as plain text. Binary attachments (e.g., images) can't be handled. Following this, the mail is viewed in the text editor (with editing disabled).

A similar action is performed when you press "r" in the Mail Menu. In that case, you can edit the text so you can add your reply. Leaving the text editor will bring you back to the Send Mail menu, where the reply address and subject will be filled in so your mail will be clear for take-off.

To delete a message, simply press D to mark it for deletion. When you leave the Mail Menu, you will be asked to confirm the deletion of the

Photo 4—This is the so-called Mail Menu, where you can browse through your mailbox. Data from the mail header (e.g., the size of the mail, its date and subject, and the sender) are displayed. The small icons in the upper right give additional information.

Compact. Versatile. Connected.

STELLARIS® LM3S6965 Ethernet Evaluation Kit

features the Stellaris® LM3S6965
ARM® Cortex™- M3-based microcontroller
with fully integrated 10/100 Ethernet (MAC+PHY).
The kit includes evaluation versions of popular software tools, spanning the design spectrum from evaluation...to prototyping... to application-specific design. Perfect for a variety of applications including connected motion, embedded web servers, network-based sensing, and building automation.

...just $79!

LUMINARY MICRO™

LuminaryMicro.com

selected message. If necessary, the firmware will reconnect to the server and delete the e-mails.

Additional actions are accessible from the Mail Menu. You can view the header of an e-mail ("h"), mark all the e-mail as read ("A"), and jump to the next new message ("n").

CONFIGURATION MENU

The Configuration menu enables you to change the settings for the Internet connection and view the version of the firmware you are using. Because you probably haven't memorized the IP address of your mail provider, you can enter the domain names (e.g., pop3.mailprovider.net) of the mail servers. The firmware will resolve the DNS to the IP addresses, which are needed for the SMTP and POP3 connections.

At the moment, my system has a hard-coded static IP because the DHCP routines, which allow clients to be assigned a dynamic IP by a DHCP server in the local network, did not fit into the flash memory. This is isn't the best scenario, but it suffices for normal home use because I typically use the same IP address.

STAND-BY MODE

The last item in the main menu is the stand-by option. In Stand-by mode, the LCD and its backlight are turned off and the MiniEmail checks for new mail on a regular basis. When you have new mail, a message is presented on the LCD and the LCD's backlight turns on again.

You can exit Standby mode by pressing the Escape key. Any other key displays a message telling you how many minutes you have left until the next automatic mail check is performed.

CODE BASICS

Now that you understand the user interface, let's take a closer look at the firmware. The backbone of the firmware is contained in the memain.c and global.h files, including the Main menu, parts of the user interface, and the interrupt handlers.

Before you start with the more involved routines handling the Internet connections, you must understand the code that controls the keyboard, the LCD, and the SRAM.

KEYBOARD INTERFACING

Atmel's application note entitled "AVR313: Interfacing the PC AT Keyboard" describes how to connect a standard PC AT keyboard to an ATmega32 using only two I/O pins, one external interrupt pin (INT1), and one general I/O pin. The keyboard interfacing in the MiniEmail system is based on the code example given in the application note, which allows the interrupt-based reception of bytes sent by the keyboard controller.

I extended the code with a third look-up table for recognizing extended scan codes of keys, such as home and the arrow keys, a parity check, and the possibility of sending bytes in the opposite direction (i.e., sending commands to the keyboard). This is needed to control the keyboard's status LED, which is not done by the keyboard controller itself. You can also use it to change keyboard parameters like the typematic rate, but at the moment I do not use this.

The decoded keystrokes are stored in a small buffer. You don't have to react to keystrokes immediately, but you can fetch the input from the keyboard when you are ready to do it using the `getchar()` function.

LCD & SRAM

The LCD is more or less accessed like RAM. Everything that is written to the RAM is immediately visible on the LCD. At the moment, the system uses a standard 64 × 128 dot-matrix graphic LCD. Because the LCD doesn't have a built-in character set, you have to provide your own, which is contained in chars.h.

The sram.c file contains the functions for writing to and reading from the 128 KB of external SRAM. In the external SRAM, you store (e.g., the text of incoming and outgoing mails) the IDs of the mails and the data extracted from the mail headers. The routines in lcd.c and sram.c are really straightforward, so I won't discuss them in detail.

IMPORTANT STUFF

The functions that provide Internet functionality are in two files, enc28j60.c and internet.c. The first file contains the low-level routines for the ENC28J60 and the TCP/IP stack, whereas internet.c contains the applications based on these protocols (i.e., the implementation of the functions for retrieving and sending mails, implementation of the POP3 and SMTP protocols, the Mail Menu, etc.).

If you have ever programmed a mail client on a PC, you probably know the post office protocol version 3 (POP3) and simple mail transfer protocol (SMTP), which are used to retrieve and send e-mail over the Internet. When writing a mail client on a PC, these two protocols are all you need to know because the operating system does the rest (and most) of the work for you. Unfortunately, there is no operating system on your ATmega32, so you have to build everything from scratch. As you can see in Figure 4, there are many protocols hidden beneath the surface.

Your hardware, namely the ENC28J60, enables you to send and receive frames over an Ethernet connection, which constitutes the physical layer. Everything on top has to be done by the firmware (i.e., you have to add the appropriate headers,

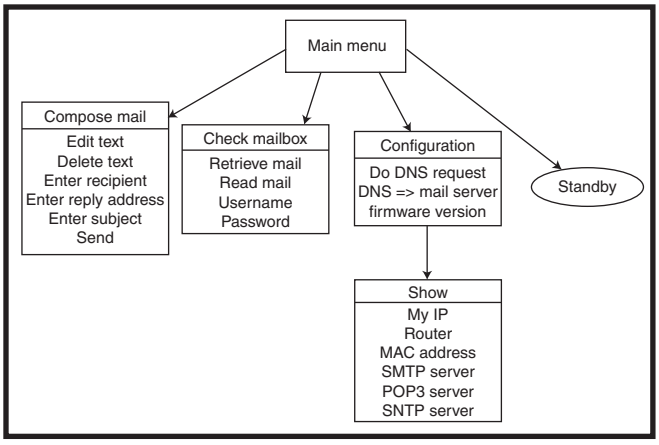


Figure 3—Study this diagram of the menu structure. Compose Mail, Check Mailbox, Configuration, and Show are submenus that are accessible from the Main menu. The Standby item in the Main menu switches the MiniEmail system to Standby mode.

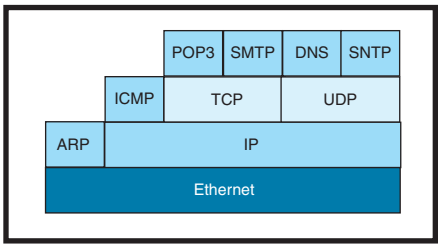


Figure 4—These are the protocols implemented in the MiniEmail system's firmware. The protocols are stacked on top of each other according to their dependencies. For instance, POP3 is based on TCP, which in turn enhances the functionality of IP, which in turn is equipped with the necessary Ethernet protocol header before it is finally sent over the physical connection.

calculate checksums, etc.). On the physical layer, network devices are addressed using their MAC addresses. Because you know only the domain names of the mail servers, you must first use the DNS protocol to translate the domain names to the corresponding IP addresses and then use the ARP protocol to translate the IP to the MAC address. Lastly, you can direct your packet.

Before you can send your data, you have to wrap it using either the TCP or UDP protocol. The transmission control protocol (TCP) provides a stable connection to a remote host so you can be sure every packet sent reaches its target. You need this as the basis for the SMTP and POP3 protocol you are aiming at.

The user datagram protocol (UDP) is much simpler and does not care if data gets lost or arrives in the wrong order, but you also need this, because DNS requests can be sent only via UDP. TCP packets, as well as UDP packets, are once again wrapped according to the Internet protocol and finally passed to the ENC28J60. Received packets have to be passed through all of the respective protocol handlers in reverse order.

There are two more protocols in Figure 4. The Internet control message protocol (ICMP) is part of the IP family. It is used for diagnostics purposes. The simple network time protocol (SNTP) is even more useful to the MiniEmail system because it is needed to set its internal clock. Fortunately, I was not the first to try to teach a microcontroller the language of the Internet. The Internet-related portion of the firmware is based on version 0.9 of Adam Dunkels's uIP (www.sics.se/~adam/uip/index.html). It's a TCP/IP stack for microcontrollers written in C and published under a BSD-style license. It contains the complete layer structure, from the physical layer up to some example applications. I adapted the routines for ARP, ICMP, UDP, IP, and TCP to work with my network device, the ENC28J60, and added the other aforementioned protocols. In order to use the 8 KB of SRAM in the ENC28J60, only the first part of received packets containing the protocol headers is loaded to the ATmega32's internal SRAM. After processing the header, and only if the packet has been accepted, the payload of the packet is passed to the respective application, thus avoiding the need for a dedicated buffer for incoming packets.

OUTLOOK

I hadn't imagined how many details would need to be considered when I started this project more than a year ago. It has been a very interesting and challenging project. It has also been a lot of fun.

The MiniEmail system provides all of the basics for communicating via e-mail, but such a project is never really finished. There are still dozens of items on my to-do list. Fortunately, the ATmega32 can be replaced with a new member of the AVR family, the Atmel ATmega644, which is pin-compatible to the ATmega32 and has twice the flash memory (and internal SRAM). That will provide enough space for many of my new ideas. I want to get rid of the static IP address, add CC and BCC fields, use a bigger display or a smaller (variable-width) font, improve the filtering and display of mail content and attachments, and add an address book (it would be best in combination with an additional external EEPROM with an SPI, such as the AT25256).

This project proves, rather impressively, that the ATmega32 and the ENC28J60 are a powerful combination. They can be used for many useful Internet applications. My e-mail client system is surely one of the most exciting. I can think of many other interesting possibilities. At the moment, my MiniEmail assembly serves as an online thermometer so I can check my room's temperature from anywhere in the world. Let me know if you have any other ideas.

Alexander Mann studies Physics at the Georg-August-Universität in Göttingen, Germany. His interests include electronics and programming. You can contact him at fuenfundachtzig@web.de. To make sure your e-mail gets past the spam filters, use the message form at www.cip.physik.uni-goettingen.de/~alemann/msgforme.php.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/204.

RESOURCES

Atmel Corp., "8-Bit AVR Microcontroller with 32K Bytes Insystem Programmable Flash: ATmega32," 2503J-AVR, 2006, www.atmel.com/dyn/resources/prod_documents/doc2503.pdf.

———, "AVR313: Interfacing the PC AT Keyboard," 1235B-AVR, 2002, www.atmel.com/dyn/resources/prod_documents/DOC1235.PDF.

Microchip Technology, Inc., "ENC28J60 Datasheet," DS39662B, 2006, ww1.microchip.com/downloads/en/DeviceDoc/39662b.pdf.

———, "ENC28J60 Rev. B4 Silicon Errata datasheet," DS80257A, 2006.

SOURCES

ATmega32 Microcontroller, ATmega644 microcontroller, and AT25256 EEPROM
Atmel Corp.
www.atmel.com

ENC28J60 Ethernet controller
Microchip Technology, Inc.
www.microchip.com

74HC4053 Analog multiplexer, 74HC573 latch, and 74HCT08 quad gate
NXP Semiconductors
www.nxp.com

Internet Password Manager

With this ATmega168-based password managing system, you can enter, display, and securely store all of your passwords and usernames. The handy device connects to your PC via a software-controlled USB interface.

I used to surf the Internet every day to look for the latest information about my favorite topics. Nowadays, more and more web pages rely on login schemes that prompt me to enter a previously registered username and a valid password. At first, because my memory is not particularly good at remembering long strings of numbers and letters, I used either the same login information for all my web accounts, or I used simple passwords that I could easily remember. Unfortunately, I learned that this was not the best solution, and I was putting valuable information at risk. To prevent those situations, I wanted an external device I could attach to my PC through a convenient interface that could securely store all pair combinations of usernames and passwords and keep them readily available. Although there are already several solutions on the market, they are mostly software-based and can be easily counterfeited with spyware or worms.

Photo 1—This is the Internet password manager in action. Just plug it into your PC's USB port, insert your smart card, and you will have Internet login information at your fingertips.

In this article, I'll describe how I designed an Internet password-manager (see Photo 1). The system can store many usernames and passwords in a Java Card smart card, which is one of the most secure methods of storing confidential information. Whenever a dialog box requests a username and password in an application or web browser, security features can be found on the externally attached device that will automatically complete the username and password fields for you, as if you were entering them.

JAVA CARD APPLET

In my article "Mobile Phone Book," I described the internal workings of smart cards (*Circuit Cellar* 190, 2006). I focused on the SIM smart card, which is used in many mobile phones. Other useful smart cards are available from manufactures such as Gemalto, Oberthur Card Systems, and Sagem Orga. Essentially, Java Card technology enables programs written in the Java programming language to run on smart cards and other resource-constrained devices. Applications written for the Java Card platform are referred to as "applets."

Java Card applets should not be confused with Java applets just because they are both called applets. A Java Card applet is a Java program that adheres to a set of conventions that allow it to run within the Java Card runtime environment. A Java Card applet is not intended to run within a browser. The name applet was chosen for Java Card applications because Java Card applets can be loaded into the Java Card runtime environment after the card has been manufactured. Unlike applications in many embedded systems, applets do not need to be burned into ROM during the manufacturing process. They can be dynamically downloaded onto the card at a later time.

Every applet is implemented by creating a subclass of the javacard.framework.applet class. The Java Card runtime environment (JCRE) invokes the methods Install, Select, Process, or Deselect. They are all defined in the base applet class when it wants to install, select, or deselect the applet or ask the applet to process an application protocol data unit (APDU) command.

The methods that any applet should have are listed below in the order in which they are invoked by the JCRE during applet creation and execution. The Install method is a static method used to create an instance of the applet subclass. The Register method is used by the applet to register the applet with the JCRE and assign the default AID to the applet instance. The Select method enables you to inform the applet that it has been selected. The Process method enables you to instruct the applet to process an incoming APDU command. And

finally, the Deselect method is used to inform the currently selected applet that another (or the same) applet will be selected (see Figure 1).

The JCRE calls the Install method to create an applet instance. The applet instance is registered with the JCRE using the register method.

When receiving a SELECT APDU, the JCRE first checks whether an applet is already selected. If one is selected, the JCRE deselects the current applet using the deselect method. In the Deselect method, the applet performs any cleanup or bookkeeping work before it is inactive. The JCRE then selects the new applet with the Select method. The applet performs any initialization necessary in the Select method.

After a successful selection, each APDU (including the SELECT APDU) is delivered to the active applet via a call to its Process method. The Process method is an essential method in the applet class. It processes APDU commands and provides an applet's functions (see Table 1).

The Install, Select, Deselect, and Process methods are applet-entry-point methods. They are invoked by the JCRE at the appropriate state of applet creation and execution. The base applet class provides only the default behavior for these methods. An applet needs to override some or all of these methods to implement its functions.

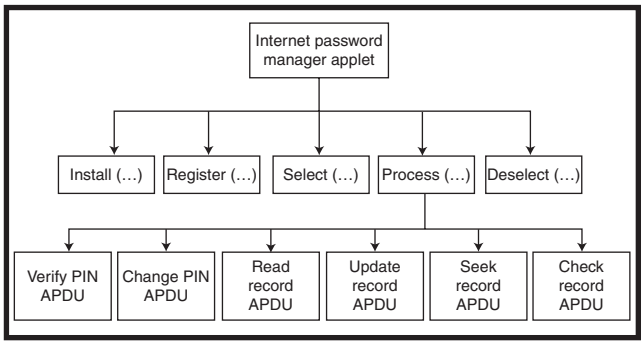


Figure 1—The applet running inside the Java Card is developed using a subset of the Java programming language from Sun Microsystems. The development tools are freely available from the Sun web site (www.sun.com).

PS2 OR USB?

I first thought about using the PS2 interface to connect the device to the computer because it is easier to use than the USB port from the developer's standpoint. However, the brand new computer models that are shipping today lack this interface. In fact, the PS2 port is found only on legacy computers. So, it was clear that if I wanted to use this device in every computer, I needed to use the ubiquitous USB interface.

At this point, I had several options for implementing the USB interface on my password-manager device. On one hand, I could use one of the many commercially available chips that deal with the USB port as if I were dealing with a standard RS-232 port. On the other hand, I could choose an advanced microcontroller with built-in USB support. Both options had the disadvantage of increased complexity, and therefore an increased price.

I went with a USB implementation into a low-cost microcontroller through emulating the USB protocol in the microcontroller's firmware. The main design challenge was obtaining sufficient speed. The USB bus is quite fast with a low speed of 1.5 Mbps, a full speed of 12 Mbps, and a high speed of 480 Mbps. The Atmel ATmega168 microcontroller is fully capable of meeting the speed requirements of low-speed USB only.

Method	Description
VerifyPIN (…)	This method checks the PIN.
ChangePIN (…)	This method allows the PIN of the smart card to be changed.
ReadRecord (…)	This method allows the retrieval of the specified record number.
UpdateRecord (…)	This method allows the specified record number to be updated.
SeekRecord (…)	This method allows the search of a specific record, either forward or backward, from the specified record number.
CheckRecord (…)	This method allows the search of the first nonblank record, either forward or backward, from the specified record number.

Table 1—These are the various methods used to implement the APDU commands that the Java Card can understand.

I decided against developing my own soft-USB solution and integrating one of the proven and freely available solutions for the ATmega168. I am aware of at least two possible implementations of the software USB peripheral. One, provided by Igor Cesko, is available as an application note on Atmel's web site. Another solution was written by Christian Starkjohann of Objective Development Software. I chose the Objective Development Software approach because of all the customizable code that was written in ANSI-C, which makes it easier to maintain. The modular concept made for easy integration into my existing design, the performance was very good, the standards conformance was well documented with a description of limitations and potential problems, and it was freely available with an open-source license.

ALPHANUMERIC KEYPAD

When you work with passwords, you cannot limit yourself to numeric characters. You need to support alphanumeric characters to make it harder for the attacker to guess the password by means of brute-force attacks. Once again, there are multiple possibilities to enter alphanumerical information. One solution is to use the PC keyboard, but due to the fact that you are dealing with sensitive data, the approach is not recommended to avoid interference from any spyware or virus software that could catch the confidential information during the updating process.

From a security standpoint, the best solution is an isolated keyboard. But, how can you implement an alphanumeric keyboard using just a numeric keypad? Make the keypad work exactly like the keypad found on any cellular phone. Every numeric key is assigned a subset of possible characters to display depending on the number of key presses sensed within a time frame (see Table 2).

It is possible to enter lowercase letters and uppercase letters. Capital let-

Keypad key number	First press	Second press	Third press	Fourth press	Fifth press	Sixth press	Seventh press	Eighth press	Ninth press
0	@	0	#	_	&	$	.	(	)
1	space	1	*	+	-	/	?	!	%
2	a	b	c	2	A	B	C	—	—
3	d	e	f	3	D	E	F	—	—
4	g	h	i	4	G	H	I	—	—
5	j	k	l	5	J	K	L	—	—
6	m	n	o	6	M	N	O	—	—
7	p	q	r	s	7	P	Q	R	S
8	t	u	v	8	T	U	V	—	—
9	w	x	y	z	9	W	X	Y	Z

Table 2—Depending on the key presses, it is possible to enter alphanumerical data and special characters using a standard numeric keypad. There are up to nine levels.

ters are entered by pressing the key several times, until the first uppercase letter appears on the display. By default, lowercase letters are entered first, since most passwords are case-sensitive and small letters are used more frequently. Figure 2 shows the algorithm used by the firmware to enter alphanumeric data.

PASSWORDS AT WORK

When you plug the Internet password manager system into a PC's USB port, a welcome message is displayed on the LCD for several seconds. After that, the display prompts you to insert a card (see Figure 3).

When you insert a smart card into the slot, the password manager automatically requests the verification of the card's PIN to prove that only you can use your card. The PIN is four digits long and entered using the keypad's numerical keys. You have three attempts to verify the PIN. After the third unsuccessful attempt, the PIN is locked and it is necessary to unlock the PIN with the PUK (PIN unlock key). If PIN verification is successful, the password manager device displays the main menu.

Upon PIN verification, the system displays the main menu, so that the function keys on the keypad (A, B, C, and D) are available for you to select an option, such as browsing through all the IDs stored in the card, finding a specific ID, adding or deleting an ID, and performing card maintenance (e.g., changing the PIN of the card or making a backup of all your stored records to a PC through the USB connection with the help of a text editor).

After pressing the function key labeled "A" (also referred to as "View menu"), the LCD shows the first ID stored in the smart card, if available. The username and password are not shown on the LCD for security reasons. You can browse the next and previous records by pressing the A and B function keys, respectively. Once the selected ID is shown, it is possible to send the username and the password to the requesting dialog box by pressing the pound key (#).

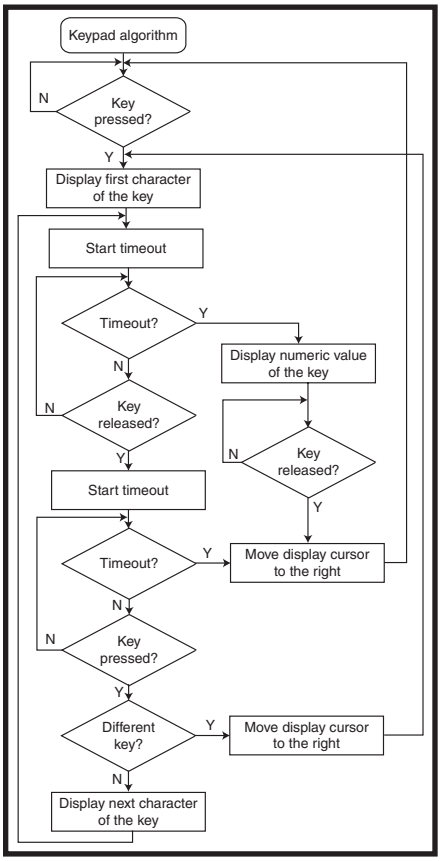


Figure 2—The algorithm used to enter alphanumerical data is a replica of the behavior found on the keypads of many mobile phones. It allows you to enter lowercase, uppercase, and special characters within a very intuitive user interface.

Pressing the function key labeled "B" (also referred to as "Find menu") enables you to search for a specific ID in the card. It is not necessary to write the full ID name. Typing the first characters is enough.

Pressing the function key labeled "C" (also referred to as "Edit menu") brings up a submenu that enables you to enter a new ID in the card or delete the previously stored IDs. By pressing the "A" key (if there is space available in the smart card), the device will ask for the new ID label, username, and password. It will then save the new record, which will be the default-selected ID, the next time you browse through the different IDs. On the other hand, by pressing the "B" key, you can delete a complete record, but it is necessary to write the full ID to ensure that accidental deletions are prevented.

Pressing the function key labeled "D" (also referred to as "Card menu") brings up another submenu that allows the card's PIN to be changed or makes a backup of all the records stored. By pressing "A" again, the PIN in the smart card can be changed. To do this, the password manager device asks for the current PIN and the new PIN (the PIN is four digits long). The PIN is changed only if the verification of the current PIN is successful; otherwise, the PIN remains unchanged. However, by pressing "B," it is possible to backup all the records. The device waits for another key press to

start sending the complete list through the USB. Because the password manager is emulating a standard keyboard, it is possible to store the received data with the help of a text editor (without a special application).

You can go back to the main menu at any time by pressing the asterisk key (*). Additionally, if you withdraw the card during operation, the device will start the authentication process again, prompting you to insert the card and verify the PIN again. So, it is possible to have more than one active smart card and make changes on the fly without restarting the system. This enables you to have one smart card for all passwords used in the office and at home.

HARDWARE DESIGN

The hardware design is built around the ATmega168 (see Photo 2). Its platform has the small footprint required for this application. Because of the low power consumption, it is possible to power the design through the USB port, so it is not necessary to use an external power supply or a battery.

The smart card interface is a special socket needed to interface the smart card to the ATmega168's USART0 (see Figure 4). It uses a 4,9152-MHz oscillator to get a communication data rate of 12,711 bps on the I/O line of the smart card. There is a strange data rate because the smart card communicates at 9,600 bps when it is clocked at 3.57 MHz. Even though many UARTs cannot gen-

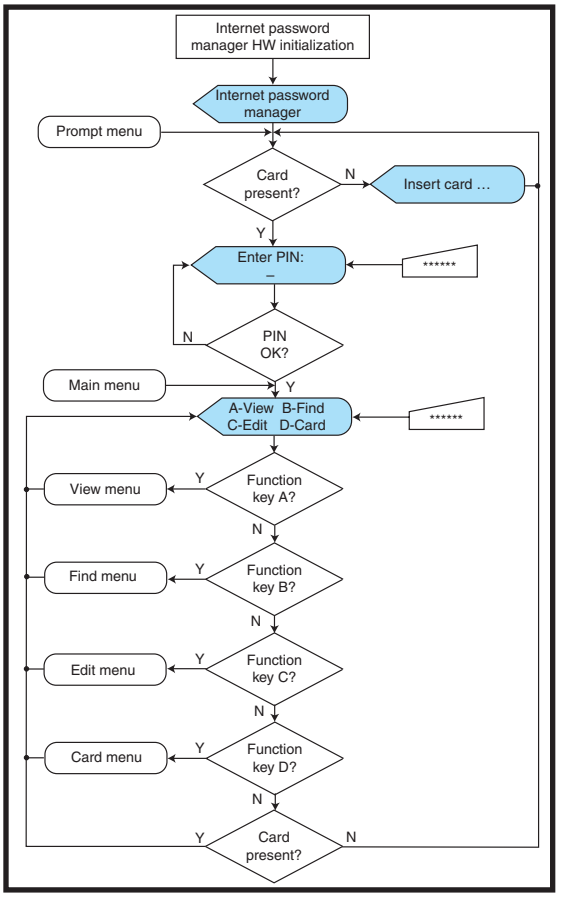


Figure 3—Here is a flowchart of the main menu showing the different options available once the PIN of the Java Card has been successfully verified. If the card is removed at any time, it is necessary to proceed again with a user authentication.

erate strange data rates, this is not a problem for the ATmega168, which can generate a stable communication speed, programming the data rate generator accordingly.

The LCD is a standard 2 × 16 display compatible with the industry-standard Hitachi LCD controller. It is connected to the ATmega168 with a 4-bit bus configuration to save I/O lines.

The 16-key multiplexed keypad is used to enter numerical and alphanumerical data when requested by the device. In order to save I/O lines, the keypad is connected to the ATmega168's analog input ADC through a resistor ladder. The resistor network values are chosen so that the impedance drop is proportionally distributed among all keys in increments of 500 Ω. To ensure good keypad reliability, choose all the resistors as precision resistors of 1% tolerance. Even though the voltage drop is not proportionally distributed between all the keys, thanks to the ADC's 12 bits of resolution, it is possible to distinguish all the keys, simplifying the keypad interface.

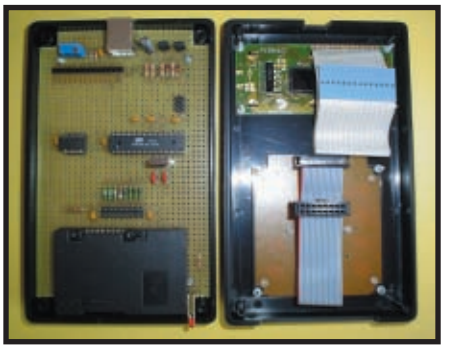


Photo 2—A close-up view of the Internet password manager's internal circuitry. The hardware is so simple because all the complexity dealing with the display, keypad, smart card, and USB interface has been moved to the software side.

The USB interface is accomplished through software emulation. USB lines DATA+ and DATA- must be wired to the same I/O port. Line DATA- must be wired to bit 0. DATA+ must also be connected to INT0. DATA- requires a pull-up of 2.2 kΩ to 5 V to be identified as a low-speed USB device. DATA+ is used as an interrupt source instead of DATA- because it does not trigger keep-alive and reset states. No UART, timer, input-capture unit, or other special hardware is required (except one edge-triggered interrupt).

Additionally, the ATmega168 must be clocked at 12 MHz, so it is necessary to specify the speed grade of the microcontroller when ordering it. It is also possible to use over clocking with low-speed AVR devices, but it is not recommended since there are high-speed components available at a reasonable price. There is an activity LED to signal data transmission on the I/O line of the smart card. Finally, there is an in-system programming port (ISP) to update flash memory if additional updates or enhancements are planned in the future.

SOFT-USB

The USB's physical interface consists of four wires: two for powering the external device (V_{CC} and GND) and two signal wires (DATA+ and DATA-). The power wires give approximately 5 V and a maximum of 500 mA. The ATmega168 can be supplied from the V_{CC} and GND. The signal wires named

Control Your Remote with Low-Cost 16-bit MCUs

Whether your design is a home theater remote or any other embedded application, the PIC24F delivers the peripherals, performance, development tools and software you need.

With 16 MIPS performance and an extensive peripheral set, Microchip's PIC24F microcontrollers are highly cost-effective solutions.

GET STARTED IN 3 EASY STEPS!

1. **Learn all about our 16-bit products in only 20 minutes!** Our *FREE* 16-bit web seminars highlight our 16-bit architecture and comprehensive support suite that includes low-cost development tools, a variety of software libraries (many free), application notes, reference designs and more!
2. **Take advantage of *FREE* resources for download!** Our MPLAB® IDE Integrated Development Environment and a full-featured trial of the MPLAB C30 C Compiler from our web site gets you up and running fast!
3. **Get hands-on technical training for *as low as* $49!** Attend a half- or full-day 16-bit course at a Microchip Regional Training Center to take your designs to the next level! Attendees also receive deep discounts on related development tools!

Purchase and program your 16-bit PIC24F devices and related development tools at... microchip DIRECT www.microchipdirect.com

Device	Pins	Flash (KB)	PIC24F Family Features
PIC24FJ64GA006	64	64	8 KB RAM
PIC24FJ64GA008	80	64	Parallel Master Port
PIC24FJ64GA010	100	64	5 - 16-bit Timers
PIC24FJ96GA006	64	96	5 - Output Compare/PWM
PIC24FJ96GA008	80	96	5 - Input Captures
PIC24FJ96GA010	100	96	Real-Tme Clock Calendar
PIC24FJ128GA006	64	128	2 - UART with IrDA® and LIN Protocols
PIC24FJ128GA008	80	128	2 - SPI, 2 - I²C™
PIC24FJ128GA010	100	128	10-bit ADC, 16 Channels
			2 Analog Comparators

Visit our web site for more information about additional 16-bit devices with higher performance and added features like DSP and enter our 16-bit Embedded Design Contest today!

Microchip 16-Bit Embedded Control Design Contest

MICROCHIP

www.microchip.com/PIC24

Microcontrollers • Digital Signal Controllers • Analog • Serial EEPROMs

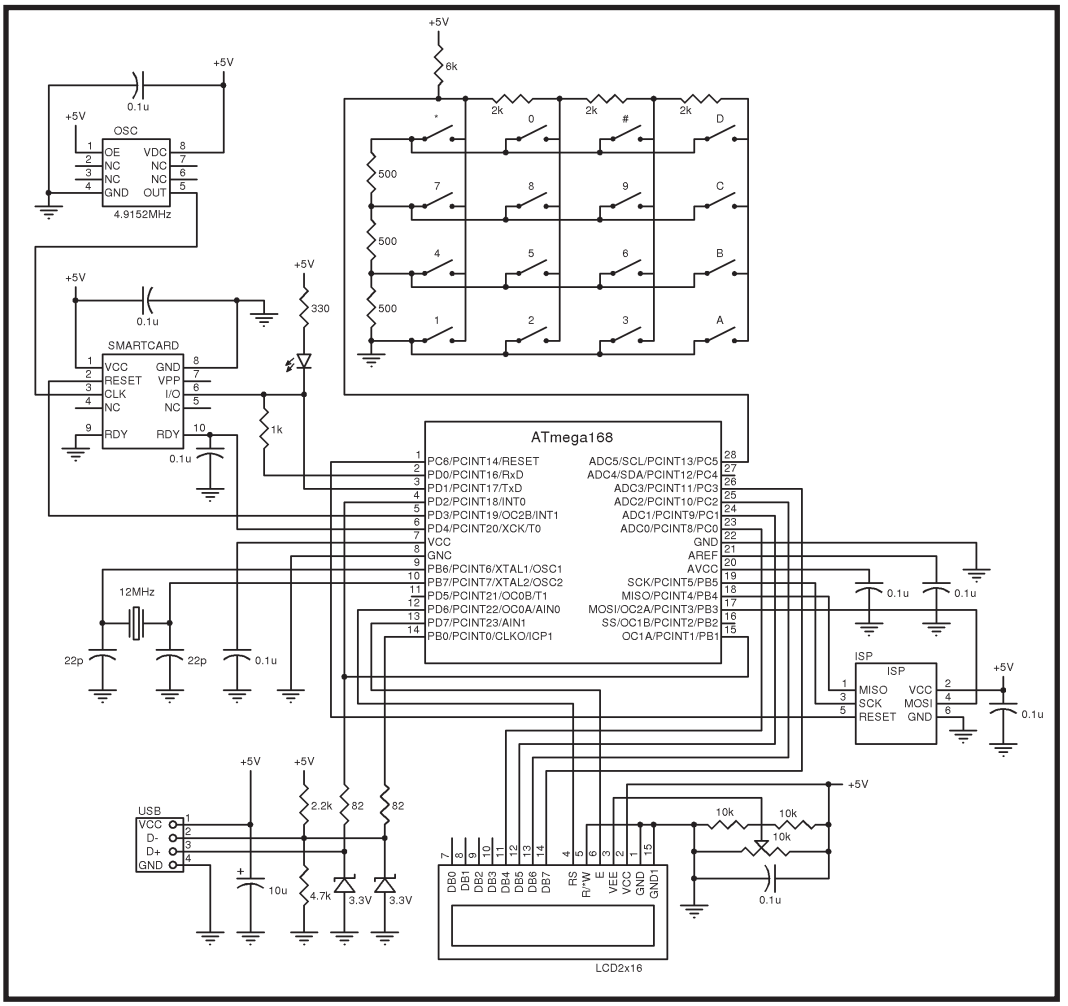


Figure 4—Here is a schematic of the main board based on the ATmega168 microcontroller. All of the microcontroller's resources are used to minimize the number of ICs present in the design.

DATA+ and DATA- handle the communication between the computer and the device. Signals on the wires are bidirectional. Voltage levels are differential. When DATA+ is at a high level, DATA- is at a low level. But there are some cases in which DATA+ and DATA- are at the same level, like end of packet (EOP). Therefore, in the firmware-driven USB implementation, it is necessary to be able to sense or drive both signals.

According to the USB standard, the signal wires must be driven high between 3 and 3.6 V, while the V_{CC} supported by the USB host is 4.4 to 5.25 V. So, if the microcontroller is powered directly from the USB lines, then the data lines must pass through a level converter to compensate for the different voltage levels, the reason for the Zener diodes at 3.3 V in the data lines.

For a USB 1.1-compatible low-speed device firmware implementation, a 1.5-Mbps bit stream must be decoded. For the ATmega168 clocked at 12 MHz, eight CPU cycles are for each bit. As a RISC processor, the AVR executes most instructions in one clock cycle. This gives the firmware about eight instructions to do the following operations on each bit.

In NRZI decoding, a "1" is encoded as no change of the data lines. A "0" is encoded as a change. Decoding can be done by a negative exclusive OR operation between the current status and the previous one (eight cycles earlier).

Bitstuff decoding is performed to preserve synchronization during long sequences of "1." A "0" (change of data lines) is inserted every six consecutive "1" bits. This "stuffed" bit must be removed during reception.

In end of packet recognition, the end of a packet is notified by an "SE0" condition. This means that both data lines (which are normally the inverse of each other) are at a logical "0" level for two bit times. The received byte also must be stored and a buffer overflow check must be performed every 8 data bits.

HID

Implementing a USB device that conforms to the human interface device (HID) specification is particularly useful if you develop for PC computers. Every vendor-class device requires a kernel driver, but the HID class driver is built into Windows. This means that you do not need an installer or a custom driver. Since Internet password manager devices try to emulate the keystrokes of a standard keyboard, you don't need special software on the host computer.

The main advantage of a HID-compliant device is you don't need a custom kernel driver on Windows. This means that you can start the controlling software application without installation by an administrator. Since other operating systems also include a kernel driver for USB HID devices, this advantage is not just for Windows. It allows the use of an Internet password manager in other environments. The main disadvantage is the added complexity because you need to understand how report descriptors are built and you must conform to the framework defined by the HID specification.

Writing your own driver gives you much more flexibility and less complexity, but you must install a kernel driver on Windows. For more information, visit www.usb.org/developers/hid page. The idea behind the HID specification is to define a wide range of sensors, buttons, and lights that you might want to connect to an application pro-

DENSER CODE
better performance

FOR RELEASE THIS MONTH

PROforPSoC®
with OMNISCIENT CODE GENERATION™

HI-TECH PRO for PSoC ANSI C COMPILER* - from the company that has, for over two decades, delivered the industry's most reliable embedded software development tools and compilers - comes with:

OMNISCIENT CODE GENERATION

Extracts information from multiple source files simultaneously, allowing more intelligent state-of-the-art code generation that:

- Optimizes the size of each pointer variable in your code based on its usage;
- Eliminates contention for PSoC index register;
- Eliminates the need for many non-standard C qualifiers and compiler options;
- Produces more efficient interrupt context switching code; and
- Customizes the functionality of the printf library function.

** Can run on multiple platforms including **Windows, Linux and Mac OS X**.*

**Integrates into PSoC Designer™ IDE and other PSoC development tools.*

DENSER CODE, BETTER PERFORMANCE

Code compiled with HI-TECH PRO for PSoC can deliver over 40% denser code than its competitor.

FREE 45 DAY EVALUATION

A fully functional 45 day trial version of HI-TECH PRO for PSoC will be available, free of charge from HI-TECH's website. Order your demo at www.htsoft.com/portal/ccpsoc.

PRICE AND AVAILABILITY

HI-TECH PRO for PSoC includes, at no extra cost, HI-TECH PRIORITY ACCESS™ - 12 months access to updates and technical support and HI-TECH SATISFACTION GUARANTEE™ - a hassle free 30 day money back guarantee.

CYPRESS PERFORM

HI-TECH SOFTWARE

HI-TECH Software LLC
6600 Silacci Way Gilroy, CA 95020 USA
Ph: 800 735 5715 **Web:** http://www.htsoft.com

gram. The USB implementer forum's universal serial bus HID usage tables specify hundreds of usages.

HID does not necessarily mean keyboards, mice, and joysticks. Any device that requires only a moderate transfer rate can be designed as a HID.

The HID deal is a two-way bargain. Both the HID device and the Windows application must agree to use particular codes in the HID specification. Like most complex specifications, the HID documentation is rough going. You need to understand only about 1% of the specification to make your device work. The rub is, which 1%? The trick to implementing a HID is not understanding everything in the HID specification, but understanding exactly which subset of the specification you need and which is actually supported by Windows. You will not find the latter in the HID specification itself. You need to check Microsoft's documentation.

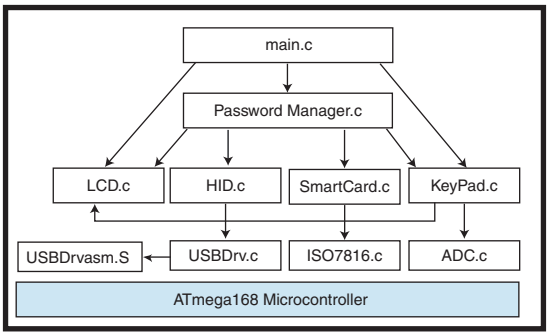


Figure 5—The module hierarchy in the software design is fairly simple. There are different layers of complexity. Each microcontroller peripheral has its own driver in a separate source file.

A HID-compliant peripheral communicates with Windows by sending reports. The main HID design task is to invent a report format that conveys the intent of your control to the operating system. The HID report descriptor was built using the HID_tool program (available at www.usb.org). Do not even think about trying to figure out the actual hex values. The tools do it for you. The system's firmware returns this table to the host when the host issues a Get_Descriptor-Report request.

I use a simplified keyboard report descriptor, which does not support the boot protocol. In addition, it does not allow setting status LEDs, and only one simultaneous key press (except modifiers) is allowed. Therefore, I can use short 2-byte input reports.

The report descriptor was created with usb.org's HID Descriptor Tool. (It can be downloaded from www.usb.org/developers/hidpage.) Redundant entries (i.e., LOGICAL_MINIMUM and USAGE_PAGE) have been omitted for the second INPUT item.

FIRMWARE DESIGN

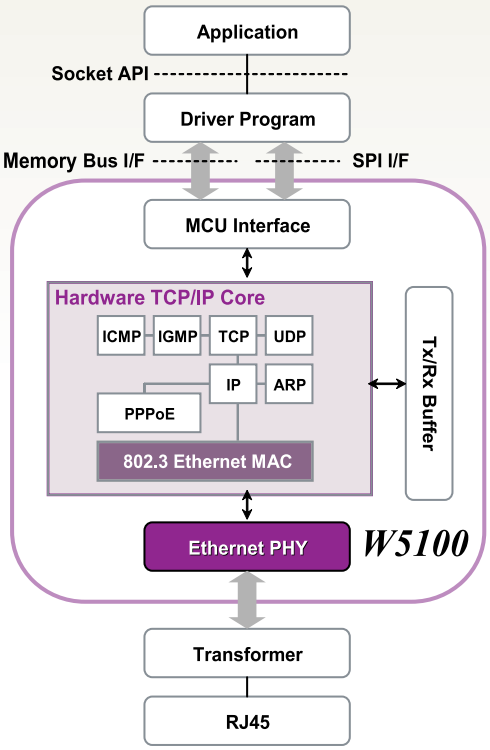
For development tools, I developed all the firmware with the help of Atmel AVR Studio 4 with Service Pack 2. Unfortunately, the C compiler is not included as a standard feature. However, it has support built in for the open-source avr-gcc compiler. The avr-gcc compiler can be configured easily with the help of the WinAVR 20050214 distribution package. To download the WinAVR distribution package for Win-

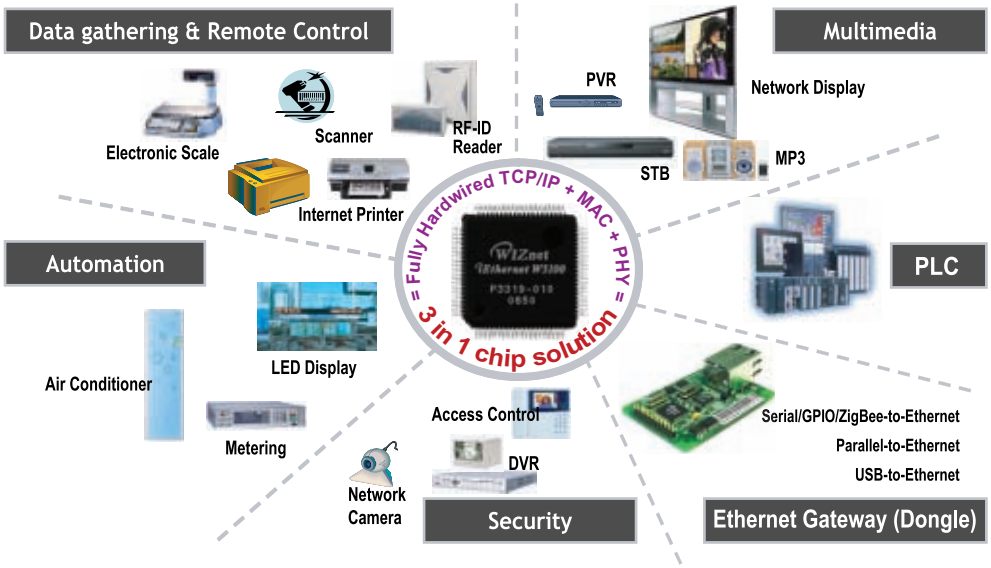
Born to What a Simple Internet!

WIZnet released new network modules with W5100

Anyone can implement Ethernet function by adding the WIZ module series without changing existing MCU of embedded system.

Serial-to-Ethernet	 *WIZ*100SR	- Support serial speed max. 230Kbps - 10/100Mbps Ethernet and auto-MDIX - Provide *Configuration Tool program to be installed in the PC (*With this program, users can configure all parameters of modules and control them through Ethernet.) - Plug-in type module having pin header (12pins) - Size : 50mmx30mmx12mm (W x L x H)
	 *WIZ*110SR	- Spec is same with WIZ100SR - In addition, Include the DB9 and RJ45 I/F for user's convenience
USB-to-Ethernet	 *WIZ*200USB	- Support USB2.0 Full Speed & OTG
All-in-One Connector	 *WIZ*800MJ	- One-stop solution for internet connectivity - Ethernet controller, Transformer, Mag Jack and TCP/IP stack are included - Support memory bus and SPI for MCU I/F





WIZnet

www.wiznet.co.kr

dows, visit http://winavr.sourceforge.net/download.html.

The firmware for the Internet password manager requires the installation of avr-gcc and avr-libc (a C-library for the ATmega168). Before downloading the firmware, it is necessary to configure the device for an external crystal by programming the device fuses accordingly.

All the software has been developed in a modular and hierarchical way in C language so maintenance and improvements are fairly simple to do (see Figure 5). A description of the modules and the functions included in every module is posted on the *Circuit Cellar* FTP site.

PROTECT YOURSELF

The need for Internet security has increased dramatically due to a variety of fraudulent activities, such as identity theft, phishing attacks, and so on. My handy device allows you to protect valuable information, and it gives you the security needed to perform transactions online successfully.

You have learned how to enter alphanumeric information with a multiplexed keypad using the microcontroller's ADC with a sophisticated algorithm to enter alphanumeric data like you do on your cellular phone. You have also learned how to exchange information using a modern USB interface, without dedicated hardware, based only on the raw performance of the ATmega168.

Carlos Cossio (ccossio@hotmail.com) has a B.S. in Physics from the University of Cantabria in Spain. He is a senior embedded systems engineer with more than 10 years of experience designing trusted security platforms. In his spare time, Carlos enjoys repairing and bringing old electronic equipment to life.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/204.

RESOURCES

Atmel Corp., "AVR309: Software Universal Serial Bus," 2006, www.atmel.com/dyn/resources/prod_documents/doc2556.pdf.

———, "ATmega 48/88/168," 2007, www.atmel.com/dyn/resources/prod_documents/doc2545.pdf.

Z. Chen., *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*, Prentice Hall, 2000.

C. Peacock, "USB in a Nutshell: Making Sense of the USB Standard," 2007, www.beyondlogic.org.

C. Starkjohann, "Objective Development, AVR-USB: A Firmware-Only USB Driver for Atmel AVR Microcontrollers," 2006, www.obdev.at/products/avrusb.

SOURCE

ATmega168 Microcontroller
Atmel Corp.
www.atmel.com

GX typhoon

infinite possibilities

:: Expansion port for optional fan control
:: MCE support for visual feedback
:: Simple USB connection with all software
:: Display system information
:: Integrated remote

WWW.MATRIXORBITAL.COM

MATRIX ORBITAL

WIN BIG EVERY WEEK AT JAMECO

Every week, Jameco is giving extra discounts on major brands like these:

- AlcoSwitch
- AMP/Tyco
- Dallas Semiconductor
- Intersil
- Maxim
- Microchip
- Molex
- Panasonic
- AVX
- Fairchild Semiconductor
- Tyco
- Aromat
- Aavid Thermalloy
- Atmel
- ST Micro
- T.I. Semiconductor
- Augat/Tyco
- Bourns
- CTS
- Cypress
- C&K Switches
- Comair Rotron
- Condor
- Grayhill
- Intel
- Micron
- Philips
- Power-One
- Raychem/Tyco
- Renesas
- SanDisk
- Toshiba
- Vishay Siliconix

TAKE AN EXTRA 15% OFF ON 2 NEW MAJOR BRANDS EVERY WEEK!

And that's in addition to the industry-best pricing you already get at Jameco every day!

You already know about Jameco's best-in-the-business low pricing...

Now we're out to get a *new* message across: *Major Brands–at Jameco pricing!*

To dramatize the point, we're giving you an *additional* 15% discount on two different major brands each week. From now through the end of May!

SEE WHO'S UP THIS WEEK GO TO THE WEBSITE

And that's on *top* of our everyday discounted pricing!

We wanted to make this so good a deal that you just *couldn't* pass it up.

So... Want to know which two major brands you can save extra-big on *this* week? Just head to the website...

www.Jameco.com/CCW

Great Products. Awesome Prices.

JAMECO®

by Albert Tran, Andrew Kwan, Kevin Brown, & Jason Vangilst

Motion Tracking and Analysis System

This group of designers combined their passion for billiards and embedded systems to design a pool tutorial system. The system tracks the motion of the balls on a pool table and provides visual shot suggestions and predictions.

If you have a group of guys who are really keen on billiards and have a full load of engineering courses in the final year of their engineering program, you have a big-time conflict. To resolve it, you have three choices: spend all your time studying and never have a life, spend all your time shooting pool and never get to have a career, or do what we did and persuade a professor that a suitable design project should combine your passion for billiards and interest in embedded systems.

We developed a pool tutorial system for beginners. The embedded system tracks the motion of billiard balls on a pool table and provides visual shot suggestions to the user. Additional information, such as ball trajectory and path history, are also provided.

In this article, we will explain how we designed and built the system. In addition, we'll describe the basic things you need to understand before moving on to the required video processing. We will detail the image-processing algorithms we developed to run the system with the limited memory on an Analog Devices Blackfin ADSP-BF561 processor, the issues we had developing the system, and possible upgrades.

The principle behind billiards is quite simple: sink balls into the pockets. However, mastering the game (learning the ball physics and trajectories) can take years of practice. Our system reduces the learning curve for novice pool players by providing visual feedback. Thus, our pool tutorial system is an invaluable tool for beginners.

SYSTEM OVERVIEW

To realize our design goals, we needed a powerful digital signal processor (DSP) capable of handling complex real-time video algorithms and integrating a wide array of peripherals. Dr. Mike Smith, our project supervisor at the University of Calgary, gave us an Analog Devices ADSP-BF561 Blackfin EZ-KIT Lite. Targeting consumer, multimedia, and DSP applications, this evaluation kit incorporates onboard audio/video encoders, audio/video decoders, and 64 MB of SDRAM around a Blackfin high-performance dual-core DSP.

The ADSP-BF561 contains two processing cores in a single package (essentially two ADSP-BF533 DSPs), each achieving clock speeds of up to 600 MHz. The BF561 has a wide range of onboard features, including two parallel peripheral interfaces (PPIs) supporting the ITU-R 656 video format, two serial ports (SPORT), a UART controller, 12 timers, 48 general-purpose I/Os, and two 16-channel DMA controllers. About 100 KB of full-speed L1 SRAM (32 KB of instruction, 64 KB of data, and 4 KB of scratch) is in each core. Shared between the two cores is 128 KB of low-latency L2 memory.

We developed the code with Analog Device's Visual DSP++ integrated software development and debugging environment (IDDE), using both high-level C/C++ and Blackfin assembly language. The EZ-KIT came with a USB debugger unit to interface the JTAG emulator port to the IDDE. However, we sped up our development by borrowing one of Dr. Smith's research in-circuit emulators (HPPCI-ICE).

Our system comprises a pool table, a standard camcorder, a TV, and the DSP evaluation kit (see Figure 1). The camcorder is mounted over the pool table. The video stream is sent to the DSP, which analyzes the video data stream and finds the locations of all the balls on the table. A live loop-back video output stream is sent from the DSP to the TV, which contains the camcorder feed and additional visual feedback information in the form of video graphic overlays. This additional information con-

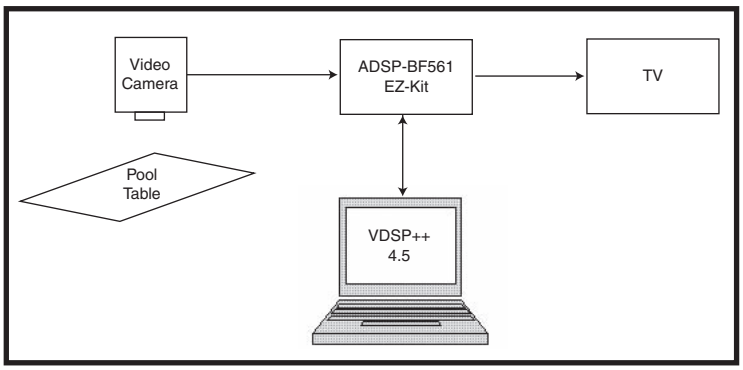


Figure 1—All the video processing is done on the DSP, but the EZ-KIT is connected to the PC via the HPPCI-ICE for development and debugging.

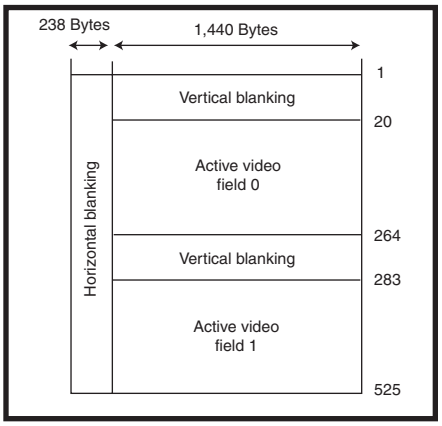


Figure 2—ITU-R 656 formatting for an NTSC video frame. Depending on the protocol version, there may be more or less vertical blanking lines. However, in general, the combined active data resolution is 720 × 480 pixels. (Adapted from the ADSP-BF561 hardware reference manual.)

sists of shot suggestions and ball-path tracking.

VIDEO BASICS

The Blackfin evaluation kit has an Analog Devices ADV7183 video decoder and an Analog Devices ADV7179 video encoder, which are compatible with different video standards. We will focus on explaining the configurations we used.

The National Television Standards Committee (NTSC) provides video standards for North America, including the ITU-R 656 data protocol, which specifies how video data should be formatted and the timing characteristics of the signals. ITU-R 656 uses an interlaced method of transferring data in which a single frame of active video (what you see on your TV set) is divided into two fields: Field 0 and Field 1. Field 0 contains the odd numbered lines. Field 1 contains the even lines. An NTSC video frame consists of 525 lines with each line consisting of 858 2-byte pixels. Blanking sections (known as vertical-blanking and horizontal-blanking data) are embedded between each field and at the beginning of each line (see Figure 2). When a TV displays the data, the odd lines are scanned first, followed by the even lines. The interlaced method reduces flickering, which was common in older TV sets. The blanking sections act as time delays to ensure that the electron gun inside the TV has enough time to align to the correct pixel location. Therefore, the viewable resolution of a single video frame is 720 × 480 pixels.

The ITU-R 656 standard requires the 4:2:2 YCrCb format. Uncompressed video requires one luminance (Y), one red chrominance (Cr), and one blue chrominance (Cb) value for each pixel (4:4:4 data format). In the 4:2:2 subsampling method, every four pixels contains four Y, two Cr, and two Cb values. Since the human eye is more sensitive to changes in luminance than changes in chrominance, the loss is relatively unnoticeable. The subsampling method is considered worthwhile due to a 33% memory reduction, compared to the minor visual data that is lost. That is why the 4:2:2 YCrCb method is preferred over the typical uncompressed RGB (red, green, blue) pixel format for video encoding.

PPI controllers are also designed to receive the ITU-R 656 video format with various customizable options. They include receiving only active video data, V-blank data, or the entire field (active and V-blank data). Another option for PPI is to use "656-compliant" video formats, where the structure of the frame is similar, but the number of V-blank lines, active video lines, and active pixels per line can be defined by the user. Since we are trying to provide a live feed to an output TV, we configured PPI0 to receive the entire field of standard NTSC frames. One DMA channel is allocated to transfer video frames from PPI0 (which is hooked up to the video decoder data lines) to a buffer in SDRAM. Another DMA channel transfers frames from SDRAM to PPI1 (which is hooked up to the video encoder) for video output to the TV.

VIDEO FRAME PROCESS

The memory structure in the Blackfin is arranged into three sections: L1, L2, and L3. L3, external memory, runs at speeds up to 120 MHz. L2, internal memory that is shared between the two cores, runs at 300 MHz. Each core has its own 600-MHz L1 memory. The limited size of L1 and L2 is insufficient to contain the entire 858 pixel by 525 line video frame. Each video frame is 900 KB, exceeding the size of the available internal memory. Processing directly on L3 is out of the question because memory access delays have a detrimental effect on system performance.

To solve the problem, we divided Field 0 of the L3 frame into 15 sections, each containing 720 pixels by 16 lines. This means that the H-blank and the V-blank are ignored because they do not contain useful data. A total of four 22.5-KB buffers were set up in L2 memory—two for current-frame data and two for the background frame. (We will describe the usefulness of the background image later in this article.)

To coordinate the transfer of data from L3 to L2 memory, we used the memory-DMA (MDMA) controllers. The dual four-channel MDMA controllers are independent from the two 16-channel DMA controllers available for peripheral devices. The four buffers in L2 are logically grouped into two blocks and used as ping-pong buffers (see Figure 3). While block 0 is used for video processing, block 1 is filled with the next partition by the MDMA controller. When video processing is done on block 0 and data transfer is finished on block 1, block 0 is used as memory space for the next section of data and processing is performed on block 1. This allows the Blackfin processor to perform instructions more efficiently because data transfer and video processing are occurring simultaneously.

When using the dual DMA/MDMA controllers, maintaining full utilization of the DMA/MDMA buses is

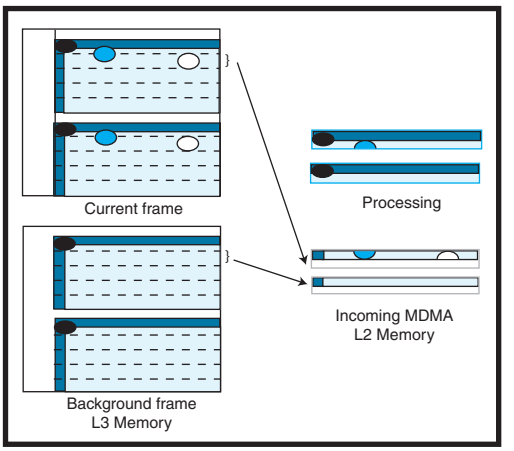


Figure 3—While the Blackfin processor is operating on block 0 current frame and background blocks buffers, the MDMA controller is transferring the next set of lines into block 1 buffers.

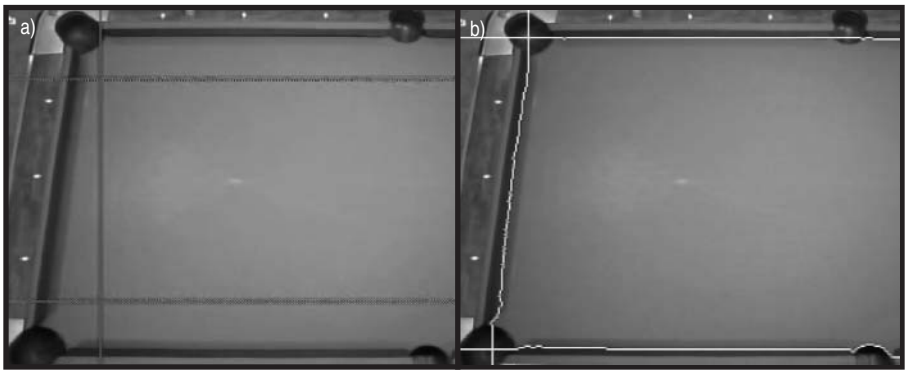


Photo 1—Guide lines are drawn to help position the camera **(a)**. Rails are marked in white by the calibration routine **(b)**. Note that the camera is not perpendicular to the table surface because the system was in Albert's low-ceiling room.

important. MDMA controller 1 (MDMA1) and MDMA2 each have two channels. The two MDMA controllers share the data bus on odd clock cycles. Using MDMA1 for current-frame block transfers and MDMA2 for background-frame block transfers avoided bus collisions and maintained full bus use. You should also consider making sure each of the L2 buffers is placed in different memory pages to avoid conflicts between the MDMA fill and core accesses during processing.

DRAWING ALGORITHMS

Drawing algorithms were necessary to overlay lines, boxes, and circles on the screen. We performed simple calculations to find the correct location of the pixel values and modify the proper Y, Cr, and Cb values.

To provide user feedback, each detected ball was highlighted by drawing a box around it. Rather than using the slower floating-point operations by calculating the lines using the standard formula ($y = mx + b$) slope-calculation algorithm, we used the high-speed Bresenham's line-drawing algorithm. The algorithm's advantage is that only integer arithmetic is used to calculate pixel locations between two points on a line. This approach is optimal for the integer-based Blackfin processor. Also required by our ghost ball prediction algorithm (more on this later) was the need to draw circles. An eight-way symmetry circle-drawing algorithm was implemented to reduce the number of pixel-location calculations needed to draw the circle.

SYSTEM CALIBRATION

System calibration needs to take place before ball tracking can begin. During calibration, the boundaries of the table cushions and the locations of the pockets were determined. A background image (a reference image for the system) was then captured. We developed an easy one-button automated calibration process to encapsulate the processes from the user. Pressing the SW8 push button on the EZ-KIT board initiates the calibration process, which draws guidelines on the display for the rail-detection algorithm search region (see Photo 1). The rails of the pool table must lie within the guidelines in the video camera's field of view. Once the camera is properly positioned, pressing SW8 again runs the calibration algorithms.

The exact location of each rail is determined by the system. The system is designed to work on half of a pool table, so we made it capable of searching for the end-rail and half of the two side rails. The reference color and intensity of the table felt are initialized to be the pixel location in the center of the frame. The boundaries of the rails are found by locating pixels with large differences compared to the reference color and intensity. Scanning starts at the middle of the frame and progresses outward in the three directions towards the rails. A low-pass smoothing filter is applied to each of the three rails to eliminate stray pixels that may be caused by noise. The final result is three (pseudo-straight) lines representing the boundaries of the playing surface. The pocket locations can be determined from the intersecting lines of the three rails.

Since the table and the cushions are made of the same felt, we either had

to spend a large amount of time trying to find the exact boundaries of the rails using minor changes of shade (difficult to distinguish) or take a more straightforward approach. We used our Canadian heritage and applied a strip

Listing 1—Here we compare the Blackfin assembly code using SIMD and software-loop pipelining (left) with the unoptimized code (right). Performing on a 720 × 16 pixel block took 110,500 and 138,500 clock cycles for optimized and unoptimized, respectively. This works out to a 20% increase in speed.

```
_CalcDiffASMOptB:
    LINK 0;
    [--SP] = (R7:4);

    I0 = R0; // CF address
    I1 = R1; // BG address
    I2 = R0;
    I3 = R1;
    M0 = 8;  // modify by 2 words
    M1 = 8;
    M2 = 8;
    M3 = 8;
    I2 += 4; // offset CF by 1 word
    I3 += 4; // offset BG by 1 word

// Prefetch first 2 words
    R2 = [I0] || R0 = [I1 ++ M1];
    (R4, R5) = BYTEOP16M(R3:2, R1:0);

    R4 = ABS R4 (V) || R2 = [I2] || R0 = [I3 ++ M3];
    R5 = ABS R5 (V);
    R4 = BYTEPACK(R5, R4);

    (R6, R7) = BYTEOP16M(R3:2, R1:0);

    R6 = ABS R6 (V) || [I0 ++ M0] = R4;
    R7 = ABS R7 (V) || R2 = [I0] || R0 = [I1 ++ M1];

    (R4, R5) = BYTEOP16M(R3:2, R1:0);

    P0 = (5760 - 4) / 2;
    LSETUP(_B_loop_start1, _B_loop_end1) LC1 = P0;
_B_loop_start1:
    R6 = BYTEPACK(R7, R6);

    R4 = ABS R4 (V) || [I2 ++ M2] = R6;
    R5 = ABS R5 (V) || R2 = [I2] || R0 = [I3 ++ M3];

    (R6, R7) = BYTEOP16M(R3:2, R1:0);

    R4 = BYTEPACK(R5, R4);

    R6 = ABS R6 (V) || [I0 ++ M0] = R4;
    R7 = ABS R7 (V) || R2 = [I0] || R0 = [I1 ++ M1];
_B_loop_end1:
    (R4, R5) = BYTEOP16M(R3:2, R1:0) ;

    R6 = BYTEPACK(R7, R6);
    R4 = ABS R4 (V) || [I2 ++ M2] = R6;
    R5 = ABS R5 (V) || R2 = [I2] || R0 = [I3 ++ M3];
    (R6, R7) = BYTEOP16M(R3:2, R1:0);
    R4 = BYTEPACK(R5, R4);

    R6 = ABS R6 (V);
    R7 = ABS R7 (V) || [I0 ++ M0] = R4;

    R6 = BYTEPACK(R7, R6);
    [I2 ++ M2] = R6;

    (R7:4) = [SP++];
    UNLINK;
    RTS;
_CalcDiffASMOptB.end:
```

Original unoptimized code

```
_CalcDiffASMB:
    LINK 0;

    P0 = R0; // CF address
    P1 = R1; // BG address

    P2 = 5760;
    LSETUP(_B_loop_start, _B_loop_end) LC1 =
    P2;
_B_loop_start:
    R2 = [P0];
    R0 = [P1++];
    (R0, R1) = BYTEOP16M(R3:2, R1:0); // quad
    byte diff

    R0 = ABS R0 (V);
    R1 = ABS R1 (V);

    R2 = BYTEPACK(R1, R0);
_B_loop_end:
    [P0++] = R2;       // store diff pixel

    UNLINK;
    RTS;
_CalcDiffASMB.end:
```

of black hockey tape on top of each cushion.

OBJECT DETECTION

We mostly used background suppression to detect objects. As we mentioned earlier, a reference background image was captured. The absolute difference between each background pixel and the current video frame pixel was calculated. By using special video-pixel and vector instructions, hardware circular buffers, and zero-overhead loop registers available on the Blackfin, the operation was performed efficiently. Further optimizations with software-loop pipelining and single instruction multiple data (SIMD) instructions make the operation even faster.

The special hardware instructions that are of interest to us are the BYTEOP16M, Vector ABS, and BYTEPACK instructions. The BYTEOP16M instruction performs a quad 8-bit subtract on a 32-bit register, treating each byte of the register as an individual value. Vector ABS instructions were used to calculate the absolute value of the individual top and bottom 16 bits of a 32-bit register. Finally, BYTEPACK combines four 16-bit values from two 32-bit registers into four 8-bit values packed into a single 32-bit register.

SIMD instructions can be used to execute up to three instructions simultaneously, including a logic operation on previously received values and two memory loads of values to be processed. There are, however, limitations on what individual instructions can be paired to make a legal SIMD instruction. Also, you must be careful implementing software-loop pipelining. We had to make sure not to place instructions that use the same register consecutively (e.g., operate on register R1 followed immediately by storing the same R1 to memory). That could degrade system performance because the execution pipeline would have to be stalled while waiting for the first instruction to finish executing the write-back process. To minimize this, two quad-byte operations performed each loop iteration using two independent sets of registers. Listing 1 shows our hand-coded assembly code for the image-differencing routine.

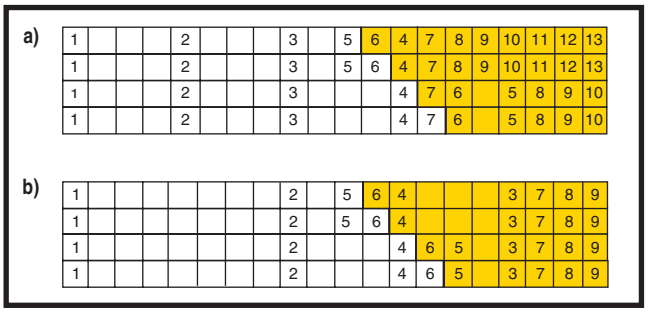


Figure 4—The numbers indicate the pixels, and the order they are probed in the step-search algorithm, with a `StepSize` of four and eight (**a** and **b**), respectively. Orange boxes indicate the presence of a ball. White boxes are the table's playing surface.

Once the difference image has been acquired, it is passed through a threshold detector for object detection. The level of the threshold detector is determined experimentally and has to be changed when the lighting environment is altered. We were planning to implement a dynamically

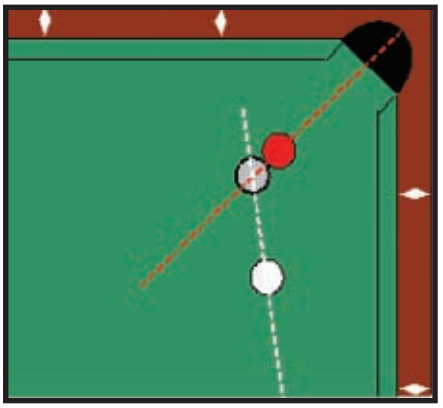


Figure 5—Here is the ghost ball prediction method. Contact with the red target ball must be made at the imaginary grey ghost ball's location.

changing threshold, but we were limited by the project's time constraints. At this point, the binary image was scanned, starting at the top-left pixel, and working row by row. Active pixels that are close to each other, both in the horizontal and vertical axes, were marked as belonging to one object.

OBJECT COORDINATE EXTRACTION

Object coordinate extraction is performed using a step-search algorithm. In this algorithm, only a fraction of the difference-image pixels are probed to detect objects. Each pool ball was about 16 pixels wide, but this will vary depending on how far or close the camera is to the pool table.

Scanning starts at the top-left corner of the difference image, we will call the pixel `(x, y)`. If the pixel is above a certain threshold, then it is considered part of an object. If it is below the threshold, then the pixel at location `(x+StepSize, y)` is tested. Since each pool ball was about 16 pixels wide, we set the variable `StepSize` to four. Setting the `StepSize` to this value would make sure that pool balls would not be skipped when scanning. If the probe pixel `(x, y)` is above the threshold, then we will try to find the left-most column that is part of the ball by probing `(x-(StepSize/2), y)`. We will probe left by another factor of `StepSize/2` if this pixel is above the threshold. Otherwise, we will probe right by a factor of `StepSize/2`. Left/right probing will be repeated until the `StepSize` equals one. At that point, we will have found the left-most edge of the ball on the row. Then, a linear search is performed to find the right-most edge of the ball in the row (see Figure 4).

We now compare this newly found left/right column location pair against each element in our array of previously found object locations. If this pair is in close proximity to an existing object location, then the data is appended. Since the balls cover only a small percentage of the table's playing surface and we are directly probing only one-quarter of the pixels, this method improves the speed of object detection.

A box is drawn around every ball that is detected. The color of the each box corresponds to the ball's color. The cue ball is distinguished by comparing each ball's color to white (or the white representation of the specific camera being used and the lighting environment).

Because of the detection algorithms and the nature of performing object detection as opposed to object recognition, problems occur when objects other than pool balls enter the frame. To alleviate this, a size limit was placed on each target detected. This helps reject the detection of larger objects that may be on the table, such as a player's hand. Imposing a 4:3 aspect ratio on object sizes also reduces sporadic detections. However, more work needs to be done on this to make the system more robust.

GHOST BALL PREDICTION

Ghost ball prediction is a technique

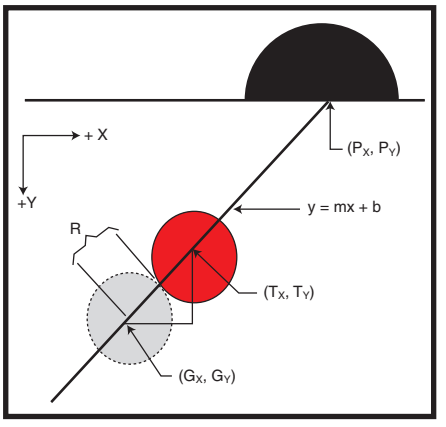


Figure 6—The red target ball and grey ghost ball positions are centered at (T_x, T_y) and (G_x, G_y), respectively. Each ball has a radius of R. $y = mx + b$ is the equation that describes the line, having a slope of m and y-intercept of b. The desired pocket has coordinates of (P_x, P_y).

Photo 2—Here the ghost ball detection is finding five object balls and drawing the expected contact point on each.

commonly used by novice pool players to aim shots. It is a simple method for estimating where an object ball should be contacted, so it follows a desired path. A line is drawn from the pocket through the target ball (see Figure 5). The center of the imaginary ghost ball lies on this line, directly behind the target ball. This places the target ball between the center of the ghost ball and the pocket. If the cue ball is shot at the ghost ball, the target ball should follow the path leading directly into the pocket.

After the object-detection phase, the locations of the pool balls and the cue ball are known. The most optimal pocket for each cue/colored ball pair is found by calculating the relative positions of the cue ball, target ball, and the distance to each pocket. We define the "optimal pocket" by using several criterion. First, is the shot in question physically possible for the determined pocket? For example, if the cue ball is positioned in between the target ball and the target pocket, then the shot is impossible without first rebounding off a rail. Second, the distance from the target ball to each pocket is evaluated to select the easiest shot (shortest path) for the user.

If an optimal pocket is found and the cue ball is present, then the center point of the ghost ball (G_X, G_Y) is calculated by solving the system of equations, which is derived from Figure 6:

$$G_Y = mG_X + b$$

$$0 = G_X^2\left(1 + m^2\right) + G_X\left(2bm - 2T_X - 2mT_Y\right) + \left(T_Y^2 + T_X^2 - 4R^2 - 2T_Y b\right)$$

The pocket location (P_X, P_Y), target location (T_X, T_Y), the line slope (m) and y-intercept (b) can be determined. You must be careful when evaluating line slopes and roots of the quadratic equation because division-by-zero or square roots of negative-number exceptions may be present. Solving this quadratic equation yields two results for G_X. If the pocket lies on the right side of the target ball, then the smallest magnitude root was chosen, while the largest root was chosen if the pocket was on the left side.

For every colored ball that is found, ghost ball positions and predicted trajectories are drawn to the optimal pocket (see Photo 2). As the balls

A Design Conference with some Real Muscle.

ARM Developers' Conference and Design Pavilion

October 2-4 2007 Santa Clara Convention Center

Strengthen your skills and speed your time-to-market

Only the ARM Developers' Conference offers:

- Over 90 track sessions providing a complete end to end design tutorial for leveraging ARM IP in advanced embedded applications
- Combined tutorials with Portable Design Conference
- Design Centers and exhibitions from leading ARM licensees and Connected Community members offer a full complement of workshops and presentations
- Forums and analyst presentations on industry trends

Plus: through the combination of track sessions, presentations and company design centers you will be fully immersed in leading strategies and methodologies for building complex designs with the ARM architecture.

And Finally-- Conference Delegates who pre-register before September 10 receive the conference proceedings on a 4 GB iPod Nano to take home!

Register early and save!

www.arm.com/developersconference

Co-Located with the Portable Design Conference & Exhibition

move across the table, their individual locations are recorded in a history buffer. Each of the ball's history buffers is 4 s and allows history trails to be drawn on the screen to allow the user to review their shot.

OBSTACLES

Changes in lighting from different environments had a drastic effect on our system. It caused our predetermined thresholds to be invalid. The thresholds therefore needed to be recalculated when the system was moved. Differences between two similarly colored balls (i.e., orange and red) also made it very difficult to differentiate between the two balls.

Another problem we encountered was an effect called keystoning. This happens when the camera is not perfectly perpendicular to the surface of the table, making the rails appear nonperpendicular. The imperfection caused the ghost-ball aiming algorithm to be less accurate. Implementing a software correction may help this problem, but we decided to opt for the easier solution and move the system to a new house where the ceilings in the basement were higher. This makes it easier to place the camera.

DEBUGGING TOOLS

A very useful feature of the BF561 is the background telemetry channel (BTC), which allows VDSP++ to read from and write to memory without interrupting the processor cycle. This is performed with spare DSP cycles during idle. By hooking up the BTC specific global variables, we could adjust data on the fly without halting and recompiling the program code. This feature made it easier to fine-tune threshold or step-size values, as well as enable and disable ghost ball predictions or history trails.

We successfully developed an initial prototype system that is able to perform object detection, simple trajectory prediction, and motion tracking. Background suppression is a very basic concept for object detection, but it worked well for us. The ghost-ball prediction algorithm, combined with the ball history trails, helps the beginning pool player. Although time was a limiting factor in the progress of the project, we met or exceeded most of our design goals. In the future, we hope to have more time to expand the project.

FUTURE WORK

At this stage of the design, we have implemented only predictions of simple shots in a half-table scenario. We intend to enhance the system to include multiple ball combinations or bank-shot predictions, as well as statistical calculations (such as successful shot likelihood if there are multiple shot options) to let you know which shot is best in a particular situation.

A more difficult challenge would be to allow compensation for the effects of the cue-imparting spin on the cue ball (known as "English"), and the spinning cue ball transferring this spin to the object ball (known as "throw"). This would be very hard to detect in a 30-frame-per-second environment, and most likely, a high-speed video-frame capturing device would be required. We're also considering a wireless system that allows multiple users to view the shot on their own screens. This would be ideal in a tutorial scenario. (At the end of the project, we implemented a prototype wireless system, which transferred a static video image from one Blackfin EZ-KIT to a second EZ-KIT, and then displayed it on a TV. The system used Analog Devices's ADF7020 radio transceivers hooked up to the Blackfin SPORT ports.) The principles of this system can be applied to other motion-tracking applications like security/surveillance, wildlife monitoring, and tracking objects in other sports, such as bocce or bowling.

Currently, one core does all the processing in the software architecture, while the second core practically sits idle, performing only user I/O routines and main system-flow control. If more computationally intensive routines are used in the future, the workload could be shard between the two cores.

The biggest problem currently facing our system is the detection of noncircular objects. This is the most limiting factor in the overall robustness of the system. A better algorithm needs to be developed to reject the detection of these objects.

Authors' note: We would like to thank Dr. Michael Smith at the University of Calgary for supervising our project and for the work he did throughout the year. We would also like to thank Warren Flaman for creating the expansion port interface and various technical support help. Hernan Vassolo supplied the video camcorder. We also want to recognize the members of Dr. Smith's other design group (Henry Chen, Simon Cheng, Maggie Zhang, and Chris Blauel) for their help with this project. Finally, many thanks go to Analog Devices (Mimi Pichey and Bob Meissenhelder) for providing the ADSP-BF561 EZ-KIT Lite and various other hardware components.

Albert Tran (alctran@ucalgary.ca) has a B.Sc. in Computer Engineering. He is currently working toward an M.Sc. in researching high-speed algorithms used in magnetic resonance imaging at the University of Calgary. His interests include DSPs and embedded system designs.

Andrew Kwan is a graduate student studying software-defined radio algorithms at the University of Calgary. He holds a Bachelor's degree in Computer Engineering. You may e-mail him at akckwan@ucalgary.ca.

Kevin Brown holds a B.Sc. in Software Engineering. He can be reached at kt.brown@shaw.ca.

Jason Vangilst holds an Electrical Engineering degree from the University of Calgary. You may e-mail him at jvangilst@hotmail.com.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/204.

SOURCE

ADF7020 Transceiver, ADSP-BF533 Blackfin processor, ADSP-BF561 Blackfin processor, ADV7179 video encoder, and ADV7183 video decoder
Analog Devices, Inc.
www.analog.com

Multipurpose Automotive Gauge

When Eric could not find the right gauges to update the instrumentation in a 1965 Mustang, he built his own multipurpose gauge. Now he can monitor everything from engine air flow to oil temperature.

When the market doesn't provide you with what you really want, you must either compromise or take matters into your own hands. This was the case when I wanted to update the instrumentation in a 1965 Mustang. I knew that replacing the carburetor with a fuel injection system would allow a host of new sensors to be monitored. I wanted the increased resolution and the full 270° sweep that mechanical gauges provide instead of the typical 180° sweep generated from electric gauges. I later found a few electric gauges that provided 270° sweep, but few if any of them were directly compatible with the sensors used by the engine control module (ECM). This meant additional sensors would be required. I could not find a manufacturer that produced gauges for all of the functions I wanted to display, creating a stew of mismatched faceplates. So, out of seeds of discontent, the multifunction gauge described in this article emerged.

I thought about converting an existing air core movement gauge, but I wanted something that could also provide a digital readout. After toying with how many T1 LEDs would fit inside a 2.0625" gauge housing, I had my answer. With three digital displays surrounded by 32 segments, analog and digital representations were covered. Since I had worked with other digital automotive gauges, I felt obligated to address some of their undesirable properties, such as washout in direct sun and fixed and often inappropriate update rates.

With a clean slate, I built a wish list of desired features. Ideally, the 32 LED segments would have an analog "feel" to simulate an analog needle. With bar-graph scaling and format-programmable options for update rates and my choice of bar graph display modes, formats such as bar growing between setpoints, bar shrinking between setpoints, bar shrinking to setpoints then growing after setpoints, and simple dot or bar graph display modes are possible.

After a format for displaying data was selected, the data and its format were determined. Sensors used by the ECM include engine airflow, coolant and intake air temperature, throttle position, distributor/camshaft position, and vehicle speed. Other engine parameters and sensors of interest include oil pressure, oil temperature, fuel pressure, fuel injector duty cycle, pulse width, and an aftermarket wideband exhaust gas oxygen sensor. The battery voltage, fuel level, rear axle temperature, and G force were also included. The sensors can also interpret ignition timing in degrees before top dead center (BTDC), miles per hour, and revolutions per minute.

Photo 1—Here is a sampling of sensors, including a mass airflow housing and a sense element, an MAF element, a fuel level float sensor, air charge temperature, and a Senstronics 0- to 100-PSI pressure sensor. The MAF and temperature sensors provide nonlinear outputs between 0 to 5 V. The pressure sensor produces a 0- to 5-V linear output and the fuel level sensor a semilinear 12-Ω full/73-Ω empty output.

Measuring these parameters requires both digital and analog capabilities. Most of the analog sensors are based on a 5-V reference supplied by an ECM, but others, such as battery voltage, require something in the 20-V range. The temperature and MAF sensor outputs are not linear and require further processing for a reasonable accuracy over anything but a small window. Digital sensors fall into frequency, pulse width, duty cycle, or phase categories. The digital-output voltages ranged from a few hundred millivolts to more than 30 V. Frequencies range from a couple of hertz to a couple of kilohertz. The

information helped define the hardware required to do the job. Take a look at Photo 1 for a sample of a few sensors that may be monitored.

PLUMBER'S HELPER

Before getting far along with the PCB layout, I looked for a suitable enclosure. With a nominal 2.25″ outside diameter, I found that a 1.5″ PVC pipe end cap was close to the standard 2.0625″ gauge body diameter. The 1.5″ inside diameter and 1.5″ cap depth defined the hardware envelope. Some minor lathe work gave a nice flange and groove to hold an O-ring, faceplate, legend, lens, and bezel. The bezel comes from a 2″ schedule 40-PVC pipe machined on the inside to match the PVC cap's outside diameter. Custom legends are constructed and easily customized with cutouts from laser printer transparencies created in Microsoft Word. Photo 2 shows what is inside and how it fits together.

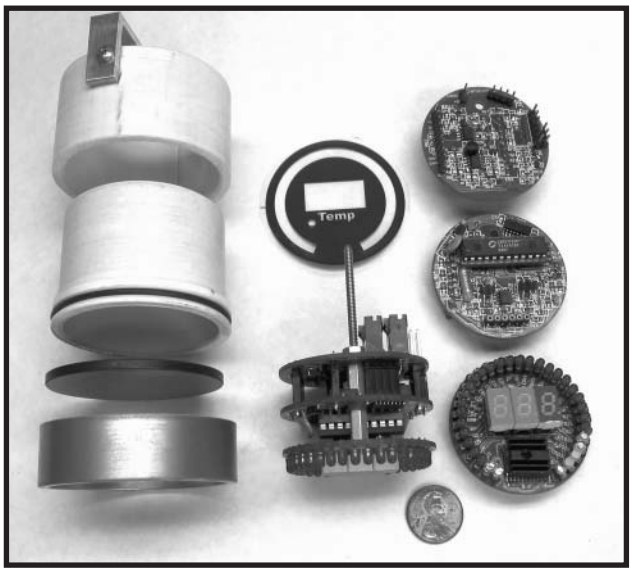


Photo 2—Take a look at a close up of a gauge assembly. The clamp in the upper left is a split piece of 2″ PVC schedule 40. A 6-32 screw threads through all PCBs and spacers, holds the PCB assembly in an enclosure, and pulls the clamps against the rear of the panel. Input, output, and ISP connectors on the rear-interface PCB fit through cutouts on the backside of the enclosure. Female 0.1″ headers on the processor PCB and male headers on display and interface PCBs provide interconnects and easy assembly.

Three round PCBs are inside the 1.5″ cylinder: display, processor, and interface. The display board contains a Maxim Integrated Products MAX7219 64-segment constant current driver and a cadmium sulfide cell to control the display elements. The processor PCB holds a Silicon Laboratories C8051F330 microcontroller, basic analog and digital signal interfaces, and both 3- and 5-V regulators. With all segments on at full brightness, it burns at about 250 mA. Due to the limited physical and heat sinking area available, I used a Texas Instruments PT5101 5-V integrated switching regulator. These two boards will provide a limited network, input protection, and analog operation. I used the C8051F330 because it is available in a prototyping 20-pin DIP package. I also made an MLP-20-to-DIP adapter in case the DIP device is discontinued and to see if hand-soldering the tiny surface-mount MLP with 0.5-mm pitch was possible for future projects, which it can be under optimal conditions.

To use most of the 10-bit 200-kHz ADC, a separate low- and high-input range is available. A variable input-gain stage using a 256-step digital potentiometer provides fine-tuning for calibrating the 0- to 5-V and 0- to 20-V ranges without a screwdriver. It can also increase the gain to better use ADC resolution for

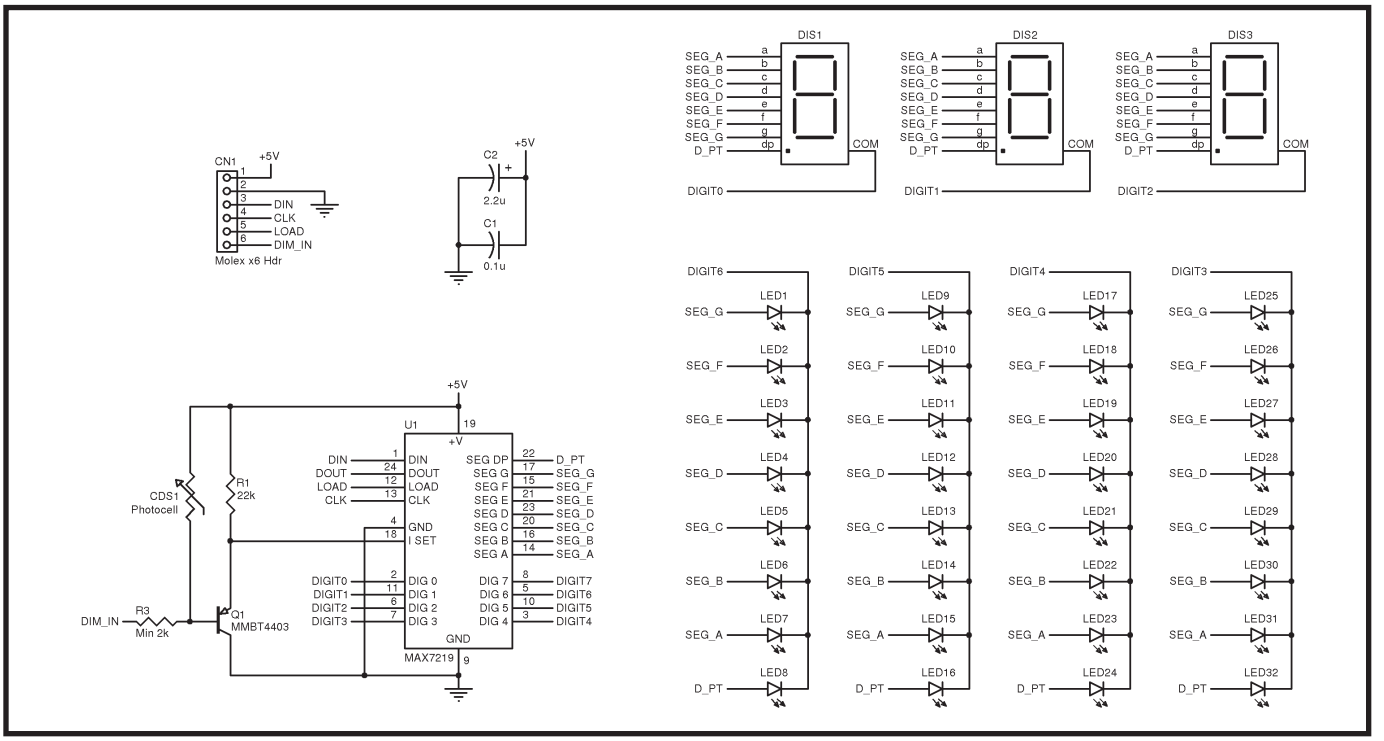


Figure 1—The display board assembly consists of a surface-mount serial driver along with through-hole, discrete, and seven-segment LEDs.

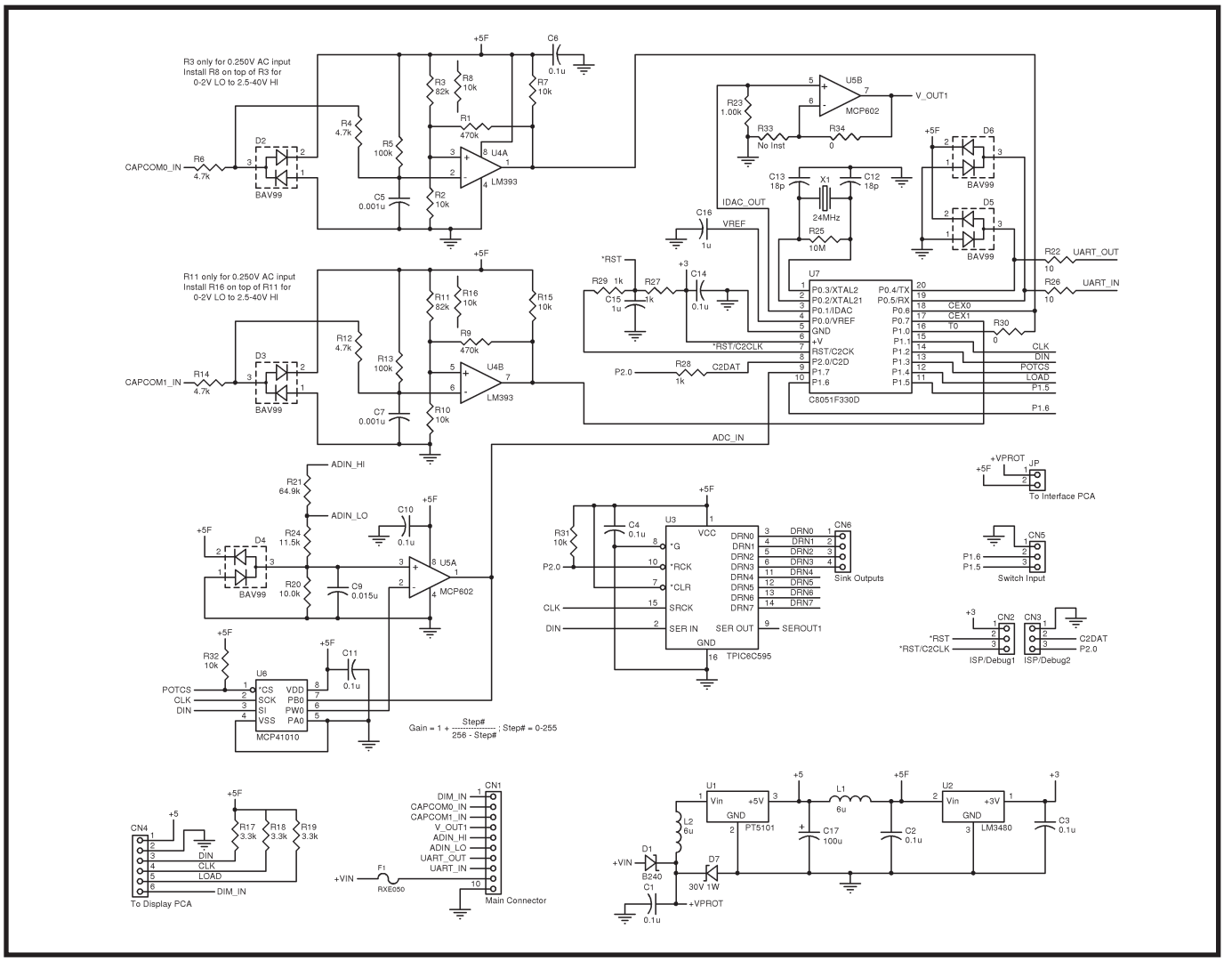


Figure 2—The processor board assembly is the meat of the three-board sandwich, which is packed tight with components on both sides of the PCB.

ranges lower than 0 to 5 V, although fine adjustments become more difficult as gain increases beyond three. The 10-bit DAC output current is converted to a voltage to be used as a sensor reference or to provide an analog output of the displayed value. Two filtered digital inputs are provided for the reading phase, with only one input used for frequency, pulse width, and duty cycle measurements. For the two high and two low alarm outputs, a serial 100-mA sink driver provides four 32-V ESD-protected outputs.

The interface PCB provides room for latching connectors, filtering for two push button inputs, additional analog buffering, and an automatically switching two-wire RS-485 driver (see Figures 1, 2, and 3).

SOFTWARE

The system's software was developed with Silicon Laboratories's IDE in assembly language. The IDE's software package includes a Keil assembler, a linker, a compiler, and support for in-circuit emulation and programming capabilities when used with an EC2 serial adapter device. The EC2 acts as the link between the processor and the host PC's RS-232 port. The IDE package, the EC2 , and the evaluation board are part of the C8051F330 development kit. I wrote the code in six functional sections: main, processor configuration, analog, frequency, pulse width/phase/duty cycle, and programming.

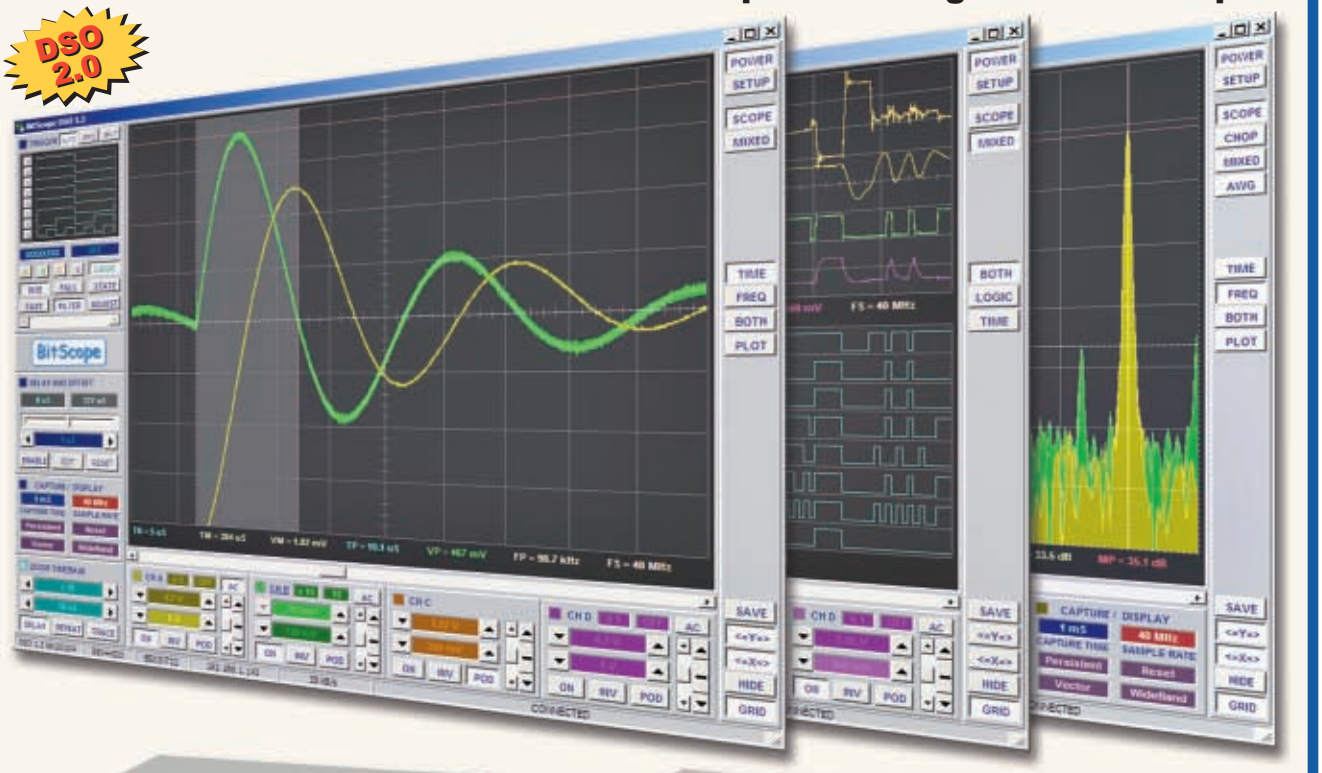
Three modes may be selected at powerup: Simple Operation, Operation with Serial Port Enabled, and Programming mode. During power-up, the processor's resources are configured and serial ports are initialized for 57,600 bps, 8 data bits, no parity, and 1 Stop bit. From here you have three options, as you can see in Figure 4.

Operation with Serial Port Enabled allows you to enter Programming mode, a bus master to disable communication with all gauges except specified and request gauge data for data logging. It also enables a gauge to send an ASCII "less than" character (<) when the gauge's MIN button is pressed and a "greater than" character (>) when the MAX button is pressed to mark events.

To request current gauge data, a master, such as a laptop computer connected to the gauge's serial bus, transmits serial data consisting of a single ASCII character from 01 to 0F. This command can be sent with HyperTerminal using characters from Ctrl A through Ctrl 0. As long as each gauge on the bus has a unique address,

BitScope PC Oscilloscopes & Analyzers

DSO Test Instrument Software for BitScope Mixed Signal Oscilloscopes



4 Channel BitScope

2 Channel BitScope

Pocket Analyzer

Digital Storage Oscilloscope
✓ Up to 4 analog channels using industry standard probes or POD connected analog inputs.

Mixed Signal Oscilloscope
✓ Capture and display up to 4 analog and 8 logic channels with sophisticated cross-triggers.

Spectrum Analyzer
✓ Integrated real-time spectrum analyzer for each analog channel with concurrent waveform display.

Logic Analyzer
✓ 8 logic, External Trigger and special purpose inputs to capture digital signals down to 25nS.

Data Recorder
✓ Record anything DSO can capture. Supports live data replay and display export.

Networking
✓ Flexible network connectivity supporting multi-scope operation, remote monitoring and data acquisition.

Data Export
✓ Export data with DSO using portable CSV files or use libraries to build custom BitScope solutions.

BitScope DSO Software for Windows and Linux

BitScope DSO is fast and intuitive multi-channel test and measurement software for your PC or notebook. Whether it's a digital scope, spectrum analyzer, mixed signal scope, logic analyzer, waveform generator or data recorder, BitScope DSO supports them all.

Capture deep buffer one-shots or display waveforms live just like an analog scope. Comprehensive test instrument integration means you can view the same data in different ways simultaneously at the click of a button.

DSO may even be used stand-alone to share data with colleagues, students or customers. Waveforms may be exported as portable image files or live captures replayed on other PCs as if a BitScope was locally connected.

BitScope DSO supports all current BitScope models, auto-configures when it connects and can manage multiple BitScopes concurrently. No manual setup is normally required. Data export is available for use with third party software tools and BitScope's networked data acquisition capabilities are fully supported.

www.bitscope.com

PC/104 Single Board Computers

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

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

- options include:
  onboard temperature sensor, A/D Converter 8 channel 12 bit, Extended Temperature, Battery Backed Real Time Clock, USB Flash 256 M (with ARM Tool Chain), USB WiFi

200 MHz ARM9
Power as low as 1/4 Watt

- 5 boards, over 2000 configurations — $99 qty 100
- Fanless, no heat sink
- SDRAM - up to 128MB — $129 qty 1
- Flash - up to 128MB onboard
- 10/100 Ethernet - up to 2
- DIO lines - up to 55
- 2 USB ports
- COM ports- up to 10
- Programmable FPGAs
- Linux, Real Time extension, NetBSD

NEW!
- SD card option
- VGA video

Off-the-Shelf Solutions ready to design into your project using DOS development tools

TS-5600 Shown with
optional flash modules,
A/D, RS-485 and
Merlin cellular modem

- options include:
  RS-485 Half and Full Duplex, A/D Converter up to 8 Channels at 12 bits, DAC up to 2 Channels at 12 bits, Extended Temperature

133 MHz 586

- Power as low as 800mA — $229 qty 100
- Fanless, no heat sink
- SDRAM - up to 64MB — $259 qty 1
- COM Ports - up to 4 ports
- Ethernet Ports
- DIO Channels - up to 40
- PCMCIA II adaptor
- Compact Flash adaptor
- USB Ports (Except on TS-5300)

see our website for 33 MHz 386 configurations

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

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

New Products and PC/104 Peripherals

Tiny WiFi Controller boots Linux in 1.1 seconds

$249 qty 1

- 200 MHz ARM9
- Up to 128MB Flash
- Up to 128M SDRAM
- 802.11g WiFi
- SD Flash Card socket
- 1 external USB port
- 1 10/100 Ethernet
- 3 TTL serial ports
- Rugged aluminum enclosure measures 1.1" x 4.9" x 3.1"

Intelligent Battery Back-up

NEW!

$119 qty 1

- Run your system for days with no external power source

NEW! ZigBee Wireless	low power wireless, simple serial interface, range up to 1 mile
Modems	33.6K baud, 56K baud, AT commands, caller ID, cellular using GSM and CDMA technologies
Non-volatile Memory	up to 2MB, 10 year lithium battery
Serial Ports	up to 4 serial ports with optional RS-485, opto-isolated available
12 bit A/D, DAC	8 channel 12-bit A/D converter, optional 2 channel 12-bit DAC, A/D jumpered for 0-2.5V, 0-10V or 0-20mA
CAN Bus Controller	Philips SJA1000, opto-isolated, up to 1 megabit/sec selectable termination resistor, Ocera Linux driver
64 Digital I/O	32 inputs, 32 outputs, 200 mA drive, optional 512 Kbyte or 1 MB battery-backed SRAM, stack up to four boards, RoHS compliant

see our website for more boards and option details

Technologic SYSTEMS

We use our stuff.

Visit our TS-7200 powered website at

www.embeddedARM.com

using Ctrl characters for commands prevents bus conflicts because the data output from a gauge in this mode will not contain the 01 to 0F characters. Gauge-to-master data is sent as four ASCII data characters and a comma. An alternative method for entering Programming mode is holding both the MIN and MAX push buttons before powerup. Figure 5 shows the available options with the serial port enabled.

Programming mode is indicated with a welcome message sent to the serial port upon entry and "prg" is displayed on a seven-segment display. Commands in Programming mode consist of single ASCII characters: L, D, S, and X (see Figure 6). Only one gauge is programmed at a time. If more than one networked gauge has a serial port enabled and receives commands to enter Programming mode, enabled units will answer and cause conflicts. To prevent this, only the gauge of interest should have the serial port enabled.

This is accomplished by sending the "I (node #)" command in Operating mode with the serial port enabled.

In normal operation, sensor input data is handled by six functional modes (four digital and two analog). Only one mode is run at a time. The two analog modes consist of either reading the microcontroller's internal temperature sensor or an external analog signal. With the on-chip ADC running at up to 200 ksps, there is plenty of time for averaging. Accumulations of 2, 4, 8, 32, or 64 may be selected. For additional flexibility with nonlinear analog input signals, a 16 entry by 16 bit interpolation routine is utilized with the formula:

$$D_{OUT} = \frac{(AT_1 - A_i) \times TD_0 + (A_i - AT_0) \times TD_1}{(AT_1 - AT_0)} \quad [1]$$

where A_i is 16 bits of raw input. AT_0 is a 16-bit table input entry less than A_i. AT_1 is a table input entry greater than A_i. TD_0 is the table output data from AT_0. TD_1 is the table output from AT_1. This assumes that the first raw input data entry is 0 and subsequent entries are positive, increasing integers to satisfy the lookup table input breakpoint search. The table entries should begin at 0 and end at the maximum possible input value to prevent undefined outputs at the extremes. Table output data entries can either increase or decrease with table input, so positive or negative slopes are permitted. Of course, nice, simple linear functions are also accepted.

The result of the interpolation routine is used by the bar graph display look-up routine, D/A output, alarm limits, serial-output buffer, and a digital display averaging accumulator. An update cycle with 64 samples takes about 7 ms. The D/A output routine also uses a 16 entry by 16 bit interpolation table routine to convert the buffer value to a desired output voltage. This comes in handy when converting nonlinear inputs to linear outputs to read with a multimeter, o-scope, or analog-to-digital-to-analog functions. An intersample A/D conversion delay can be configured to take as long as 1.6 s to accumulate and then average 64 samples, providing the means for a crude but

effective symmetric finite impulse-response filter. This can help sift DC levels from low-frequency, amplitude-modulated inputs like sloshing fuel level from the fuel level sensor.

The seven-segment display buffer can also be configured to accumulate 2, 4, 8, 32, or 64 interpolated averages for further filtering and to reduce digit skipping between large-slope table entries. With 64 averages, the update cycle slows to about 460 ms.

Sensor inputs in digital form are processed by the Pulse Width, Duty Cycle, Phase, or Frequency modes. The pulse-width function measures pulses from 0.1 to 99.9 ms, duty cycles from 1 to 2.5 kHz, phases from 1 to more than 10 kHz, and frequencies from less than 1 to more than 10 kHz. Pulse-width and duty cycles are handled by a single 16-bit capture compare routine with a cascading byte bumped by timer overflow for a total of 24 timer bits. Phase measurement uses an additional capture-compare register to hold the extra events. Either mode may be configured to use the first rising or falling edge as a reference.

For frequency measurement, a 16-bit counter keeps track of how many pulses occur during the sample period. With more than one input pulse, sampling ends with the 32-ms rollover of the 16-bit timer. Otherwise, sampling ends with either the first pulse or the configurable timeout period from the 16-bit timer and cascaded byte. The frequency or rate is:

$$rate = \frac{(constant \times T_1\ count)}{(pulses\ per\ unit \times period\ in\ microseconds)} \quad [2]$$

Data acquired in these modes can be directly displayed or passed through the interpolation routine for further manipulation.

EXCEL-ERATED CONFIGURATION

While I was working on scaling factors with Microsoft Excel, I wanted to import data into the source code. With the ability to create graphs, trend lines, format data, and hex and decimal conversions, this worked well and provided a worksheet to test, view, and modify sensor parameters. The final seven-sheet Excel configuration file consists of three sheets for entering configuration data, two for internal formatting, one parameter output sheet formatted for compiler, and one parameter output sheet for serial downloading. The basic parameter edit sheet contains general-operating, analog, and digital parameters. The ADC linear interpolation table provides ADC calibration, function graphing, and table data entry. The DAC linear interpolation table holds the DAC output data for converting the digital display data to a voltage function. The three sheets edit over 20 parameters.

For the compiler-formatted sheet, a compiler-friendly file is generated and saved as gaugecfg.prn. A simple command file renames it gaugecfg.asm and places it in the correct directory for the compiler. To program a device with the EC2 serial-in-circuit programmer, the linkable gaugecfg.asm is assembled, compiled, and downloaded with the rest of the program files.

The output sheet for serial downloading contains configuration data, programming commands, and inverted checksums. The sheet is saved in .TXT format so it can be transferred to the gauge via the RS-485 serial port. I used HyperTerminal to communicate and download data through an RS-232-to-RS-485

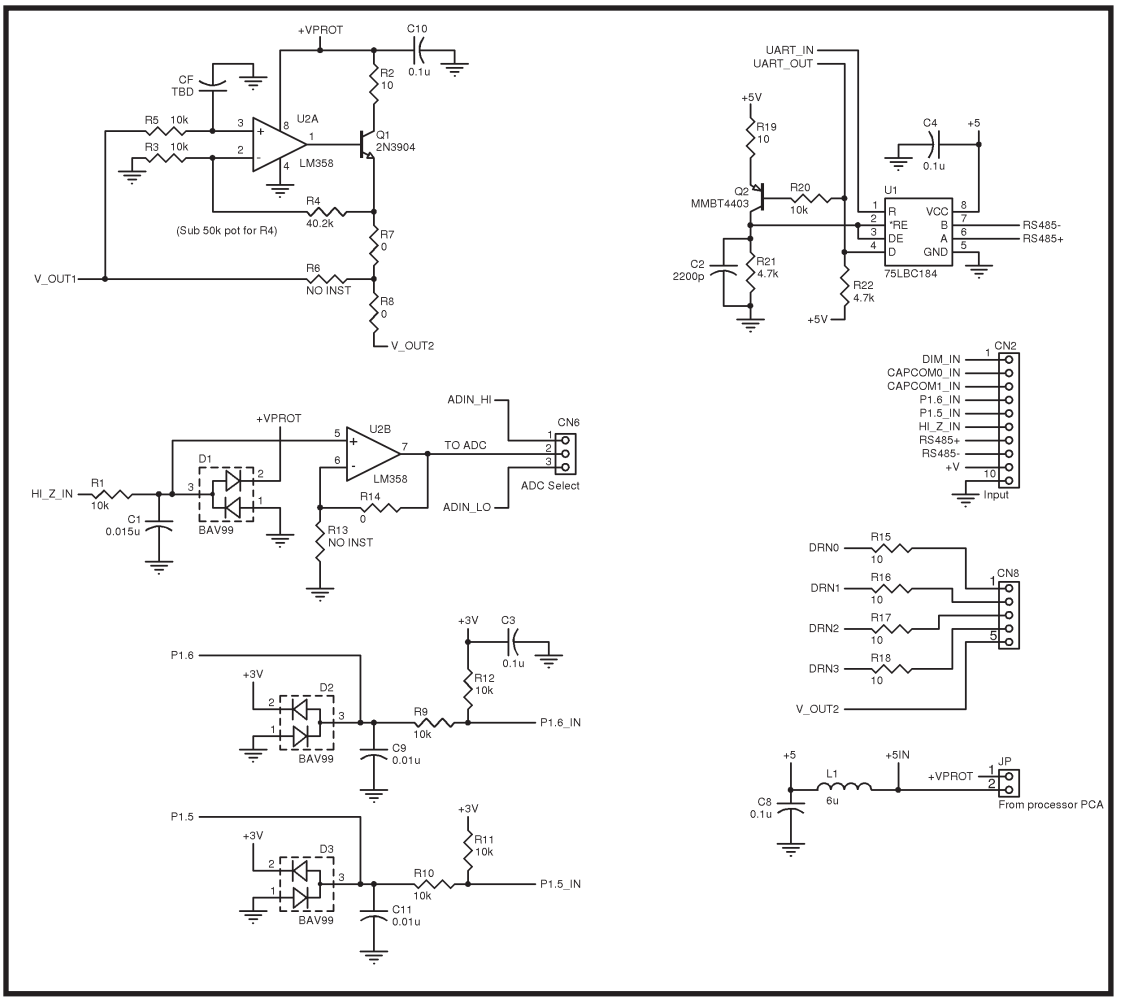


Figure 3—The interface board assembly adds buffering, locking connectors, and increased protection to inputs and outputs.

converter dongle.

MAKING THE CONNECTION

I built a test panel to hold the six assembled units. For analog measurements, either the high or low analog input must be connected. For phase measurements, a reference and the measured signal are required. Frequency, pulse width, and duty cycle require only a single input. With a host of options available, I had to decide which parameters to monitor. A bad sending unit from my old oil pressure gauge intermittently produced scary readings, and a less-than-accurate fuel level sensor piqued my interest to monitor them. Monitoring estimated horsepower and timing was unique. So, with fuel pressure and water temperature filling the last two slots, I looked for signals and sensors.

For fuel and oil pressure sensors, I chose a stainless steel Senstronics Storm ST00635 sensor with a range from 0 to 100 PSI, 0 to 5 V of output, and an operating range of –40° to 125°C. The 0- to 5-V temperature output is an added bonus that may be monitored as well. The operating temperature range could handle fuel pressure, but it would require thermal isolation for hot oil pressure sensing. Some distance from the engine block and a small heat exchanger built from mechanical oil pressure gauge tubing could help bring the oil temperature down. You have to be careful when using copper and certain aluminum tubing because the vibration can cause them to harden and eventually break. But mounting sensors directly to the engine can subject them to some serious failure-inducing torture, so there are gotchas either way. For now, the fuel sensor is mounted on a bulkhead made where the fuel lines enter the fender well. I plan to mount an OEM fuel pressure sensor directly on the fuel injector rail supply line, where it will give a more accurate indication of pressure at the injector. Since these are both new sensors, a small eight-conductor cable is run from the sensors to the gauge-test panel.

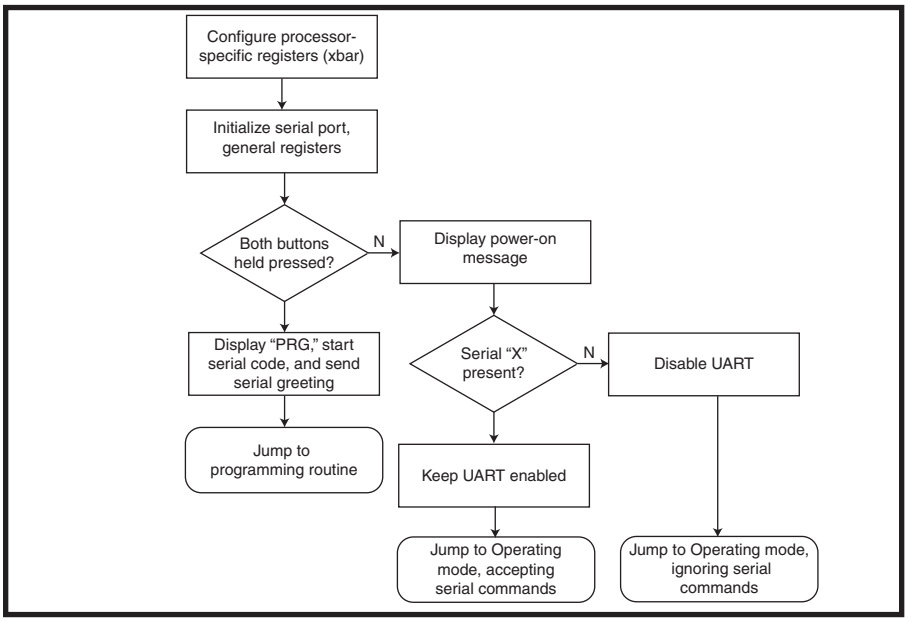


Figure 4—Here are the power-up flow and options for entering programming or data-logging operations. Cycling power is required to reinitialize serial communications.

Fuel level information could be tapped from the node connecting the original fuel level sensor and the gauge. The original fuel level sensor provided a 12-Ω full, 73-Ω spec empty resistance output when measured between the sensor output terminal connection and vehicle ground. Internally, the fuel level sensor was a wire-wound variable resistor with one end connected to the output terminal and the wiper connected to vehicle ground. Since I installed an aftermarket fuel level gauge and wanted to leave it in place, I had to determine how the sensor was excited. After probing with a multimeter, I found that the gauge connection to the level sensor contained an 8-V source in series with a 50-Ω resistor. The data along with the 16-gallon fuel tank capacity provided a voltage-to-fuel level function. Later, I discovered the 40-plus-year-old level sensor had a couple dead spots, which explained the erratic levels. When I replace the sensor, I will map the resistance and voltage versus capacity at gallon intervals for better accuracy.

Water temperature, horsepower, and time sensing all require a connection to the engine coolant temperature (ECT), mass airflow (MAF) sensor,

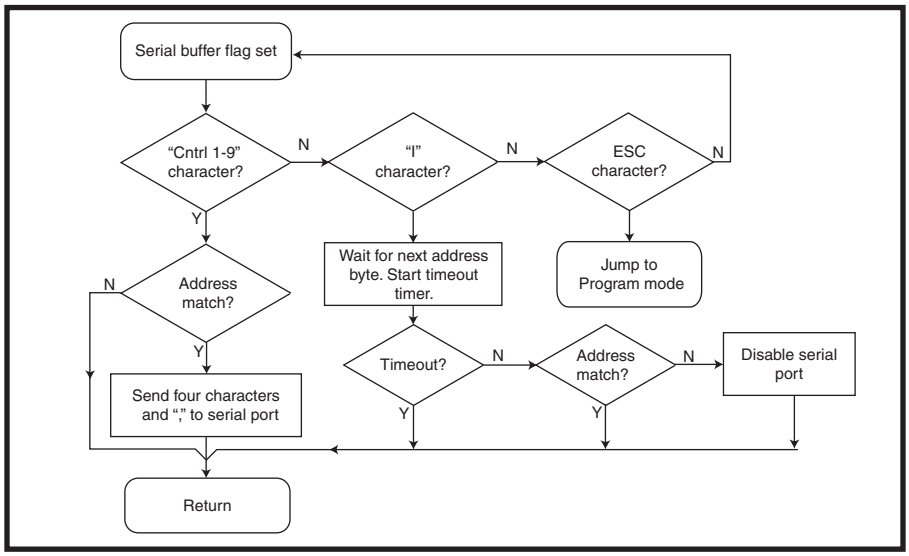


Figure 5—With the serial port enabled, here are some Operating mode options. You can query single-node data, isolate a node, or enter Programming mode.

Programming mode serial commands

L = Load parameters to flash, return status message
D = Dump parameters to port, terminates with <CR> <LF> <</CS
X = Exit this mode
<Esc> = Abort current operation
S = Start continuous serial output stream in dddd, 4 ASCII digit BCD format. Will show MSB not present on LEDs. Exits programming mode.

Examples:

L18 00 dd dd dd ..... dd<</CS = load flash memory at 1800h data unit << chars and inverted checksum received.
Returns programming success or fail status message. Entire block of flash erased. Must begin load at 1800, which holds parameter data.

D 18 00 01 2A = Dump 012Ah (298d) memory locations starting at 1800h.
(dd dd dd dd .. <</CS). Locations from 0 to 1FFF may be dumped.

S replies with Data Logging On message, then begins outputting dddd,dddd,dddd,.....dddd, unit <Esc> received, which reenters programming mode

Figure 6—Communications with the gauge can be performed using an RS-232-to-RS-485 adapter and a terminal program, such as HyperTerminal running at 57,600 bps, 8 data bits, no parity, and 1 Stop bit. Configuration files in text format may also be captured and copied to other units for duplication.

spark output (SPOUT), and profile ignition pickup (PIP) signals. With the ECM mounted behind the glove box, I spliced into the harness near the ECM connector. Since some glove box disassembly was necessary to reach the ECM, I spliced into the ECM V_{REF}, throttle position sensor (TPS), air charge temperature (ACT), and fuel injector signals for future measurements before putting things back together.

Creating a calibration for the ECT and ACT sensors requires entering the temperature versus voltage transfer function, as explained in the Ford shop manual.[1] That's all I needed, but I investigated further and found that the sensor appears and behaves like a 30-kΩ at 25°C MS96 material NTC thermistor, similar to the RL0530-17.56K-96-MS available from Digi-Key. This information helps if a Ford EEC-compatible ECT or ACT sensor is needed with a nonstandard thread. The ECT sensor can also be used as an engine oil, transmission, or rear-axle temperature sensor.

The horsepower measurement uses the MAF sensor. The transfer function in volts per approximate HP for stock and aftermarket MAF sensors are available from various Ford engine-tuning web sites. A simple cut and paste put the data into the gauge-interpolation table. I made assumptions about the engine's air and fuel efficiency, which vary throughout its operating range, so this provided a general starting point.

Determining timing readout in degrees BTDC required more work. To gather timing from the Ford EEC IV system, two 0-to-12-V nominal signals of interest are used: PIP (from distributor-mounted Hall effect sensor and shutter wheel) and SPOUT (spark output command from ECM). The rising edge of the PIP typically indicates 10° BTDC. The rising edge of the SPOUT indicates the actual firing of the ignition coil. With four PIP and SPOUT events per each four-stroke, eight-cylinder engine revolution, timing was computed:

$$degrees = \frac{360}{4\ events/rev} \times \frac{(T_3 - T_2)}{(T_1 - T_3)} + offset \quad [3]$$

where T_1 is the PIP reference rising edge. T_2 is the SPOUT rising edge. T_3 is the PIP next event rising edge. The *offset* variable is the ignition system's offset calibration degrees BTDC (typically 10).

Changing the number of cylinders and offsets is a simple entry in the Excel sheet. Some of the newest electronic engine systems have multiple reference events per spark event that would not be calculated correctly with the present gauge-software configuration. However, timing can be easily read from about any engine, with the addition of a TDC sensor and a spark pickup. Diesel engine-injector timing could be checked with a TDC sensor and a piezoelectric sensor on an injector line.

For easier assembly, each gauge has a rear-mounted five- and 10-position latching connector housing that mates with two detachable pigtails. Output signals are on the five-pin connector with the inputs on the 10-pin connector. An additional terminal block provides a way to connect gauges with external signal and network wires. Power is tapped from vehicle ignition on source. An additional tap comes from the vehicle instrumentation's lamp power for night dimming, which feeds all gauge units and works alongside the built-in photocell dimming circuit.

SMOKE-FREE ENVIRONMENT

With everything connected, it was time to power up and reload the gauges with their calibration. There is a separate Excel file for each gauge containing a unique network address, alarm setting, analog or digital calibration data, and parameters. Since I was using the existing sensors for the analog units, the best accuracy would be obtained with measured ECM reference voltage entered into the spreadsheet.

I had some trouble with intermittent communication errors. After quite a bit of frustration, I discovered the new, lower-power RS-485 driver did not have a strong enough input pull-up for the automatic transmit/receive switching circuit. The test unit worked fine on the bench with the standard driver, but I built six additional units incorrectly because I assumed the new device would require the same pull-up. After stacking another resistor on top of the pull-up, all was well again.

After the configuration data was loaded, I checked the operation and accuracy. Ignition brought the displays to life. The oil pressure gauge flashed a 0 reading because it is configured to flash when display readings are below the alarm's setpoint. Starting the engine showed that all the new gauges closely matched the existing gauges, except for the timing gauge, because I had the SPOUT and PIP inputs reversed. Once I corrected that, timing matched the reading from the Crane Interceptor, an aftermarket engine-management device that connects between the vehicle harness and the ECM. The horsepower function was difficult to verify without a dynamometer, but readings of 6 to 16 hp at idle and 40 to 50 hp at 60 MPH were close to the values in the shop manual.

Some hot weather nighttime driving chased another bug into the light.

Occasionally, the water temperature and oil pressure gauge displays would have all the segments lit in Self-test mode and the serial port would still communicate. Cycling power cleared them, but they reoccurred sporadically. I assumed these errors were caused by corrupt data entering the display driver device. To prove this, I added a display driver reset function in the code every seven-segment display update cycle, which was a quick fix without having to disassemble it. Eventually, I will revise the pull-up resistor values on these serial lines. As I later found out, my alternator had lost a diode and didn't have enough oats to run both the headlights and the cooling fan, which probably pushed things over the edge with more overall electrical system noise. It was nice to have the flashing water temperature gauge give me a bold indication of the less than optimal operating conditions when the fan could not keep up with the cooling requirements. This meant I had to upgrade the alternator and the cooling fan.

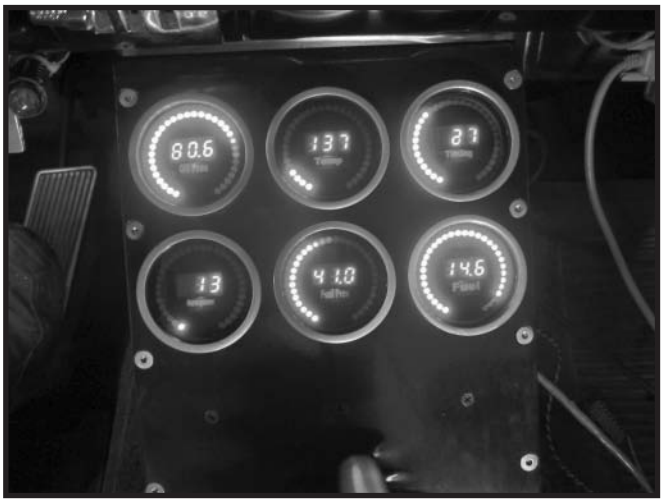


Photo 3—This is a test panel with room for three more units if needed. I plan on adding Min and Max buttons, probably in a single pair with a rotary switch selector configuration.

After these changes, the new gauges have given me reliable and entertaining information. The test panel between the center tunnel and lower dash was supposed to be a temporary unit, but I like it, so it will stay there for a while. Photo 3 shows a snapshot of the panel with the engine idling.

UNFINISHED BUSINESS

There are still plenty of things left to improve. The enclosures may distort in the direct sun if interior temperatures get too far north of the 140°F PVC rating. Although the bezels survived a 160°F paint bake, a higher temperature enclosure is probably a good idea. An application program to fully utilize the data-logging capability is waiting to be written. I am still contemplating whether to develop something laptop-based or a dedicated hardware master node with some type of removable memory. I would like to avoid running a laptop in a vehicle, but it would allow more flexibility.

Currently, only one mode of operation is possible at a time and they cannot be mixed. Functions like miles per gallon or quarter-mile elapsed time would be nice, but they are not possible with the current software configuration. Since all but a few bytes of the 8 KB of program memory are used, some of the multifunction flexibility would have to go to accommodate the functions.

Different display colors are possible and easily mixed with the constant-current driver, but ultrabright green or blue T_1 sizes and seven-segment devices are still expensive. Even without multiple colors, the programmable bar graph with an expanded scaling and flashing capability works well to get your attention while providing some eye candy.

There are plenty of other interesting functions to explore, including a boost indicator/controller, a tachometer with a shift light and rev limiter, torque sensing by measuring shaft twist and resulting phase shift between two sensors, and a G meter using an accelerometer duty cycle output.

In the end, I was satisfied with Microsoft Excel as an effective, low-cost method to configure custom gauge functions and user calibrations. With the configuration implemented with a simple, nonencrypted text-data format, anyone with Excel experience can create new gauge function, configuration, and calibration sheets.

Eric Kesselring works with electron paramagnetic resonance (EPR) and NMR imaging systems at The Ohio State University Center for Biomedical EPR Spectroscopy & Imaging. He enjoys converting wood, metal, and plastics into particles, smoke, and occasionally useful forms, as well as exploring automotive traction limits and discovering new methods for removing countless layers of old paint from areas around his home. You may e-mail him at kaemkb@yahoo.com.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/204.

REFERENCE

[1] Ford Service Publications, *Engine/Emissions-Diagnosis: 1990 Shop Manual.*

RESOURCE

C. Probst, *Ford Fuel Injection & Electronic Engine Control,* Bentley Publishers, 1995.

SOURCES

MS96 Material device
General Electric
www.gesensing.com

MAX7219 Constant current driver
Maxim Integrated Products, Inc.
www.maxim-ic.com

ST00635 Storm sensor
Senstronics
www.senstronics.com

C8051F330 Microcontroller
Silicon Laboratories, Inc.
www.silabs.com

PT5101 ISR
Texas Instruments, Inc.
www.ti.com

WINNERS ANNOUNCEMENT

The DesignStellaris2006 Contest challenged engineers to implement Luminary Micro's Stellaris family of ARM Cortex M3 controllers and Keil's RealView Microcontroller Development Kit in innovative new designs.

Thank you to everyone who participated in this amazing contest! To see the complete projects and more, visit www.circuitcellar.com/designstellaris2006/.

FIRST PRIZE

Handheld Multifunction Scope

The LM3S811-based Handheld Multifunction Scope is an impressive multifunctional oscilloscope that supports several modes: Oscilloscope, Voltmeter, Ohmmeter, Capacitance Meter, Inductance Meter, Frequency Counter, Logic Probe, and Pulse Generator. You can power the well-designed system with three AAA batteries or via your PC's USB port. When you run the scope along with your PC, you can display the traces and readouts on your monitor.

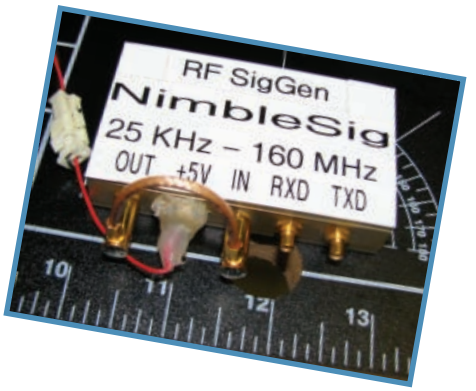
Jingxi Zhang & Yang Zhang
U.S.
jzhang@jupiter.com

"Our goal was to build a scope based on Luminary Micro's LM3S811 evaluation board. I like the Stellaris family of ARM Cortex-M3 processors. The Stellaris devices are physically small, but functionally great. I also like the Stellaris Drive Library. It encapsulates the access and setting of registers for the on-chip peripherals and provides an easier, efficient, and less error-prone mechanism to use the device's peripherals. The rich on-chip peripherals and plentiful GPIO pins of the LM3S811 made it possible to build this multifunctional tool with few external components." —Jingxi Zhang

SECOND PRIZE

RF NimbleSig Generator

The versatile NimbleSig is a compact DDS RF signal generator built around an LM3S811 microcontroller. The system provides a frequency-agile RF output signal source with 1-Hz step resolution from below the VLF to the mid-VHF spectrum segments (25 kHz to 160 MHz). It is also capable of low-level (–50 to 10 dBm) RF power measurement.



Thomas Alldread
Canada
nimblesig@telus.net

"The heart of the DDS generator is the relatively low-cost, 400-MHz AD9859 DDS (alternately AD9951) chip from Analog Devices, which is driven by a 20-MHz TCXO reference clock. The DDS is controlled by Luminary's Stellaris LM3S811 MPU, which receives commands from the host PC via serial ASCII data. I found the firmware development for the MPU to be relatively easy as my Luminary JTAG EVM provides a slick interface for debugging under the Keil µVision IDE. I also found Luminary's easy-to-use API simplified the writing of the source code. As time passes, I expect program development for the Stellaris microcontroller family will become even easier as more example code becomes available to refer to." —Thomas Alldread

THIRD PRIZE

Squawk Box

The well-designed Squawk Box is a VoIP user device that interfaces to Asterisk, a widely accepted open-source soft telephone switch. The LM3S811-based system enables you to participate in worldwide conference networks as well as local ones. The Squawk Box can be used as a public address system, a two-way intercom, or a courtesy phone at a local airport.

Jake Gunderson
U.S.
jgunde@tampabay.rr.com

"My project is a VoIP instrument targeted for use in intercoms, conferencing networks, and operator-assisted courtesy phones. I built the Squawk Box to determine the feasibility of using a low-cost, general-purpose MCU in VoIP instruments. The design is centered on the LM3S811 EVM. I added a Microchip ENC28J60 for network connectivity and a couple of analog ICs for audio conditioning and drive capability. I was very pleased with the Luminary product line, and the provided software libraries sure made things easier. I have nothing but positive things to say about the whole experience from the support from the Luminary user community on the forums to fast responses from the Circuit Cellar team." —Jake Gunderson

To see these projects and more, visit www.circuitcellar.com/designstellaris2006/.

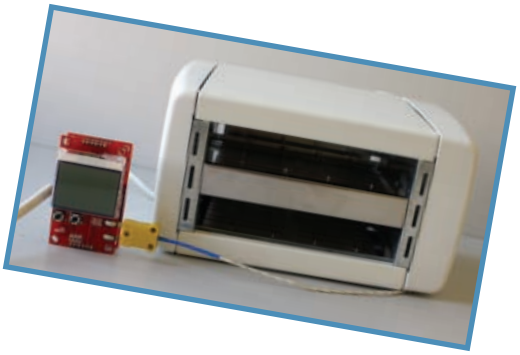
HONORABLE MENTION

"*I designed a reflow oven for SMD boards that uses a regular cheap toaster as the heating element. The contest was great. The design kit provided was easy to use and set up. The support helped when I needed it.*" —Alexandre Guimaraes

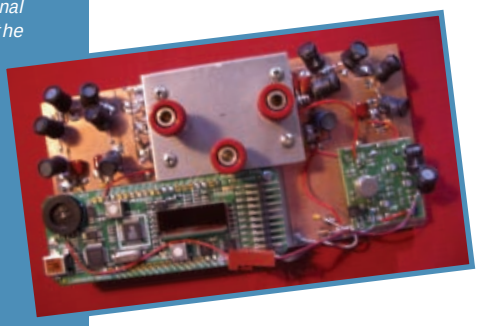
Not Just Another SMD Reflow Oven

This new approach to reflow soldering incorporates a bread toaster with a controller built around an LM3S102. The system features a Nokia 3310 graphical display and buttons for starting the cycle.

Alexandre Guimaraes
Brazil
alexandre@microcode.com.br



"*I built an instrument to measure the noise-figure, in the audio range, of an external audio amplifier. The instrument used the Luminary evaluation board as part of the design. The particular Luminary microprocessor supplied in the evaluation board was perfectly suited to this project. It was capable of doing the entire job with lots of spare capacity. I did not have to do any code optimization or cut any corners. Indeed, I had plenty of excess power, which I used to do things like FFTs and other things which were omitted from the final submitted project in order to stay within the 16K limits of the demo software development license.*" —James Koehler



Audio Noise Figure Meter

The handy Audio Noise Figure Meter is used to measure the relative noise figure of audio amplifiers. The system is built around an LM3S811, which acts as both the audio noise generator and the synchronous power detector.

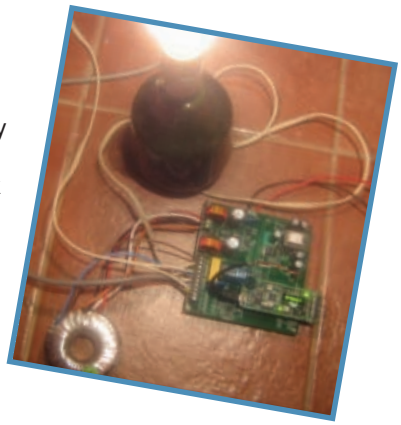
James Koehler
Canada
jark@shaw.ca

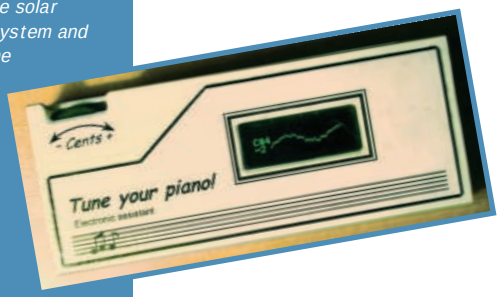
"*The uninterruptible solar power supply is an inverter capable of powering a small AC load direct from a 12-VDC solar panel. The idea behind the project was to present another technological solution of saving on greenhouse gas emissions. Hopefully, this project will inspire some people out there to do their bit to reduce greenhouse gas emissions, or at least find another use for a spare solar panel. I found the development system and evaluation kit easy to use, and the peripheral libraries and examples that were supplied helped to shorten the development time.*" —Lindsay Meek

Uninterruptible Solar Power Supply

The LM3S811-based Uninterruptible Solar Power Supply takes small standby loads off the grid and powers them from solar panels. The supply switches seamlessly back to the grid if the solar power is unavailable at night or during periods of inclement weather.

Lindsay Meek
Australia
lindsaymeek@hotmail.com





Stroboscopic Pocket Tuner

The Stroboscopic Pocket Tuner is a handy tool for fine-tuning acoustic music instruments. You use push buttons to select tuned notes and a potentiometer to adjust a note's pitch. The LM3S811-based system includes an organic LED (OLED) display with 96 × 16 pixel resolution.

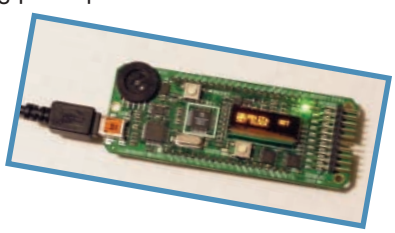
Ilya Mamontov
Russian Federation
illinoys@narod.ru

"*ArmExe is a compact real-time operating system (RTOS) specifically designed for the Luminary Micro Stellaris ARM microcontrollers. It was built to take advantage of the new ARM Cortex-M3 architecture, the Thumb-2 instruction set, and the Stellaris-specific processor features. As this was my first ARM project, it was a way to learn a lot about the Cortex-M3 architecture, the Keil environment, and the specifics about the Stellaris microcontroller's features. It also left me with a full-feature familiar RTOS as a platform to develop advanced projects for this CPU family.*" —Naubert Aparicio

ArmExe: An RTOS for Luminary Micro Stellaris Cortex-M3 Microcontrollers

The ArmExe is a useful RTOS for the Luminary Micro Stellaris ARM Cortex-M3 family of microcontrollers and Keil's µVision environment. The multitasking preemptable executive uses task priorities, separate tasks and kernel stacks, and execution privileges. It handles all interrupts with priorities, supports nested interrupts, and implements binary semaphores and events as the resource-sharing mechanisms.

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



To see these projects and more, visit www.circuitcellar.com/designstellaris2006/.

Multitone Music Keyboard

Designed for musicians to experiment with alternative scales, this innovative LM3S811-based multitonal instrument features three inexpensive keyboards. The keys are combined in various ways to support 12, 19, 24, and 31 tone scales.

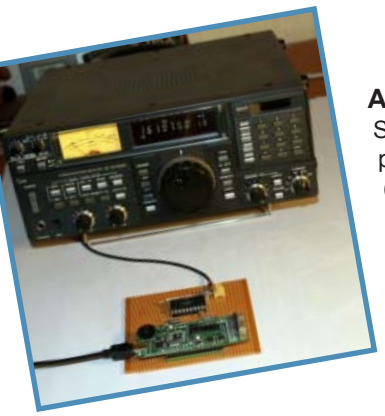
Henry Pfister
U.S.
henrypfister@hotmail.com

"This project was to create a keyboard design for multitone music inputs. It implements (in addition to the 12 white and black key tones per octave) 19, 24, and 31 tones per octave. This is an ideal project for a computer to reprogram a keyboard as needed. The Stellaris MCU was very adaptable to this simple project that only reads and writes ports to detect key closures. I am very impressed by the other winning projects that involved such a large amount of work. I enjoy design contests, and I have entered a number over the years. In general, they motivate me to actually finish something that is just an idea rattling around my head." —Henry Pfister

The Virtual Bicycle

The LM3S811-based Virtual Bicycle provides a virtual bicycling experience for riders. The system provides speed and direction sensing, virtual wind control, and pedaling resistance control. It also features a software interface to Google Earth.

Jeffrey Berezin, Paul Gelling, and Brian Post
U.S.
jeff.berezin@virtualbicycle.com



A Stellaris-Based AIS Decoder

Ships with cargo weighing over 300 tons must broadcast their positions using the universal automatic identification system (AIS). This well-designed, LM3S811-based system decodes AIS transmissions in software. The transmissions are then transferred to a host system using the NMEA protocol.

Peter Baston
U.K.
ais@petebaston.co.uk

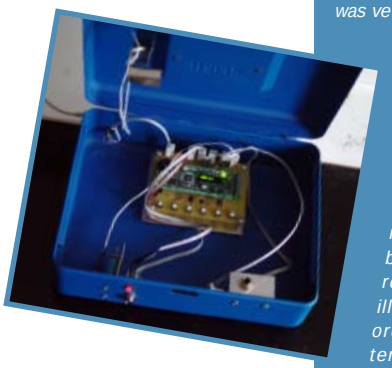
"My project is a decoder for the Universal Automatic Identification System (AIS). This is a system which is used by ships at sea to broadcast their position and other information useful to navigators. As a radio ham and active developer of embedded systems, I have always been interested in decoding 'off air' data. This contest provided an opportunity to decode a new broadcast media using a state-of-the-art microcontroller." —Peter Baston

PGKey

The PGKey is an electronic lock and key system that uses AES encryption for key verification. The system features an LM3S811 for the lock and an LM3S102 for the key. The lock is accessed by the key via two ports: Admin and User. The former is used for administration functions. The latter is used to open the lock.

Alex Wolfing & Charlie Krauter
U.S.
a13x5pam@hotmail.com

"PGKey is an electronic lock and key system. After looking at the features on Luminary chips, we decided they would work well for encryption, and security has always been one of my interests. The system comprises two pieces of hardware, the lock circuitry and a key 'fob.' The fob is physically inserted into the lock and the two go through a verification process using 128-bit Rijndael encryption. I used only the LM3S811 evaluation board, but was very happy with it. I especially like the OLED display; it makes debugging much easier." —Alex Wolfing



Shock-N-Awe

Powered by an LM3S828, this interesting table design responds to audio stimuli and produces amazing visual effects. The design features 96 tricolor LEDs and four microphones. The table can generate a single color display or exciting lighting effects.

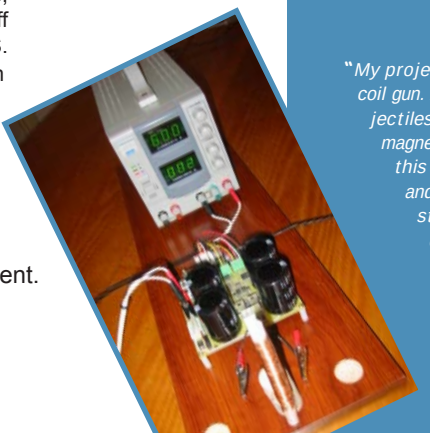
Benjamin Wolpoff, Thaine Hock, Matt Corne,
Chad Harvey, & David Wolpoff
U.S.
bwolpoff@fivemanconspiracy.com

"Shock-N-Awe is an audio-reactive illuminated coffee table. The table utilizes custom electronics to determine point-of-impact of any object being placed upon the table, and it reacts visually in response to impact by illuminating patterns with the 96 tricolored LEDs located at the table's perimeter. Using a Luminary LM3S828, we sample the audio data from four microphones affixed to the corners of the acrylic and determine difference of arrival times of the sound. The Luminary Stellaris family was good to work with because it offered a strong set of peripheral modules in an easy to layout package." —Benjamin Wolpoff

Coil Gun Controller

This educational system works as a controller and data recorder for a dual-coil coil gun. You can use the LM3S811-based controller to experiment with various aspects of the coil gun, including timing and voltage control. You can also measure the real-time voltage and current.

Andrew Sterian
U.S.
steriana@gmail.com

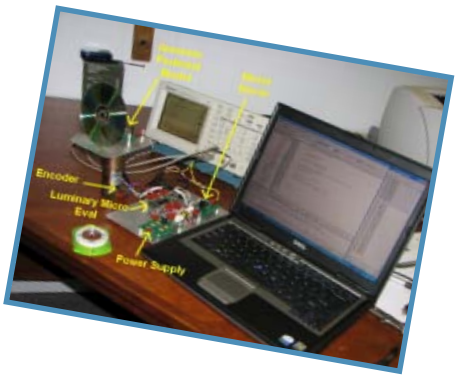


"My project is a controller for a dual-coil coil gun. A coil gun can launch metal projectiles using the principles of electromagnetism. My main goal for doing this was to create a demonstration and testing system for engineering students in a Circuit Analysis class that I teach. Developing an embedded system project is a lot of fun, and it is even more fun to be able to teach others about what you learned by doing it." —Andrew Sterian

HONORABLE MENTION

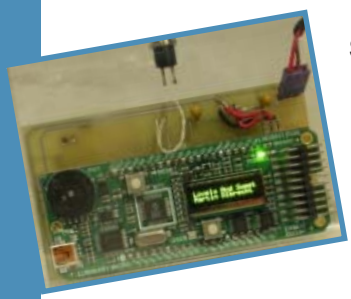
Automatic Antenna Positioner

This well-designed, LM3S801-based system autonomously points an antenna (base station) toward a target (remote transceiver) indicated by a latitude-longitude pair. The microcontroller reads the quadrature encoder signal from the motor and the current antenna heading from the sensor. It then processes the data and sends the PWM result to a motor driver.



Hoa Phan, Nghia Tran, & Vincent Dinh
U.S.
hphoa@yahoo.com

"I am a jazz lover and electronics engineer. The contest was a good way of putting it all together. The idea was to have web page information available in real time without having to have a PC connected all the time. The jazz station in my area, Swissjazz, plays music 24 hours a day, but the music information is only available in their web page. Or was! Now the current music (also previous and next) is available in a display on top of the radio!" —Fernando Jordan



Swissjazz List

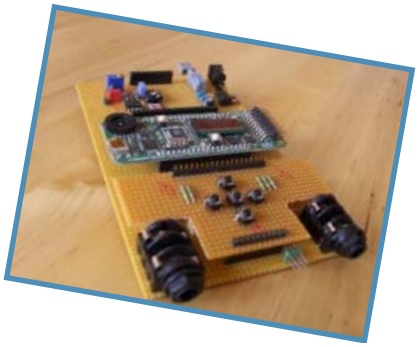
The Swissjazz system retrieves artist and song title information from an Internet radio station's web site and displays it on the LM3S811 evaluation board's display. The information updates every time a new song begins. The system connects to the Internet via an 802.11b wireless CompactFlash card.

Fernando Jordan
Switzerland
jordanf@swissonline.ch

"I built a basic guitar effects processor. The idea came to me after I studied the documentation about the built-in peripherals in the LM3S811 microcontroller. I developed the system step by step. The circuit parts were built on different blocks and were plugged around the LM3S811 evaluation kit on a test board, which gave me quite high flexibility to change or redesign any part of it." —Robert Papp

StellarisGFX

The StellarisGFX is a sound effects processor for guitarists. The LM3S811-based system's keyboard is used for navigating in the user menu. You can set the volume level and select effects such as overdrive, tube overdrive, and distortion.



Robert Papp
Hungary
probert@edaboard.com

SD Card Display Controller

The LM3S811-based SD Card Display Controller enables you to input data to portable devices. This system regularly reads the temperature and relative humidity from an on-board sensor. The host device is a tablet PC that runs a picture slideshow program.

Sylvain Davaine
France
davaine_s@yahoo.com

"The purpose of my system is to replace an SD card. It has an SD card pinout replica on its PCB, which plugs itself into a host. The system regularly does a temperature and relative humidity measurement and writes it into an SD card, a real one this time. I'm impressed by the high quality of the other projects as much as by their originality." —Sylvain Davaine

Timecode Signal Generator

The well-designed Timecode Signal Generator supplies the audio signal for syncing recording machines. The compact system creates an audio signal that conforms to the SMPTE standard, generates MTC, and is capable of being synchronized with an external reference video.

Fabian de la Fuente
Argentina
fdelafuente@speedy.com.ar

To see these projects and more, visit www.circuitcellar.com/designstellaris2006/.

Flash Drive Connection

Add a Flash Drive to Your Next Design

Interfacing flash memory drives just got a lot easier with the Vinculum ASIC. The 48-pin LQFP includes a DOS-like command line interpreter that can read and write USB flash memory drives.

The flash memory drive is probably the most successful USB product. Its capacity has increased exponentially over the past few years while its price has fallen at a similar rate. (You can buy 1-GB drives for less than $10.) Up until now, they have been excluded from embedded projects due to the complexity of interfacing them, but this article will change all that!

My web site receives many requests from readers who "just want to connect a flash drive" to their electronic creations. The issue, of course, is that a flash drive is a USB device, and to control it, you need a USB host controller. The USB specification deliberately put most of the communications complexity within the host controller, since there is never more than one in a system. The structure also enables USB devices to be simpler and therefore less expensive.

A flash drive is a mass storage class (MSC) device. Although you can download the specifications for free at www.usb.org, they are not easy to understand. MSC specifications define only basic track/sector and read/write operations. You also need to understand the specifications for the FAT file system (used on all commercial flash drives) to be able to read and write data. The amount of information you need to understand to be able to "just connect a flash drive" is overwhelming. You need a component that implements all of these specifications for you. (After all, they are industry-agreed specifications that you have no freedom to change anyway; you just want to use them.)

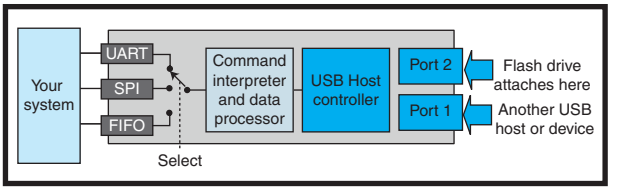


Figure 1—This block diagram of the Vinculum chip shows the command line interpreter that provides a DOS-like front end to a USB host controller. The Vinculum supports two USB ports that can be programmed as host or device ports.

INTRODUCING THE VINCULUM

Future Technology Devices, a pioneer and advocate of the idea that "you don't have to fully understand USB to use it," has introduced an application-specific integrated circuit (ASIC) called Vinculum, which provides a DOS-like command line interpreter front end to a flash drive. This single 48-pin LQFP part encapsulates all of the functionality required to read and write standard USB flash drives. Internally, it is implemented as a microcontroller, with specialized I/O devices, running embedded firmware, but you do not need to know this (see Figure 1). Vinculum's command line interface is accessed via a UART, an SPI, or a FIFO. It actually supports two USB ports. Each can be programmed to be a host or a device, but my first series of examples will assume a single host port with a connected flash drive.

The Vinculum ASIC is available on several preassembled modules, which speed up development. A VDIP1 plugs into a 0.1" prototyping board (see Photo 1). The 24 pins provide access to the three interfaces and two jumpers select the interface. The UART interface is the easiest to use. It runs at 9,600 bps, although this can be increased to as much as 3 Mbps at run time with monitor commands. For even higher serial performance, the SPI can be selected. The Vinculum is an SPI slave and your application processor can clock it at up to 12 MHz. The FIFO interface is selected for the highest throughput. The bidirectional, byte-wide interface is controlled by read/write strobes and FIFO full/empty signals. A 128-byte TX FIFO and a 256-byte RX FIFO can sustain data rates up to 1 MBps. The DIP module also includes a 12-MHz crystal and reset circuitry that enables the monitor to start running at power-up.

The Vinculum is microcontroller-based. The command-line interpreter is implemented by a program running from the on-chip 64 KB of flash memory. The monitor firmware can be

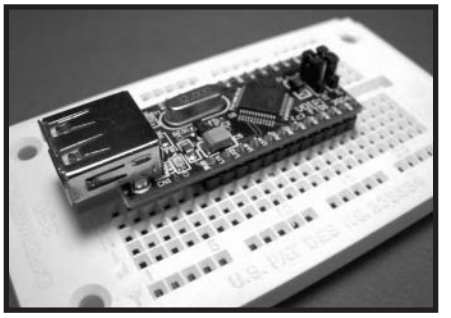


Photo 1—The Vinculum is available in a variety of modules. Here is the 24-pin DIP module that provides access to all three monitor interfaces and both USB ports. This module plugs directly into a 0.1" prototyping board.

updated using PC software or a firmware update file can be incorporated on the flash drive. FTDI has several sets of firmware that match its array of modules (check www.vinculum.com for the latest versions). New features are added to the monitor all the time (including support of other USB classes, such as HID and printer), which are included using one of the firmware-update methods. The monitor includes commands that allow the detection of attachment and removal of a flash memory drive so your application processor can incorporate the "hot-swapping" of USB devices (the examples use these features and more).

I connected a USB-to-serial cable to my PC that is running HyperTerminal at 9,600 bps to show you how easy a Vinculum is to use. I used FTDI's TTL-R232-3V3 cable because it uses TTL levels rather than RS-232 voltage levels and provides 5 V to power a device. I connected the cable to a Vinculum module called VMUSIC (see Photo 2). Ignore the name for now because you won't use the "music" part until the second example. For now, it is a Vinculum chip mounted on a board with a convenient serial connector. Choose any of your flash drives and plug this into the USB A socket of the VMUSIC board (see Photo 2).

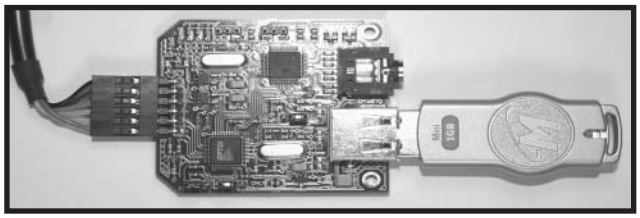


Photo 2—Take a look at a Vinculum development board interconnecting a USB-to-serial cable with a flash memory drive. In this configuration, the Vinculum operates as a serial-to-flash memory drive adapter.

The board will sign on and offer a "D: >" prompt. In HyperTerminal, enter "DIR" and (presto!) the contents of your drive are displayed. Now enter the following commands:

```
OPW test1
IPA
WRF 12
Hello World!
CLF test1
```

The command opens a file called test1 for writing, changes the Input mode to ASCII, and writes 12 bytes. "Hello World!" is the data that is written, and finally the CLF command closes test1.

Next, remove your flash drive and connect it to your PC, Mac, or Linux system and open test1. Notice that the data written by the Vinculum is present. Edit test1 to add the message "Hello from my PC, Mac, or Linux." Next, reattach the flash drive to the Vinculum board and enter RD test1. Voila, new text is displayed!

Now, stop and think about what you accomplished. You wrote, read, and exchanged data files between a PC, Mac, or Linux system and an embedded system using a flash drive. You didn't even have to learn USB, the MSC specification, or the FAT file system. It was as easy as entering DOS-like commands on a serial connection.

The hard work of understanding and implementing three industry specifications has been done already, allowing you to focus on your application task.

The Vinculum monitor powers up in Extended Command mode, where all the commands and data are in ASCII. Some of the commands are summarized in Table 1. (The list represents about 25% of the available monitor commands. Refer to the firmware guide for more details.) The monitor can be switched into Short Command mode, where binary commands and data are exchanged. The VMUSIC board provides access to only the UART connection, but this is enough for my first set of examples. The Vinculum is also available in an OEM 24-pin DIP module, which provides additional access to the SPI port, the parallel port FIFO, and the other USB port (see Photo 1).

Your mind must be racing—the things that I can do with this!

Command	Directory operations
DIR	Lists the current directory
MKD <name>	Creates a new directory <name> in the current directory
DLD <name>	Deletes the directory <name> from the current directory
CD <name>	The current directory is changed to the new directory <name>
CD ..	Move up one directory level
	File operations
RD <name>	Read file <name>. This will return the entire file.
OPR <name>	Opens file <name> for reading with RDF
RDF <size>	Reads <size> bytes of data from the current file
OPW <name>	Opens file <name> for writing with WRF
WRF <size>	Writes <size> bytes of data to the end of the current open file
CLF <name>	Closes file <name> for writing
DLF <name>	This will delete the file from the current directory and free up disk space
VPF <name>	Plays an MP3 file. Sends file to SPI then returns.
REN <n1><n2>	Renames a file or directory
	Management commands
SCS	Switch to the short command set
ECS	Switch to the extended command set
IPA	Input data values in ASCII
IPH	Input data values in Hex
SUD	Suspend the disk when not in use to conserve power. The disk will be woken up automatically when a disk command is sent to it.
WKD	Wake disk and do not put it into suspend when not in use
SUM	Suspend monitor and stop clocks
FWV	Get firmware versions
FS	Returns free space in bytes on disk

Table 1—Some of the monitor's DOS-like disk and management commands are shown in this table. Additional commands that manage devices attached to the other USB port (not used in my examples) are listed in the firmware guide available at www.vinculum.com.

PORTABLE GPS TRACKER

I have more equipment in my lab

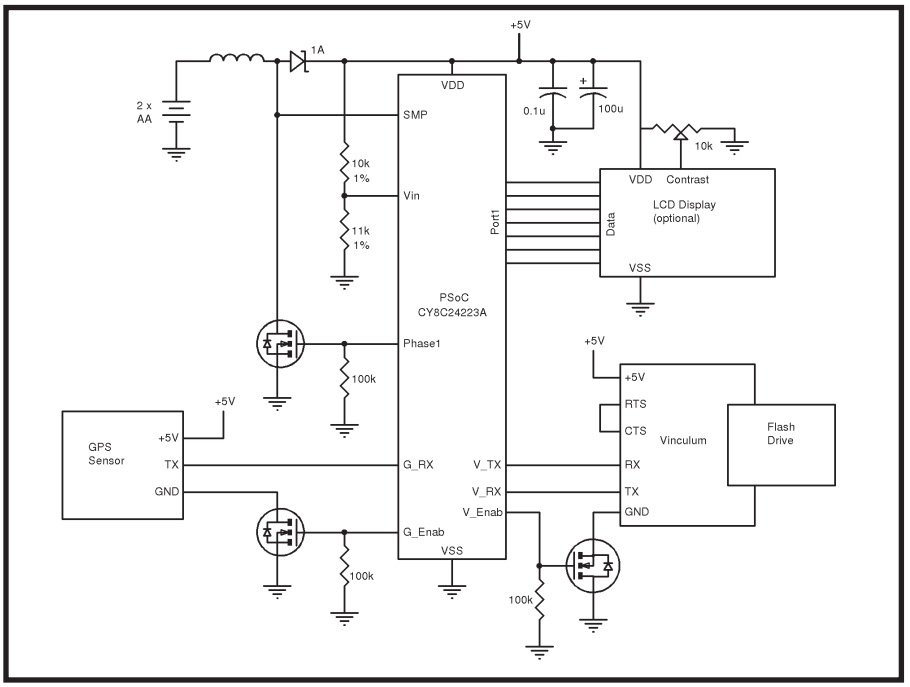


Figure 2—The GPS tracker is battery powered and the design allows the firmware to power down the GPS sensor and/or the Vinculum and flash memory drive independently to conserve battery life. Analog and digital blocks within the PSoC are used to generate and control the power needed by the other modules.

than most Radio Shack stores, so I could choose from a wide range of projects. I first grabbed a Garmin International GPS 18 LVC tracker with a serial interface. Note that the GPS's RS-232 interface runs at a default 4,800 bps with TTL signal levels and inverted signal polarities. This is not a problem for FTDI's cable, which is fully programmable. Using the MPROG utility from the FTDI web site, I reprogrammed the cable to inverted RX and TX. I soldered a connector to the Garmin cable to match the FTDI cable socket and then plugged the two together. Next, I connected it to my PC running HyperTerminal at 4,800 bps. At power-up, the GPS is quite verbose and starts sending multiple reports after about 3 s.

I now had two serial devices (a serial-to-flash drive and a serial-to-GPS receiver) that operate independently with HyperTerminal. I replaced the PC with a microcontroller that has two serial ports (a significant cost savings!) and replaced HyperTerminal with some firmware on the microcontroller.

I chose a Cypress PSoC for my target microcontroller because it has firmware-configurable hardware that enables me to solve a wide range of problems with a single device. I develop and debug with a high-end PSoC that has ample analog and digital resources. Near project completion, I can select a less expensive device within the same family. For the first example, I used the Cypress PSoC evaluation board with a VMUSIC board and a GPS already attached (see Photo 3). This development board uses the feature-rich Cypress CY8C29466 PSoC device.

I configured two serial ports inside the PSoC and wrote a small program that filters the GPS reports and then writes the data to the flash memory drive. First, I initialized the GPS to reduce the number of reports it sends. During runtime, the PSoC spends most of its time in a low-power sleep state. It periodically wakes up and checks if a flash memory drive is present. If one is not present, it powers down, since there is no point collecting GPS data if there is nowhere to store it. If a flash memory drive is present, I record the $GPRMC report once the position status is valid. The GPS sends a report every second, but I record the data at a slower, selectable rate. Depending on the amount of buffer RAM in the selected PSoC device, I collected a series of samples before writing them to the Vinculum. The Vinculum buffers a sector's worth of data before writing to the flash memory drive. I found a large variation of power draw from the flash memory drives that I tested, so I programmed the Vinculum to keep the drive suspended except when it was accessing the drive. This saved a lot of power. The final and completed project files can be downloaded from the *Circuit Cellar* FTP site. I included an LCD that I used to simplify development by also displaying the GPS position during debugging.

For portability, the example needs to be battery operated. This is easy for the PSoC because it contains an integrated charge pump that enables it to operate from a single 1.5-V battery. I used some of the programmable blocks and a few external components to generate 5 V for the Garmin and Vinculum modules. The final schematic of the first example is shown in Figure 2. I chose to enable each module's attachment to the 5-V generator because they can be individually powered off to save battery power. I measured flash memory drive

Photo 3—The first example uses a Cypress PSoC to interconnect a Vinculum chip with a Garmin GPS sensor. The character LCD is used for debugging, but it could be incorporated into the final design.

currents between 50 and 300 mA, so I programmed the Vinculum to use Suspend mode that powers the drive only when it is being accessed. Deciding when to power the GPS module is a trade-off, depending on the period between samples. It will also vary depending on your application. If the period between samples is long, you can save battery power by turning it on only during its satellite data-acquisition phase. My application waits for a valid position report and then powers down the GPS.

I have used this design in a variety of GPS applications. The most obvious design (recording a vehicle's movement) was easy to implement. Just attach a battery and a flash memory drive and insert it into the vehicle to be tracked. Later, remove the flash memory drive and insert it into a PC that is running Garmin map software. Data from the flash memory drive can be imported into the map and a history of the vehicle's movements is displayed. If you have a fleet of vehicles, such as delivery trucks, you can hardwire the design into the vehicles, so you don't have to deal with the voltage generators. Each week you could swap out the flash memory drive and the recorded data could be used to analyze delivery routes with a goal of optimizing your fleet's travel. The digital tachograph would pay for itself in fuel savings.

My favorite GPS application, however, is GeoTagging my photographs. While on vacation, synchronize the time on your digital camera with the Garmin GPS tracker, attach a flash memory drive, and place the tracker in your camera bag. As you travel around taking pictures, the tracker records your X, Y, and Z coordinates. When you get home, import your photos and the GPS data into your PC and use software such as GPS-Photo Link, WWMX, or RoboGEO to match your photos to a location using the timestamps. Use Google Earth to display 3-D maps of your vacation with your photos. This presentation will make viewing vacation pictures a memorable event!

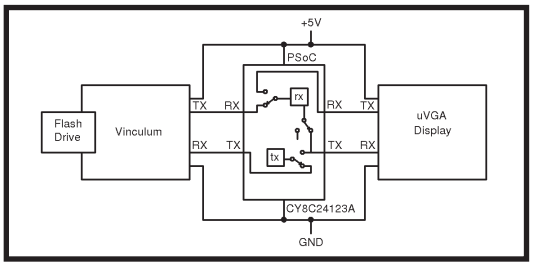


Figure 3—The PhotoFrame can be implemented with the smallest eight-pin PSoC. The firmware uses a single UART and dynamically switches its connections between the I/O pins.

ACTIVE PHOTO FRAME

The first example showed how ASCII data could be logged onto a flash memory drive using a Vinculum. The second example is a binary data example, but it follows the same design philosophy. My example is an active photo frame, but it can evolve into an interactive display (see Figure 3). I have a PSoC that reads image files off a flash memory drive using a Vinculum and the PSoC displays this image. If a matching MP3 file is also present on the flash memory drive, the Vinculum plays it using VMUSIC's VLSI VS1003 codec. This could be music or a person speaking. A PC creates images and MP3 files, which are copied onto the flash memory drive. The PSoC/Vinculum-based player then runs the show.

The download package includes the source code as a PSoC Designer project. This will allow you to customize the examples to match your requirements. Cypress has over 100 application notes that describe building blocks, which can be used in your design. The PSoC could scan buttons, which could use CapSense technology. The application program would use these button inputs to navigate through the images and MP3 files. Or the PSoC could support a touchscreen with a few of its configurable, analog, and digital blocks. This could be a simple resistive screen overlay or a more reliable CapSense implementation. An active photo frame is the base example, but an interactive display, which could be used in stores, museums, product demonstrations, and art galleries is a straightforward design extension. A series of flash memory drives in English, Spanish, Japanese, and more could be used to create a more universal solution.

The size of the graphical display will determine the complexity and choice of the PSoC. Since this is an article about embedded flash drive applications, I chose the simplest display to implement here and I will cover interfacing to a (Q)VGA LCD panel in a future article. I found a serial interface µLCD at Dontronics. I was very impressed with how easy the 128 × 128 16-color displays were to use. The 1.5″ × 1.5″ displays are not expensive. You should get some and experiment with them. I'm sure you will find many uses for them, just as I did. I found the OLED displays much better to look at when compared with the LCDs, but the firmware to drive both displays is identical. From a hardware perspective, just swap the GPS serial connection with a µLCD connection (see Photo 4).

The µLCD module is a very capable subsystem that supports graphics rendering and several fonts. My example uses about 5% of its capability because I just download images to it. The µVGA images are 128 × 128 × 16-bit color. The download package includes a PC application called Imager, which converts 24-bit color BMP files into this format and adds a variety of transitions. In this example, the images will be copied to a flash memory drive and called nnn.img (nnn = 000 to 999). MP3 files are also created for each image and they will be called nnn.MP3. (They could be your favorite songs renamed.)

My base application starts by looking for 001.img and copies it to the dis-

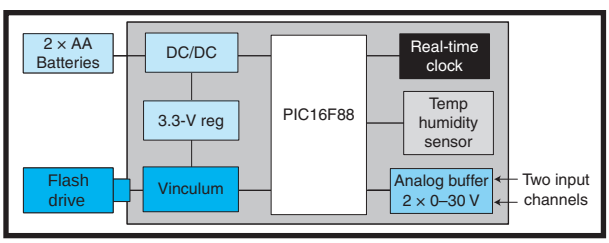


Figure 4—This block diagram of DLP's VLOG development kit shows the resources available on the board. This complete kit—including software, design tools, and a CCS C compiler—was designed to make it easy to learn and use the Vinculum chip.

PCF8574 Remote 8-bit I/O Expander

Designed for two-line bidirectional bus (I^2C), 2.5V to 6V VCC operation, and provides general-purpose remote I/O expansion for most MCU families via the I^2C interface.

Texas Instruments

mouser.com/ti/a

PIC24 MCUs

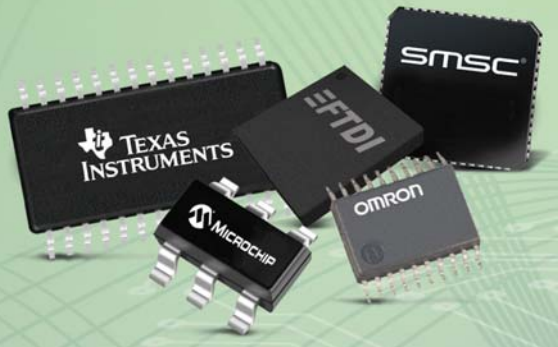
High integration 16-bit MCUs that meet real-time control demands.

Microchip

mouser.com/microchip/a

Newest

The Semiconductor Products For Your Newest Designs



- The ONLY New Catalog Every 90 Days
- NEWEST Products & Technologies
- Over 800,000 Products Online
- More Than 330 Manufacturers
- No Minimum Order
- Fast Delivery, Same-day Shipping

mouser.com (800) 346-6873

MOUSER ELECTRONICS®
a tti company

The Newest Products For Your Newest Designs

B6TS – Capacitive Touch Sensing ICs

Capacitive touch sensing IC highly tolerant of its working environment.

OMRON ELECTRONIC COMPONENTS

mouser.com/omron/a

LAN91C111 Ethernet Controllers

Facilitates the implementation of third generation Fast Ethernet connectivity solutions for embedded applications.

smsc SUCCESS BY DESIGN

mouser.com/smsc/a

Vinculum VNC1L

Embedded USB host controller IC with two independent USB 2.0 low-speed/full-speed and USB host/slave ports.

FTDI Chip

mouser.com/ftdi/a

The NEWEST Semiconductors | Passives | Interconnects | Power | Electromechanical | Test, Tools & Supplies from Mouser Electronics

play. If it finds 001.MP3, it will play it or it will wait 60 s (easy to modify) before moving on to 002.img. The application increments through the filenames until one is not found. It then restarts at 001.img. To change the photos and music, swap the flash memory drive. The complete PSoC project is posted on the *Circuit Cellar* FTP site. It is easy to expand the design to add functions. I plan to add a feature-rich alarm clock once I get some spare time. It would be easy to make this battery powered like I did with the first example; however, displays consume a lot of power, so I would also add a battery charger. A battery charger uses a few of a PSoC's analog and digital resources, a few FETs, an inductor, Rs, and Cs. The design extension is covered in detail in Cypress's application note collection.

PORTABLE DATA LOGGER

I know many of you use a PIC microcontroller, so I chose to use one in my third example. I didn't like the weird instruction set, but my colleague Don Powrie of DLP Design introduced me to the Custom Computer Services toolset, which makes PIC designs fun to create! I used to be a staunch advocate of using only assembler code for microcontrollers. I argued that a compiler would always generate larger object code than my tuned assembler code. Now, these microcontrollers are available with 16 KB, 32 KB, and more flash memory! So, what is the point of saving a few hundred bytes when you have not used over half of the flash memory space. C code is also much easier to write and debug when compared with assembler code.

The CCS compiler was designed to create optimized code for PIC microcontrollers. In addition to the standard features you would expect from a quality

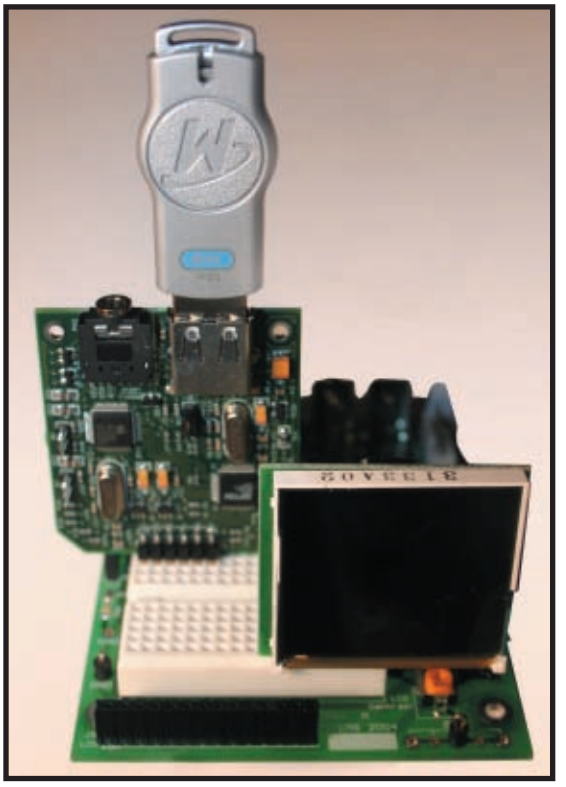
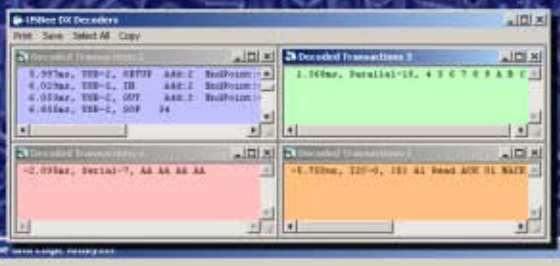
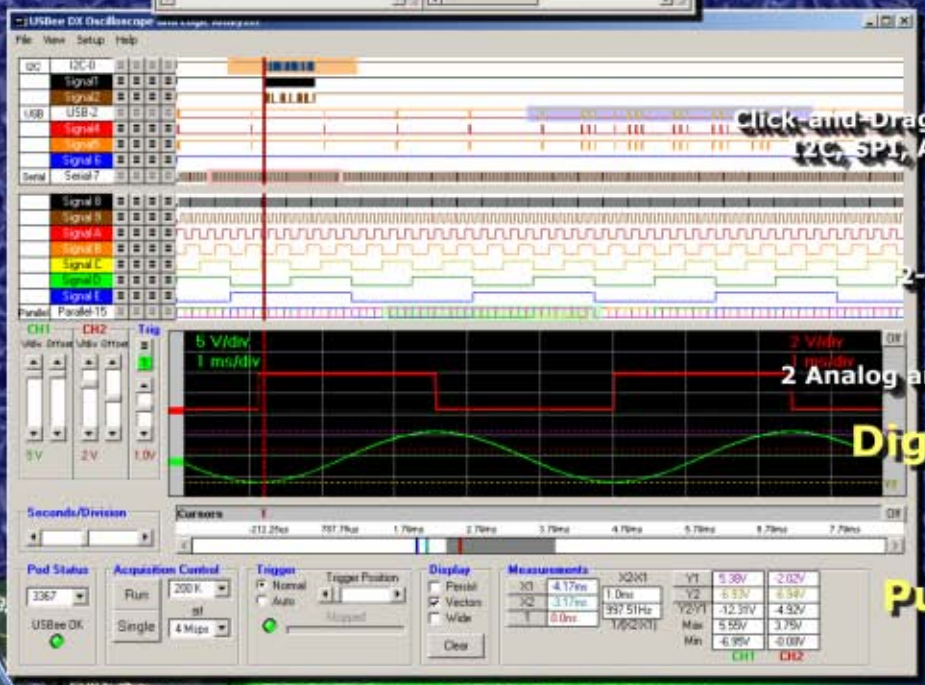


Photo 4—The second example is an active photo frame that displays images from the flash memory drive on a µVGA display and plays MP3 files through headphones. The same design can be used to drive a larger display for a low-cost information terminal.

USBee DX





Oscilloscope/Logic Analyzer

2 Analog and 16 Digital Signals
Up to 24Msps, 100+ Million Samples
Dual 8-bit ADC, +/-10V inputs

Bus Decoding

Click-and-Drag Instant Decode of Bus Transactions
I2C, SPI, ASYNC, CAN, USB Low and Full Speed
I2S, SM Bus, 1-Wire, PS/2

Digital Voltmeter

2-Channel, +/-10V, 8-bit ADC, Logging

Data Logger

2 Analog and 16 Digital Signals Plus Timestamp

Digital Signal Generator

16 Digital Outputs, Up to 24Msps
Playback of Logic Analyzer Traces

Pulse Width Modulator

16 User Controlled PWM Channels

Frequency Counter

16 Channel Counter to 24MHz

Frequency Generator

Generates Sets of Common Frequencies

I2C Controller

Control I2C Devices Using Simple Text Scripts

Remote Controller

Easily Control Your Hardware Using Your PC

Pulse Counter

16 Channel Pulse Counter With Gating Control

plus the USBee Toolbuilder Source code and Library

Create Your Own Applications to Control the USBee DX

Actual Size

Data Extractors

Optional Software Modules for the USBee AX-Pro and USBee DX

Continuous Real-Time Embedded Bus Data Streaming

Store Bus Data to Disk or Send to Your Custom Application

Capture and Process Entire Test Sequences

Parallel, Serial, I2C, USB, ASYNC, CAN, SMBus, SPI, I2S, 1-Wire

USBee.com (951) 693-3065
support@usbee.com

Register by July 24th for best conference rates.
www.embedded.com
Use Priority Code: U120

WITNESS THE POWER OF BRAINPOWER

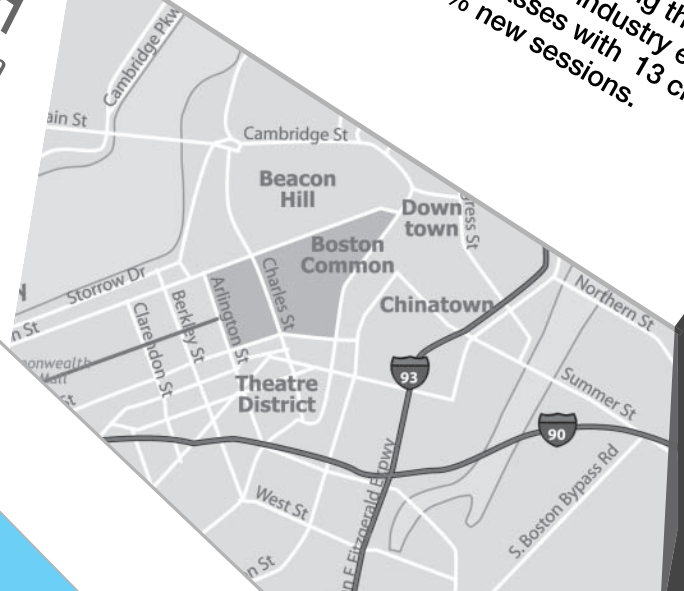
WHEN BRAINS COLLIDE

Embedded Systems CONFERENCE BOSTON

HYNES CONVENTION CENTER, BOSTON, MA
SEPTEMBER 18-21
www.embedded.com

Be there at ESC. Join the power of brain power in action. These, after all, are engineers. On their own, they tackle the toughest problems. But when these brains collide, any challenge confronting humankind can be taken down.

At ESC Boston, you can put your brains against the best in the world. Including 180 leading exhibitors showcasing the latest technologies. Learn from industry experts in 90 in-depth design classes with 13 class tracks and over 60% new sessions.



Is this a must-attend event?
Let's just file that under no-brainer.

CMP
United Business Media

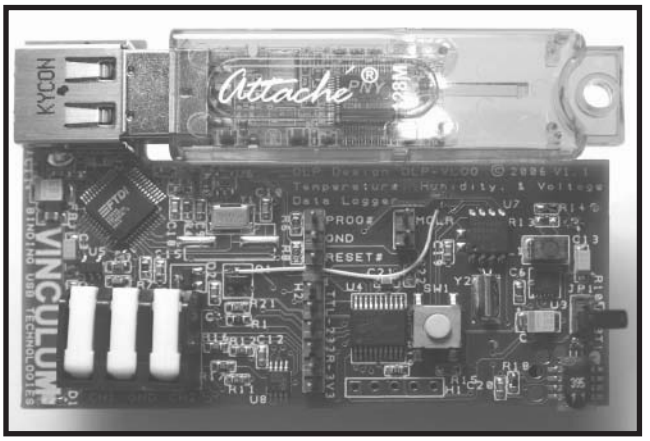


Photo 5—The third example highlights a design kit offered by DLP Design. It uses a PIC microcontroller to measure time, temperature, humidity, and two external analog voltages. It uses the Vinculum to save these values to a flash memory drive.

C compiler, it includes built-in functions to support a PIC microcontrollers's on-chip features. A good example is the #use RS-232 directive. There, you specify that you need to use a serial port and give the compiler details, such as the data rate and the I/O pins that will be used for TX and RX. If the chosen PIC has a hardware UART, then the compiler will use this for `printf` and `scanf` functions or else it will include subroutines to manage the low-level bit manipulations for you. Your main program uses `printf` statements like before. The CCS compiler also contains built-in functions to drive the on-chip ADC and the real-time clock. This way you can focus on what your program is doing instead of the lower-level how.

Don designed the battery-powered data logger to demonstrate the capabilities of the Vinculum (see Photo 5). The design's block diagram is shown in Figure 4. The example program uses RS-232 to control the Vinculum, which writes data to the flash memory drive and the hardware connection is a standard four-wire serial port using TX, RX, RTS, and CTS.

The Microchip Technology PIC16F88 runs an application that has access to a flash memory drive with Vinculum, a real-time clock, a temperature/humidity sensor, and two analog input channels. A connector for FTDI's TTL-232R-3V3 cable is included because it is a connector for a PIC debugger, such as CCS's ICD-U40 unit.

The program first checks to see if a flash memory drive is present. If one is not found, the PIC goes back to sleep since there is no point collecting data if there is no where to store it. Once a flash memory drive is found, the PIC starts a data-collection cycle. It reads the time from the Maxim Integrated Products DS1302, then the two analog signals and the battery voltage, and finally the temperature and humidity. The data is written on the flash memory drive and the system goes back to sleep to save battery power. The cycle repeats while a flash memory drive is present and the battery is charged. The flash memory drive may be removed at any time and the data collected may be analyzed using a PC, Mac, or Linux-based system.

The source code for the application is available with the development kit so you can customize the data and the time interval between samples. Don designed the board as an evaluation tool for Vinculum designs, but the battery-operated, portable data logger would be a great fit in many projects as is.

EMBEDDED FLASH DRIVE

I hope that I have shown you that projects built around a flash drive are easy. The Vinculum ASIC encapsulates all of the required industry-standard specifications and presents a simple DOS-like command-line interface that is accessed via a serial port (or SPI or parallel FIFO). Add your favorite microcontroller with an application program to control the Vinculum. I presented a few projects to fuel your imagination. Source code for my examples using a Cypress PSoC are available on the *Circuit Cellar* FTP site. If desired, this code can be ported to a different microcontroller. Your project can collect data that will be analyzed on a PC, Mac, or Linux-based system. It can also be used to redistribute data that was created on a CPU via lower-cost platforms. Project data may be updated by simply swapping flash drives.

If you can read and write to a serial port then you can read and write data files on flash memory drives with Vinculum. I will be interested to hear about projects where you have creatively used a Vinculum and a flash memory drive. Happy developing.

John Hyde has been involved with USB since its inception. After working at Intel Corp. for 25 years, he left in 2002 to start his own consulting firm. John has been involved in many of the products that you can buy in your local electronics store. He has also written several books on USB and is currently working on a USB Projects By Example *book. He can be contacted via his web site at www.usb-by-example.com.*

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/204.

SOURCES

ICD-U40 In-circuit debugger/programmer
Custom Computer Services, Inc.
www.ccsinfo.com

CY8C24223A Microcontroller and development tools
Cypress Semiconductor Corp.
www.cypress.com

µVGA Display
4DSystems
www.4dsystems.com.au

TTL-232R-3V3 Serial converter cable
Future Technology Devices International
www.ftdichip.com

Vinculum ASIC, VDIP1 and VMusic modules
Future Technology Devices International
www.vinculum.com

GPS 18 LVC Tracker
Garmin International, Inc.
www.garmin.com

DS1302 Timekeeping chip
Maxim Integrated Products, Inc.
www.maxim-ic.com

PIC16F88 Microcontroller
Microchip Technology, Inc.
www.microchip.com

Pyxos Power

If your project is in need of a standard network or has wiring concerns, nodes, or a complex protocol, Echelon's Pyxos FT chip may be the perfect fit. Tom explains how it will give you the reliability, security, and power of wires without the mess.

With all the action on the wireless front that I've been covering lately (e.g., Wireless USB, WiMedia, ZigBee, etc.), you'd think wires were an endangered species. Au contraire. There are still a lot of places where wires make sense from a reliability and security perspective, not to mention the small matter of distributing power.

If wires are still a good thing, that's not to ignore the fact that too much of a good thing can be a bad thing. Recently, my clothes dryer went on the fritz. After popping the hood, I discovered a rat's nest of wires running hither and yon between the control panel, motor, and various sensors. Just a reminder that old-school wiring lash-ups persist in many blue-collar embedded apps.

Is there a way to enjoy the beauty of wires with less of the beastly aspects? Echelon has got a new gadget, the Pyxos FT chip, that does just that (see Figure 1).

LAN-IN-THE-BOX

Echelon is best known for their LonWorks distributed processing and control scheme, which has found success over the years in large-scale building automation (e.g., lighting, security, and HVAC) and metering applications. However, while LonWorks may work for a skyscraper, it is overkill for smaller scale "in-room" or "in-box" applications. Enter Pyxos FT, designed to handle "last meter" tasks at the end of the wire. To that end, low-cost, low-power, and real-time capabilities are key requirements.

Here are the basics. A Pyxos FT network supports up to 32 nodes over tens to hundreds of meters of low-cost cable (e.g., Cat 5 twisted pair). The exact distance depends on a variety of factors, such as the type of wire, the network layout, and whether it just carries data or provides link power to attached devices as well.

Speaking of network layout, the "FT" in Pyxos FT stands for "free topology." Logically, the network can be considered a bus in the sense that all nodes see all traffic. But, just because it's logically a bus doesn't mean it has to be physically wired as such.

The range of wiring options can be understood as follows. Imagine you're able to connect the probes of a VOM to different nodes, in which case the wiring rule is simply as follows. Any configuration in which every node has electrical continuity with every other is allowed.

Obviously, a true physical bus (i.e., single-wire) fills that bill. But so does a star configuration. And so does a bus of stars, or a star of buses. Most notably, even a loop configuration (something most networking schemes frown on) is allowed. The cool thing about a loop is that it can be broken and the network will still work because a broken loop devolves to a bus.

Network management is centralized in a single Pyxos FT "Pilot" serving the individual nodes, which are known as "Points." The decision to use a centralized approach has a number of ramifications.

On the downside, centralized control means there is a single point of failure. If the Pilot dies, so does the network, and nodes will have to fall back to a limp-home mode that can deal with the loss of connectivity.

That's a significant price to pay, but one that delivers significant benefits as well. Perhaps most important, centralized control allows a degree of determinism that's difficult to obtain with distributed peer-to-peer schemes. The Pilot schedules network activity so that each Point receives a certain amount of bandwidth with a dependable schedule.

Of course, no networking scheme can "guarantee" data

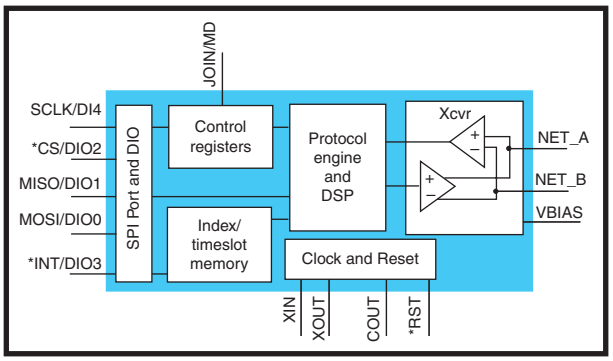


Figure 1—Consider the Pyxos FT chip kind of a network transceiver on steroids. In addition to differential signaling for range and noise immunity, it handles higher layers of the network protocol so you don't have to.

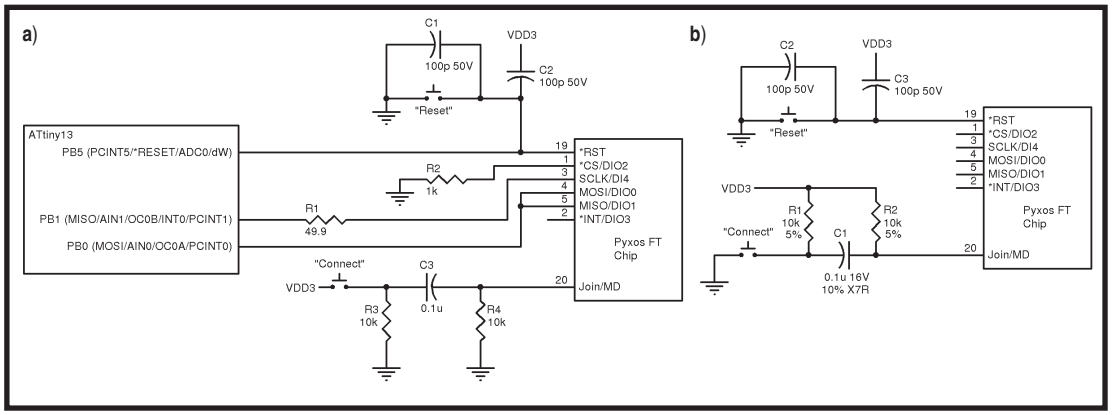


Figure 2a—In Hosted Point mode, the pins on the Pyxos FT chip implement an SPI for connection to the host MCU. **b—**In Unhosted Point mode, the pins revert to direct digital I/O functionality (four I/Os and one input).

delivery in all cases (e.g., wire short/open), so you the designer always have to make a provision to fail gracefully. But under normal conditions, Points can be assured they'll get the bandwidth and precise timing they need when they need it.

Another benefit of the Pilot & Point scheme is that the asymmetry puts most of the complexity in the Pilot, making life easier (and cheaper) for the more numerous Points. Indeed, a so-called "unhosted" Point need consist of little more than a Pyxos FT chip by itself (see Figure 2).

DOWN TO THE WIRE

The deterministic bandwidth and latency derives from the Pyxos FT "frame" (see Figure 3). In turn, a frame comprises timeslots, typically one for each Point. Each timeslot gives its associated Point the opportunity to send 8 bytes of data and receive 8 bytes of data.

This frame scheme has some obvious implications for system designers. First, the overall frame time directly depends on how many timeslots it contains, since the latter are of fixed duration. Thus, the network bandwidth and latency for an individual Point is directly and inversely related to the number of Points in the network. As you can see in Table 1, bandwidth ranges from about 5 to 70 kbps, fine for a sensor or actuator, but leave your multimedia A/V at home.

The fixed frame and timeslot scheme also dictates that when you design your application you have to decide how many Points it supports now or in the future. For instance, if you want to give the network the ability to "grow" (i.e., add Points), you'll have to allocate spare timeslots to accommodate future additions. Another option is to give the Pilot some extra intelligence (i.e., knowledge of multiple network configurations and the ability to switch between them) with the understanding that changing the frame time changes the Points bandwidth and latency.

Repeatable timing obviously works well in scenarios where data needs to flow like clockwork. But, what about applications where Point activity is more sporadic, perhaps triggered only infrequently by an alarm condition (e.g., temperature over threshold) or user intervention (e.g., a switch)? Fortunately, the Pyxos FT chip and protocol offers ways to accommodate, notably including interrupt and polling capability. The Pyxos FT chip interrupt output with a programmable trigger condition gives Points the ability to sleep until the Pilot actually sends them something. On the other side of the coin, polling allows a Point to query for any updates that might have occurred while it was asleep or otherwise preoccupied.

Besides the data, a Pyxos FT network can be designed to ship power over the same pair of wires. The link-power scheme is specified to use a 24-V AC or DC supply from which each Point derives its local supply (e.g., 3.3 V for the Pyxos FT chip along with any other voltages the Point requires).

How far you can stretch a link-powered Pyxos FT network depends on a lot of different factors. These include the type of wire, whether the link power is 24 VAC or VDC, and the amount of power required by the different nodes. A small network of low-power devices using the DC option might have a reach of 400 m. At the other extreme, a fully loaded network of power hogs using the AC option could top out at 10 m.

The Pyxos FT documentation includes reference designs for link-powered supplies giving Points 3.3-V power budgets of 100 mA (switching) or 15 mA (linear). Since getting the wires mixed up during installation is a matter of when, not if, the fact that

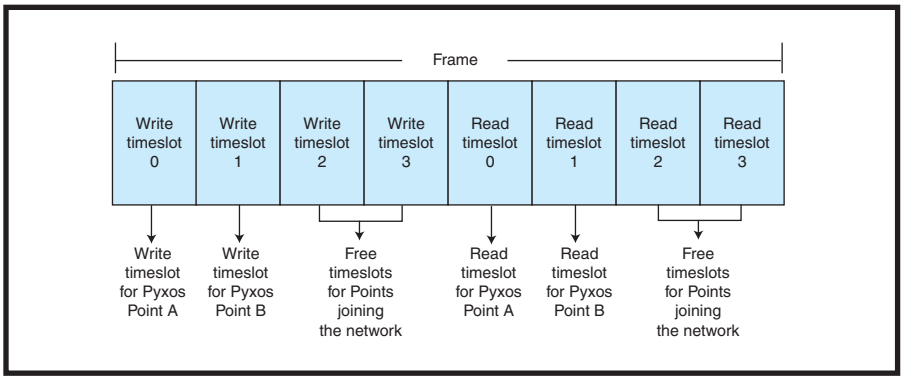


Figure 3—The Pyxos FT protocol relies on fixed-duration timeslots for each Point grouped into a fixed-duration frame. That means the bandwidth and latency from a Points perspective remains fixed (i.e., deterministic) even as the aggregate network loading (i.e., total of all Points activity) ebbs and flows.

Pyxos FT networks are designed to be polarity insensitive (both link power and data) is a very nice touch. Thanks to that feature and the "Free Topology" aspect, "wiring mistakes" are a real-world dollars-and-cents problem that Pyxos FT virtually eliminates.

VARIABLE INTEREST

Another thing I like is the way Pyxos FT networks (like their LonWorks brethren) utilize a simple "publish and subscribe" model based on "Standard Network Variable Types" (aka SNVTs). With this data-driven approach, the only thing a Pyxos FT Point knows about, or needs to know about, is the collection of network variables (eg., temperature, humidity, and light level) that are relevant to its specific purpose. A particular Point is defined by those network variables it has interest in (i.e., inputs) and/or those it has the duty to provide (i.e., outputs). The Point will receive a message from the Pilot that includes input variable updates. The Point will send a message to the Pilot when an output variable needs updating. Simple, just the way I like it.

Here's where the Point/Pilot asymmetry really pays off. For Points, life is easy since they need to do little more than wait for input updates to appear, and/or fire and forget output updates (see Table 2). That leaves plenty of cycles for a Point MCU to handle the local application or for that matter, just sleep a lot.

By contrast, it's the Pilot that has to keep track of all the network variables in the system, figure out where updates should go, and make sure they get there in a timely fashion. Furthermore, the Pilot is responsible for monitoring the health of the system and intervening if necessary (e.g., resetting or hot-swapping a crashed Point). Finally, the Pilot needs to be able to handle the aggregate peak network bandwidth while Points need only the horsepower necessary to deal with their own contribution.

Timeslots	Frame time[1]	Pilot average SPI bandwidth[2]	Point average SPI bandwidth[2]
2	1.8 ms	175 kbps	70 kbps
4	3.4 ms	190 kbps	38 kbps
6	4.9 ms	195 kbps	26 kbps
8	6.5 ms	198 kbps	20 kbps
10	8 ms	200 kbps	16 kbps
12	9.5 ms	201 kbps	13 kbps
14	11.1 ms	202 kbps	12 kbps
16	12.6 ms	203 kbps	10 kbps
18	14.2 ms	203 kbps	9 kbps
20	15.7 ms	204 kbps	8.1 kbps
22	17.3 ms	204 kbps	7.4 kbps
24	18.8 ms	204 kbps	6.8 kbps
26	20.3 ms	205 kbps	6.3 kbps
28	21.9 ms	205 kbps	5.9 kbps
30	23.4 ms	205 kbps	5.5 kbps
32	25 ms	205 kbps	5.1 kbps
[1] Approximate value			
[2] Sustained average rate. Peak rate may be significantly greater.			

Table 1—The bandwidth and latency (i.e., frame time) specs for the Pilot and Points depends on the number of devices attached to the network. One implication is that a hosted Point may well be able to get by with a bit-banged SPI, but a Pilot probably requires a host MCU that includes a high-speed (e.g., 1 Mbps) hardware SPI.

HOOK 'EM DANNO

When it comes to understanding the

Function	Description
`PsInit`	This is a user-provided (i.e., hardware-specific) function called at start-up to initialize the SPI the host MCU uses to communicate with the Pyxos FT chip.
`PyxosPointInit()`	Function called at startup to initialize the Pyxos FT chip and API.
`PyxosPointSendUniqueId()`	Each Pyxos FT chip is factory programmed with a unique 48-bit ID that identifies it within the network. In most cases registration is handled automatically by the Pyxos FT chips themselves, but this function allows a Point to manually register with the Pilot if necessary.
`PyxosPointAnnounceTimeslot()`	This function allows a Point to tell the Pilot which timeslot it should be assigned to. Such "hardwired" registration is suitable in networks with a known and unchanging configuration.
`PyxosPointGetTimeslot()`	After initiating registration (either manual, automatic, or hardwired), this function allows a Point to confirm which timeslot the Pilot has assigned to it.
`PyxosPointIsOnline()`	This function returns true or false to allow a Point to determine that it has successfully registered with the Pilot and the network is operational.
`PyxosPointEventHandler()`	This is the main function that your application must call periodically to handle Pxyos network events.
`PyxosPointPollPnv()`	Though a Pilot automatically updates network variables that are inputs to a Point, this function allows a Point to manually check for updates if necessary.
`PyxosPointUpdatePnv()`	This function initiates the transmission of a Point output network variable to the Pilot.
`PyxosPointIsPnvUpdatePending()`	This function returns true or false indicating whether previously initiated outputs from the Point have completed transmission to the Pilot.

Table 2—The Pyxos FT network strategy minimizes overhead and complexity for Points, as demonstrated by the simplicity of the Point application program interface (API) leaving the heavy lifting to the Pilot. Compared to the mere 10 functions (many only needed during initialization) that make up the Point API, the Pilot API has nearly two dozen functions.

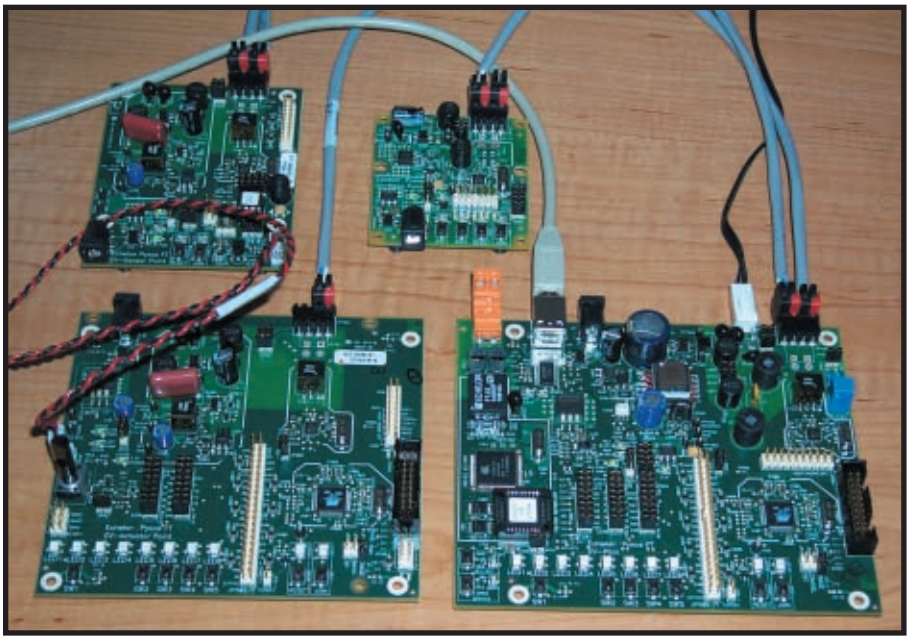


Photo 1—The Pyxos FT evaluation kit includes boards covering the likely application spectrum. Starting at the bottom right is the ARM7 MCU-based Pilot with USB (or optional LonWorks) connection to a host PC. Headed clockwise you see an ARM7 MCU-based actuator Point, an AVR MCU-based sensor Point, and an unhosted Nano Point.

range of Pyxos FT possibilities, a picture is worth a thousand words, so let's take a closer look. At $399, the Pyxos FT kit isn't an impulse buy, but you do get a significant amount of gear (see Photo 1). Here's the rundown.

Not unexpectedly, the Pilot, using an Atmel ARM7-based flash MCU, is the largest and most complicated board. Besides handling the Pilot duties, it also supports USB and/or LonWorks (the orange connector) connections to a host PC.

Next up is the "Accuator Point," which, using the same ARM7-based MCU as the Pilot, demonstrates an example of a "high-performance" Point. In addition to the complement of digital I/O that every board

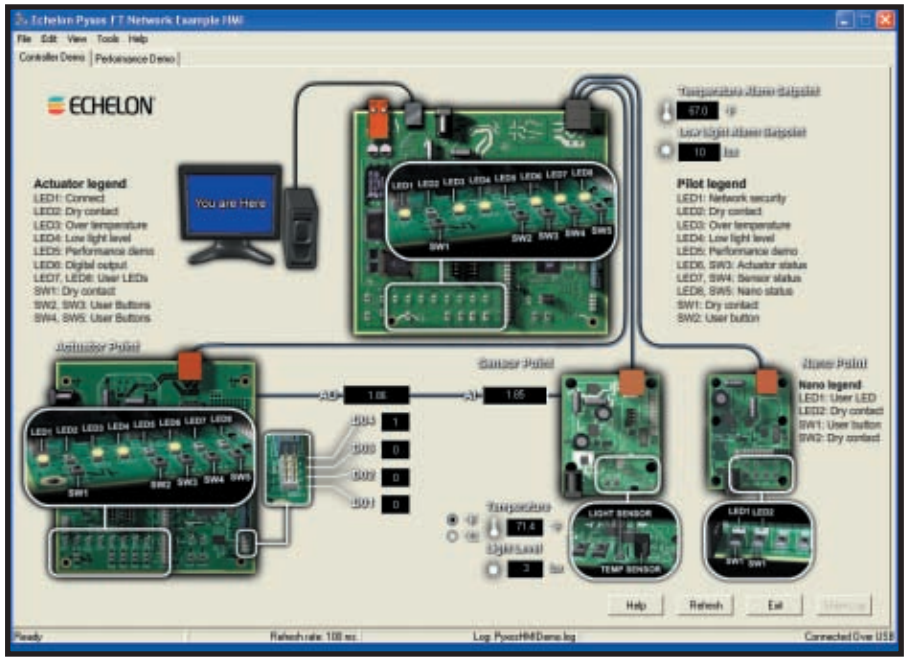


Photo 2—The human-machine interface (HMI) demo application centers on a visual representation, which along with an activity log, allows monitoring transactions between devices attached to the network.

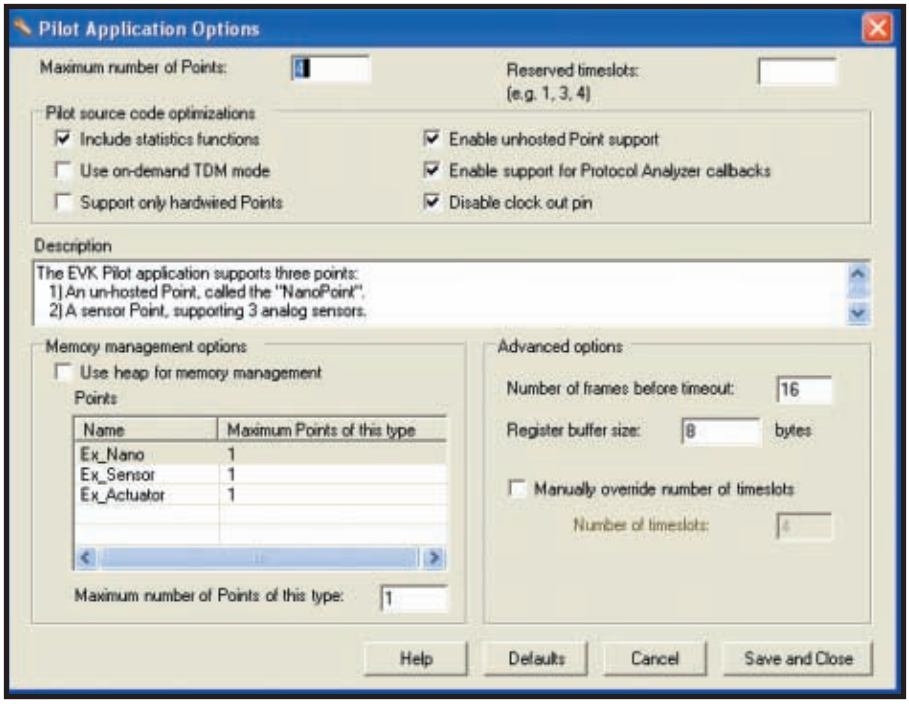


Photo 3—Another step in creating a Pyxos FT project is defining the basic characteristics of the network (e.g., the number of Points and timeslots), as well as embellishments, such as statistics gathering (e.g., for dynamic error recovery) and a protocol analyzer feature (i.e., network logging) for debugging.

includes (i.e., lights and switches), it includes an ADC for capturing an analog input voltage.

Moving down the ladder of cost and complexity, we find the "Sensor Point," which uses an Atmel TinyAVR 8-bit, eight-pin MCU. Befitting its name, in addition to the lights and switches, it includes temperature and light-level sensors as well as an analog voltage output.

Finally, we come to the "NanoPoint," which is an example of an "unhosted Point" (i.e., one that just uses the Pyxos FT chip by itself without a host MCU). In the latter configuration, the five pins used for the host MCU interface in "Hosted Point" mode are used instead as digital I/O to directly control attached devices, in this case the NanoPoints own complement of lights and switches.

A significant portion of the board space and circuitry you see is associated with the use of the 24-VAC link power option. Keep in mind that a minimal (i.e., nonisolated, common power supply) nonlink powered Point could consist of little more than the Pyxos FT chip and its required 10-MHz crystal. Also note that while the crystal is required, at least there's the

option to drive the 10-MHz clock waveform out of one of the Pyxos FT chip pins for use by other logic (e.g., a host MCU).

Software that comes with the kit includes a show and tell human-machine interface (HMI) application. The HMI demo centers around a screen representation of the entire network and the functions associated with each device (see Photo 2). For example, pushing a button on one of the Points causes both a "real" LED on the Pilot board, and the associated "virtual" LED on the HMI screen, to light at the same time.

Referring again to Photo 1, notice the small twisted pair (red and black) cable between the Sensor and Actuator Point boards. This cable connects the output DAC on the former to the input ADC on the latter. The HMI software includes a "Performance Demo," which is a loopback test that has the Pilot commanding the Sensor Point to output a voltage, which the Actuator Point in turn captures, converts, and publishes back to the Pilot.

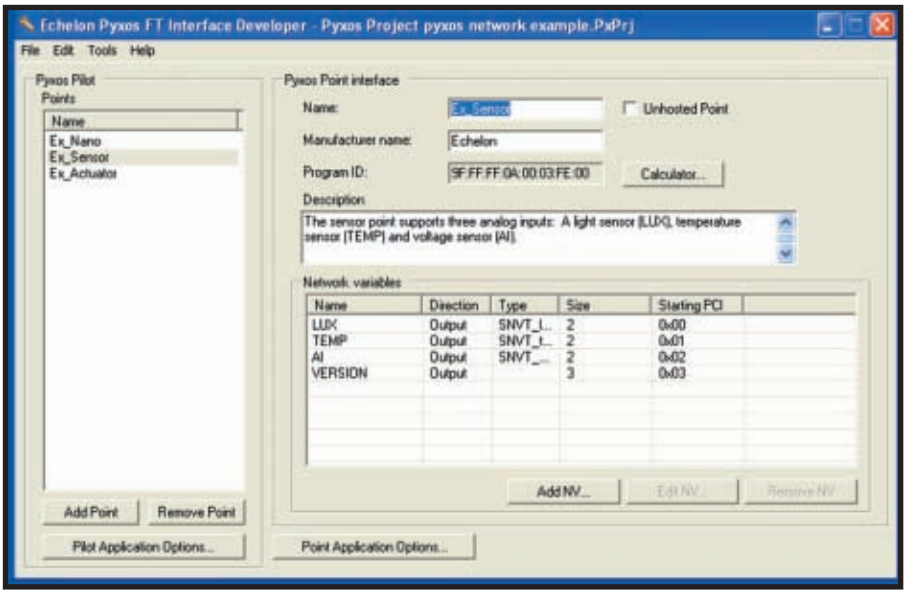


Photo 4—A Pyxos FT Project starts with the big picture representing each of the Points on the network. In turn, each Point is defined by the standard network variable types (SNVT) associated with its particular application.

The kit also includes the source code for the Pilot and Point APIs and, as mentioned earlier, the Pilot is quite a bit more complex. But, it's all relative considering the Point API requires less than 1 KB while the Pilot calls for a "whopping" 6 KB or so. Note that minimal hosted Point applications have the option of reducing host MCU overhead even further by dismissing the API altogether and sim-

HOT CHIPS 19

ADVANCE PROGRAM

HOT CHIPS 19

A Symposium on High-Performance Chips
August 19-21, 2007, Memorial Auditorium,
Stanford University, Palo Alto, California

HOT CHIPS brings together designers and architects of high-performance chips, software, and systems. Presentations focus on up-to-the-minute real developments. This symposium is the primary forum for engineers and researchers to highlight their leading-edge designs. Three full days of tutorials and technical sessions will keep you on top of the industry.

Sunday August 19

Morning Tutorial — ***Approaches to System Design for the Working Engineer***

Ralph Wittig, Xilinx; David Witt, Texas Instruments
Peter Alfke, Xilinx; Shephard Siegel, Mercury Computer Systems

Afternoon Tutorial — ***Enterprise Power and Cooling: A Chip-to-Data Center Perspective***

Norm Jouppi, HP Labs; Chandrakant Patel, HP Labs
Parthasarathy Ranganathan, HP Labs

Monday August 20

IBM Power6 — **IBM**

- Fault-Tolerant Design of the IBM POWER6™ Microprocessor
- System Performance Scaling of IBM POWER6™ Based Servers
- The 3rd Generation of IBM's Elastic Interface (EI-3) Implementation on POWER6™

Keynote I: **Vernor Vinge** — **Computer scientist, science-fiction writer, author of *True Names* and *Rainbows End***

Multi-Core and Parallelism A

- NVIDIA GeForce 8800™ GPU — **NVIDIA**
- NVIDIA GPU Parallel Computing Architecture — **NVIDIA**
- Performance of Non-Graphics Applications on the GeForce 8800™ — **UIUC**

Multi-Core and Parallelism B

- Radeon R600, a 2nd Generation Unified Shader Architecture — **AMD**
- Teraflop Prototype Processor with 80 Cores — **Intel**
- Design and Implementation of the TRIPS Prototype Chip — **UT Austin**
- Tile Processor: Embedded Multicore for Networking Multimedia — **Tilera**

Embedded and Video

- SH-X3:SuperH Multi-Core for Embedded Systems — **Renesas**
- An HD Image Processor for Low-Cost Entertainment Products — **TI**
- A Professional H.264/AVC CODEC Chip-Set for HDTV Broadcast — **NTT**

Panel: What's Next Beyond CMOS?
Moderator: Norm Jouppi — **HP Labs**

Tuesday August 21

Technology and Software Directions

- Multi-terabit Switch Fabrics Enabled by Proximity Communication — **Sun Micro**
- Thyristor RAM: A High-Speed High-Density Embedded Memory — **T-RAM Semi**
- Raksha: A Flexible Architecture for Software Security — **Stanford U**

Wireless

- A 4Gbps Wireless Uncompressed 1080p 60 GHz HD Transceiver — **SiBeam**
- A 2x2 MIMO Baseband for Wireless Local-Area Network (802.11n) — **Broadcom**

Keynote II: Multicore and Beyond: Evolving the X86 Architecture
Phil Hester, CTO, AMD

Networking

- A Packet Processing Chipset — **Cisco**
- Chesapeake: A 50Gbps Network Processor and Traffic Manager — **Bay Micro**
- A System-on-a-Chip with Integrated Accelerators — **Intel**
- Focalpoint II, A Low-Latency, High Bandwidth Switch/Router Chip — **Fulcrum Micro**

Mobile PC Processors and Chipsets

- Power Management Features in Penryn 45nm Core2™ Duo — **Intel**
- Next Generation Mobile X86 Processor — **AMD**
- nForce 680i and 680 Platform Processors — **NVIDIA**

Big Iron

- VictoriaFalls - Scaling Highly-Threaded Processor Cores — **Sun Micro**
- The Next-Generation Mainframe Microprocessor — **IBM**

IEEE computer society

This is a preliminary program; changes may occur. For the most up-to-the-minute details on presentations and schedules, and for registration information, please visit our web site where you can also check out HOT Interconnects (another HOT Symposium being held following HOT CHIPS):

Web: **http://www.hotchips.org** Email: **info2007@hotchips.org**

1997 2007 SSCS 10th Anniversary

IEEE

A Symposium of the Technical Committee on Microprocessors and Microcomputers of the IEEE Computer Society and the Solid State Circuits Society

ply driving the Pyxos FT chip directly. And of course there's no MCU overhead (because there's no MCU) in "nonhosted Points."

BUILD IT AND THEY WILL COME

The final piece of the puzzle is the "Pyxos FT Interface Developer" program. It's largely a point-and-click affair by which you define overall network options and specify the network variables associated with each Point. The tool uses the completed network specification to automatically generate header and include files that are combined with the source for the API and your application to create the final code.

Beyond the obvious, such as the number of points in the network, Photo 3 shows other options that can be selected. The default configuration assigns a timeslot for each Point in the network, but it's possible to override that by reserving additional timeslots. That allows Points to be added to the network without having to change the overall frame timing (i.e., Point bandwidth and latency remains fixed). Other options allow the Pilot to monitor the health of the network (e.g., error statistics) or take advantage of a protocol analyzer feature that enables logging all network activity. If you don't have a use for the Pyxos FT chip 10-MHz clock output pin, here's where you can disable it to reduce power consumption and noise.

Once the overall network and Pilot options are defined, it's time to move onto the Points themselves (see Photo 4). Point-specific characteristics include those mentioned above (e.g., statistics gathering, clock output disable) as well as defining a Points registration method at startup. Registration options include automatic (the Point and Pilot negotiate for a free timeslot), hardwired (the Point uses a specific reserved timeslot), and manual (an external event, such as pressing a switch, triggers registration). Note that manual registration is the only option for an unhosted Point.

The next step for configuring a Point depends on whether it is hosted or unhosted. For the latter, it's simply a matter of specifying the direction for each of the five I/O pins on the Pyxos FT chip. Note that four of them can be defined as either input or output, but the fifth pin is input only.

Hosted points are defined in terms of their network variables, which are the inputs a Point receives from the network and outputs it delivers to the network. As described earlier, network variables can be chosen from a list of existing standard types or, using another utility (resource editor), you can define your own.

Once everything (i.e., network, Pilot, Point) is specified, the Interface Builder program automatically generates the header and includes files that are combined with the API and each device's own application code. Run that through all the compilers involved (i.e., for the Pilot and hosted Point MCUs) to generate the different binaries and you're in business.

MATCH POINT

Before finishing up, I want to complement Echelon on the quality of the kit and tools. This is complicated stuff, but for the most part, everything worked as advertised right out of the box.

The only glitch I encountered was that the network activity-logging feature of the demo application didn't appear to work. It took a call to Echelon to find out that the feature is only enabled if your display resolution is greater than 1024 × 768 since otherwise there's just not enough room on the screen to show everything.

Between the various manuals for the chip, kit, and tools, the documentation is exceptionally complete and well-written. For example, the chip databook includes a very comprehensive 10-page itemized "check list" for your design. Follow it and you will find and correct 99% of the potential hardware problems before you power up the first time.

With the kudos out of the way, let's get to the bottom line. Is Pyxos FT right for your application?

Of course, for existing LonWorks applications the answer is clearly yes. Pyxos FT gives those designs the ability to cost-effectively extend the network's reach from the building down to the room and box level while using familiar techniques and tools.

On the other hand, there are clearly simple applications that you can get by without. For a physically small and self-contained device with relatively few nodes, key Pyxos FT features (up to 32 nodes over tens to hundreds of meters including the link-power option) go unutilized. At the same time, the costs (less than $2 for the Pyxos FT chip in volume, MCU codespace required, etc.), little they may be, loom relatively larger. For the simplest applications, hacking a one-shot hardwired solution, or even just sticking with the old wiring harness approach, remain viable options.

Somewhere in the middle is the sweet spot for Pyxos FT. Ask yourself these questions. Are there enough nodes and protocol complexity that a dedicated resource (i.e., the Pilot) is called for? Are there enough wiring concerns (e.g., range, noise, and immunity) that a transceiver chip of some sort is required in any case? Is the link-power capability a must-have that justifies the possible extra cost of dealing with a 24-V supply and the need to derive power locally at each node? Is your application a one-shot project, or will it grow and evolve to encompass a range of devices, which could benefit by using a single standard network? Does rolling your own networking solution represent a development chore that you'd just as well leave to Echelon?

There are many world-beyond-wiring-harness applications for which at least one of the answers is "yes." If yours is one of them, Pyxos FT is definitely an option worth considering.

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.

SOURCE

11500R Pyxos FT network chip and Pyxos FT evaluation kit
Echelon Corp.
www.echelon.com

by Jeff Bachiochi

Are You Up for 16 Bits?

A Look at Microchip's Family of 16-Bit Microcontrollers

Microchip Technology and *Circuit Cellar* have teamed up to bring you the Microchip 16-Bit Embedded Control Design Contest. Jeff introduces Microchip's 16-bit microcontroller and digital signal controller (DSC) families. Are you up for the challenge?

There will come a time when no matter how well you optimize your code, your favorite 8-bit microcontroller just won't get the work done. That's why Microchip Technology wants you to consider switching to one of its 16-bit MPUs.

If you've ever attended one of Microchip's local seminars, you realize (as Microchip does) that gaining hands-on experience is an effective way to learn something new. While the articles presented in *Circuit Cellar* offer this experience to readers who later reproduce the featured projects on their own, the magazine also recognizes the power of incentives. That's why *Circuit Cellar* runs design contests. There are numerous incentives for entering a contest. The most obvious incentives are cash, prizes, and notoriety. But let's be honest: the best part is the excitement of creating a project.

I'm sure you've read an article or two in *Circuit Cellar* that was written by a past contest winner. Now it's your turn to shine. *Circuit Cellar* and Microchip Technology have teamed up for the magazine's first contest of 2007, the Microchip 16-bit Embedded Control Design Contest. Are you up for the challenge?

In this article, I'll introduce you to Microchip's 16-bit family of parts. I am always a bit apprehensive about leaving my comfort zone and trying a new part. If you haven't used a 16-bit part, I hope this article piques your interest. As you'll see, you can start working with a 16-bit part with a minimal amount of effort. I encourage you to take this opportunity to take the leap and enter the contest. After all, it's really just a small step.

MICROCHIP'S 16-BIT FAMILY

If you have experience working with Microchip's 8-bit microcontrollers, you won't be surprised that Microchip's policy has been smooth migration from the lowest (six-pin) 8-bit MPU to the highest (100-pin) 16-bit DSC. The code compatibility, packaging, and tools remain remarkably consistent when migrating between families.

One of the best ways to present the advantages of the 16-bit family is to compare it to its 8-bit counterpart. Table 1 shows the total 8-bit family (which is extremely flavorful) compared to the two groups in the 16-bit embedded controller family. You may be surprised by the cost comparison in the last row of the table. There is no large premium for moving up to the higher-performance family.

EENY, MEENY, MINY, MOE

When it comes to picking a microcontroller, there are a few factors that can quickly narrow the search. The PCB technology you are designing for will be through-hole, SMT, or a combination of the two. You will be restricted to using an SMT part for many applications that require a microcontroller with high I/O counts.

	8-bit (PIC10–18)	16-bit (PIC24)	16-bit (dsPIC30–33)
I/O	6 to 80 pins	18 to 100 pins	18 to 100 pins
Program memory	384 bytes to 128 KB	12 to 256 KB	12 to 256 KB
Data memory	16 bytes to 3,968 bytes	1 to 16 KB	1 to 16 KB
EEPROM	0 to 1 KB	—	0 to 4 KB
Speed	Up to 10 MIPS	Up to 40 MIPS	Up to 40 MIPS
Communication peripherals	UART, SPI, I²C, CAN, USB, LIN	UART(1 to 2), SPI(1 to 2), I²C (1 to 2), CAN(0 to 2)	UART(1 to 2), SPI(1 to 2), I²C (1 to 2), CAN(0 to 2)
Display peripherals	LED, LCD	—	—
Control/timing peripherals	Capture/compare, PWMs, counters/timers, watchdog timers	Capture/compare (4 to 8), PWM (2 to 8), counter/timer (3 to 9), watchdog timer	Capture/compare (4 to 8), PWM (2 to 8), counter/timer (3 to 9), watchdog timer
Analog peripherals	ADC, comparator, op-amp, brownout detector, low-voltage detector, temperature sensor, oscillator, voltage reference, DAC	ADC (1 to 2), comparator (2 to 3), brownout detector, low-voltage detector, oscillator, voltage references	ADC (1 to 2), brownout detector, low-voltage detector, oscillator, voltage references
Additional peripherals	—	CRC generator, DMA, RTCC, JTAG	CRC generator, DMA, RTCC, JTAG, codec (0–2), DSP
Cost (quantity)	Less than $1 to $7	$3 to $7	$3 to $9

Table 1—Some peripherals appear the same across the 8-bit/16-bit boundary, but many of the 16-bit peripherals are enhanced and can be found in multiples.

The I/O requirements should be well defined before you choose a microcontroller. This will allow you to zero in on a package size. Through-hole parts come in 18-, 20-, 28-, and 40-pin DIP styles. DIP SMT parts come in 18-, 20-, and 28-pin SOIC and the smaller 20- and 28-pin SSOP style. Tiny leadless QFM styles include 28- and 44-pin packages. The higher-pin-count parts (e.g., the 60-, 80-, and 100-pin TQFPs) are available in both a 0.4-mm and a 0.5-mm lead pitch.

If you looked at all of the 28-pin parts (both 8-bit and 16-bit) offered by Microchip, it would still be overwhelming. That's why it is helpful to look at what's available using a couple of different methods. The selection guides on the Microchip web site for 8- and 16-bit microcontrollers are based on a few different factors, including family memory size, memory type, and application features. I prefer using the pin count or application features to narrow down the field.

Once all the peripheral bases have been covered, you may need to estimate memory size or speed requirements. Memory size can be affected by your choice of language, use of tables, and the application's computational requirements. Speed may be an important factor if you need fast sampling or data throughput. One advantage of having standard-size packages available throughout a product line is the ability to substitute either higher- or lower-performance parts. I often start with the highest performance part for my prototype and adjust my choice downward to fit the final code size and speed necessary to achieve my design requirements.

PIC24F

The PIC24F is the least expensive part in the 16-bit family of microcontrollers. It runs on a 2- to 3.6-V supply with 16-MIPS operation at 3.3 V. Each peripheral has its own interrupt vector with multiple priority levels. Sixteen working registers include designated registers for single-cycle 16-bit (signed or unsigned) multiply and 32- and 16-bit (signed or unsigned) division (18 cycles). Single and multibit shifts are also performed in a single cycle. A repeat instruction simplifies multiple iterations of any instruction.

The enhanced peripherals in the 16-bit family include 32-bit timers, high-speed 10-bit A/D conversion (up to 500 ksps) with a sample buffer, JTAG, multiple I^2C/UART/SPI with FIFO buffers and IrDA support on the UART, and parallel master port support for external peripherals. A real-time clock/calendar (RTCC) with an external 32-kHz crystal supports 99 years (leap year aware) and alarms based on a mask of any and all registers. The CRC generator module uses software-configurable terms and lengths for making cyclic redundancy calculations.

PIC24H

For a higher performance, H-series parts can provide 40 MIPS at 3.3 V. In addition to the PIC24F's enhanced peripherals, the PIC24H includes CAN, a high-speed 10- to 12-bit ADC capable of multiple simultaneous samples, and a 2-KB dual-port RAM buffer for the DMA transfer of most peripheral data. When applications include collaboration from multiple sources, the PIC24H series can provide code-guard protection so that each can have independent security over its code.

NEED DSP?

In reality, most signals start out (and may very well end up) as an analog signal. However, to make use of digital signal processing, they are converted into a digital representation of the signal by sampling with an ADC (or they are returned to analog via a DAC). The sequence of samples is in the time or spatial domain. Algorithms can be used on the sequence to measure or filter it. The DSP is specifically designed for computational efficiency and throughput of data. A typical DSP application might include audio processing, image processing, or compression, as well as data processing, analysis, and control of industrial processes.

The dsPIC series integrates a DSP engine into the 16-bit PIC24 series of microcontrollers. Today, the dsPIC series consists of two families, the dsPIC30 and the dsPIC33 families.

dsPIC30/33

The dsPIC30 series is for those of you in need of 5 V of operation (2.5 to 5.5 V). This family comes in four varieties: general-purpose, sensor, motor control, and power conversion, all operating at 30 MIPS. Each flavor has peripherals based on general or specific applications. For instance, the sensor family is based on low-pin-count devices in support of high-performance yet cost-sensitive applications. The motor control family includes a quadrature encoder for position and velocity feedback. Analog comparators and high-speed ADCs and PWMs in the power conversion family are needed for SMPS, UPS, and power-factor-correction applications. The general-purpose family features codec support for speech and audio applications.

Executing at 40 MIPS (3.3 V), the dsPIC33 family has the highest performance. The general-purpose family (GP) has both low-pin-count devices for sensor applications and larger devices with codec interfaces and DMA. The motor control and power conversion family (MC) has quadrature encoders and DMA.

STANDARD FEATURES

Over the years, additional features have crept into Microchip's embedded controllers. Many now contain an on-chip precision oscillator that can eliminate the need for external crystals, resonators, or clock sources. Low-jitter PLLs can boost clocks to full internal speed (lower-speed external sources means reduced EMI). Watchdogs have separate oscillators for independent operation. An on-chip clock monitor can restart the system using the internal oscillator in case of an external-clock failure. Power consumption can be optimized by shutting down peripherals or running at a reduced speed during idle periods. Internal reset and brown-out circuitry adds reliability without external components. Many I/O pins have a high current-drive capability (drives LEDs directly). In addition to individual interrupt vectors for each peripheral, 16-bit microcontrollers also have individual error trap vectors for oscillator failure, illegal access, stack over and underflow, and math errors.

Where would a good microcontroller be without good tools? MPLAB is the "free" integrated development environment (IDE) that has been helping users edit, build, and debug programs for many years (and continues to improve). This contains an assembler/linker/librarian, a visual device

initializer, and a simulator. It supports Microchip's hardware development tools: in-circuit debugger (ICD2), in-circuit emulator (REAL ICE), and production programmer (PM3). MPLAB also makes it easy to hook in your favorite third-party software. And remember, MPLAB is the only tool you'll need for Microchip's complete line of microcontrollers.

PERIPHERAL PIN SELECT

One of the problems with low-pin-count microcontrollers is that they are limited by their available I/O pins. As a result, the family has more peripherals than the part has pins to support. Peripheral pin select (PPS) does away with this limitation by allowing certain peripherals to be dynamically mapped to a group of I/O pins. Although a single pin can be shared among peripherals, you should be careful. This opens the door for some pretty tricky configurations. Because this is a dynamic function, some safeguards are implemented to lock the configuration registers and report if a malfunction (illegal change in configuration) occurs.

Change-of-state inputs have been a standard peripheral on microcontrollers for some time now. They are practical for implementing a wakeup on key-press functions. COS on the 16-bit microcontrollers is now implemented with dynamic mapping. Each pin mapped for COS can enable an internal pullup (reducing the need for external components).

DSP

Support is a key issue when using the DSP. To eliminate the extensive development time, a set of speed-optimized functions for the most common digital signal processing (DSP) applications is included in the DSP library (see Table 2). Most functions are written in assembly language and make extensive use of the DSP instruction set and hardware resources. The library provides vector, matrix, filtering, transform, and window operations. FIR filter functions include lattice, decimating, interpolating, and LMS filters. IIR filter functions include canonic, transposed canonic, and lattice filters. Transform functions include in-place, out-of-place, DCT, FFT, and IFFT transforms. Window functions include bartlett, blackman, hamming, hanning, and kaiser windows. Control functions include PID control.

Not only will you find additional free libraries for the complete 16-bit family covering math functions (see Table 3) and peripheral use (see Table 4), but there are also application-specific libraries. They include speech encoding and decoding, noise suppression, acoustic and line echo cancellation, speech recognition, TCP and IP, V.32/22/22bis, encryption, FAT16, advanced AC induction (ACIM), brushless DC (BLDC), and permanent magnet synchronous motor (PMSM) control. Please visit the "16-bit development boards, tools, and libraries" page on Microchip's web site for more information about the Microchip and third-party libraries and tools.

Not every library is free. While there may be licensing fees involved with using some of these in production, evaluation (single use) copies are available to designers for a $5 charge.

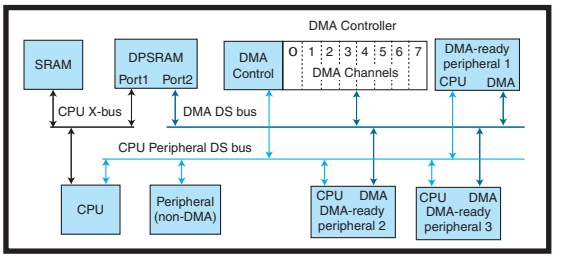


Figure 1—Separate buses allow DMA transfers to occur simultaneously with CPU program execution. Many peripherals support this background operation.

DMA

As you can see in Figure 1, the 16-bit parts with DMA have a separate (and dedicated) bus for DMA activity. The CPU communicates with conventional SRAM across the data space X-bus and to DPSRAM port 1. It talks to the peripherals across a separate peripheral data space bus (shown in light blue). The DMA controller communicates with port 2 of the DP SRAM and the DMA port of each of the DMA-ready peripherals across a dedicated DMA transfer bus (shown in dark blue). Microchip's 16-bit architecture allows the CPU and DMA to complete the transfer of a byte or word every bus cycle across its dedicated bus.

Since either the CPU or the DMA controller can respond to a peripheral-interrupt request, you must choose any peripheral interrupt to be either a CPU or a DMA request. A DMA request will be arbitrated with any other coincident requests. If the channel

Function execution times				
Function	**Cycle count equation**	**Conditions**	**Number of cycles**	**Execution time at 40 MIPS**
Complex FFT**	—	N = 64	3,739	93.4 µs
Complex FFT**	—	N = 128	8,485	212.1 µs
Complex FFT**	—	N = 256	19,055	476.4 µs
Block FIR	53 + N (4 + M)	N = 32, M = 32	1,205	30.1 µs
Block FIR lattice	41 + N (4 + 7 M)	N = 32, M = 32	7,337	183.4 µs
Block IIR canonic	36 + N (8 + 7 S)	N = 32, S = 4	1,188	29.7 µs
Block IIR lattice	46 + N (16 + 7 M)	N = 32, M = 8	2,350	58.7 µs
Matrix add	20 + 3 (C × R)	C = 8, R = 8	212	5.3 µs
Matrix transpose	16 + C (6 + 3 (R – 1))	C = 8, R = 8	232	5.8 µs
Vector dot product	17 + 3 N	N = 32	113	2.9 µs
Vector max	19 + 7 (N – 2)	N = 32	229	5.7 µs
Vector multiply	17 + 4 N	N = 32	145	3.6 µs
Vector power	16 + 2 N	N = 32	80	2 µs
Proportional integral derivative (PID)	30	N = 1	30	0.75 µs
*C = number of columns, N = number of samples, M = number of taps, S = number of sections, R = number of rows				
**Complex FFT routine inherently prevents overflow				

Table 2—This table will give you a feel for the execution times you can expect when using the DSP algorithm library. The PID function is the latest to be included in this free library.

wins, the transfer will be completed during the next cycle. The DMA controller retrieves the source and destination addresses from the active channel's registers (DXAxSTA, DMAxSTB, and DMAxPAD). Block transfers are also supported. When the transfer counter reaches a user-defined limit, a CPU interrupt can be initiated to signal the CPU to process the newly received data.

The CPU and DMA controller may simultaneously read or read/write to any dual-port SRAM or DMA-ready peripheral data register. The only conflict occurs when the CPU and DMA controller simultaneously write to the same address. The CPU write will win and a "DMA fault" trap exception will be initiated. If the DMA controller writes to a location during the same bus cycle when the CPU is reading it (or vice-versa), the location is read prior to the write during the cycle in question.

JUMP-START PCBs

You can find a hardware development board for practically every part that Microchip manufactures. Although they are not free, they are inexpensive and most come with a plethora of information and demo code to get you rolling your own quickly. I have a few favorites that you will find helpful for this contest (see Photo 1).

As a participant, your contest kit includes a batch of parts, both 16-bit microcontrollers, and a slew of peripheral components. The included CD was specially prepared for the contest and includes MPLAB and a special issue of the C30 compiler.

Note that the compiler will run as a full compiler during the contest and will then revert to the function-limited student edition. Additional documentation on the CD includes datasheets and reference manuals for all the parts plus application notes, programming specifications, and other goodies.

If you choose to use special offers and purchase a demo board, you will be pleased with the level of hand holding provided to get you started. Every demo board comes with a tutorial that takes you from project creation through debugging, so you are familiar with the process. All of the necessary tutorial source code is provided so you can hit the ground running.

Function Group	Function
Basic floating point	Addition
	Subtraction
	Multiplication
	Division
	Remainder
Trigonometric and hyperbolic	acos
	asin
	atan
	atan2
	cos
	sin
	tan
	cosh
	sinh
	tanh
Logarithmic and exponential	exp
	frexp
	ldexp
	log
	log10
Power functions	pow
	sqrt
Rounding functions	ceil
	floor
Absolute value functions	fabs
Modular arithmetic functions	modf
	fmod

Table 3—The functions in the math library are IEEE-754-compliant (operated in the "Round to Nearest" mode), with signed zero, signed infinity, not a number (NaN), and denormal support.

SMPS FOR $750

Although Microchip includes four different 16-bit microcontrollers in its contest kit (all are 28-pin DIP parts), the effective use of any of these could bring you home some serious cash. There are three main prizes: first place ($5,000), second place ($3,000), and third place ($1,750). Judging of the top three entries is based on technical merit, effective use of on-chip resources, usefulness, effective use of bonus parts, originality, and cost-effectiveness. Cover these and your entry will be sure to collect points from the judges.

In addition to the top three honors, prizes will be awarded for the top picks in seven categories. Most do not have a particular16-bit part associated with them and any 16-bit device can be used in your design for these categories. You will walk away with $750 if your design is chosen by the judges as "best in category."

Photo 1—These are the development boards I find useful for investigating Microchip's 16-bit microcontroller families. Microchip is offering some nice discounts for contest entrants on select items (i.e., the DM300027 is approximately $40).

Note that each entry will automatically be entered into a category that fits; however, your entry will be eligible to win a maximum of two categories. The first category is for the effective use of SMPS resources (dsPIC30F2020). This microcontroller with DSP has the peripherals you'll need, including an ultra-fast PWM and analog/digital for your switch mode power supply (SMPS) design. You'll be looking at methods for sensing current, bias supplies, gate drive issues, transient response, and how topology choices affect system design.

The second category is for the effective use of motor control resources (dsPIC30F2010). Although this microcontroller with DSP has a quadrature encoder and six PWMs, you might want to design for something other than motor control. Try this device for power conversion, lighting, or other controls using the fast 10-bit ADC with multiple samples and holds.

The third category is for the use of smart sensing methodologies (any 16-bit microcontroller). Design a sensor application that uses the intelligence of one of the devices to improve on traditional sensor methods. Digital signal conditioning may be the answer to simplifying a cost-sensitive application.

The fourth category is for connected (communications) applications (any 16-bit microcontroller). All of the 16-bit devices include various serial-communication channels. Their inherent nature allows them to easily access off-chip peripherals in larger systems. This might be a good category for your LIN or CAN bus application.

The fifth category is for the effective

use of a peripheral pin select (PIC24FJ64GA002). The new peripheral pin select module allows you to determine which of the on-board digital peripherals is brought to the external I/O pins. You can configure, or reconfigure, their connections via software registers. This dynamic selection allows for some very clever configurations.

The sixth category is for the effective use of DSP technology (dsPIC30/33). The functionality of a DSP in a microcontroller greatly expands your design possibilities. You might show how this added power can be used to eliminate external circuitry and actually reduce overall costs.

The last category is for the best use of bonus parts (any 16-bit microcontroller). The design kit includes a number of bonus parts that can be used to compliment the microcontroller in your design. Your design for this category must include the external analog memory or connectivity components.

Timers
Input capture
Output compare
Quadrature encoder interface (QEI)
Motor control PWM
I/O ports and external interrupts
Reset
UART
SPI
I^2C
Data converter interface (DCI)
CAN (standard and enhanced)
10-bit ADC
12-bit ADC
Real-time clock and calendar
Cyclic redundancy check
Direct memory access (DMA)
Functions for controlling an external LCD via I/O port pins

Table 4—This is a table of functions used to set up and control the operation of all the peripheral modules available in the PIC24*xx* and dsPIC*xx* devices. The peripheral library serves as a convenient layer of abstraction over the specific details of the peripherals and their associated control and status registers.

THINK, ENTER, WIN

The required elements for any design are critical when balancing system costs for a particular solution. Not only do the number of I/O pins, memory size, CPU/clock speed, and voltage and temperature operating ranges contribute to a design's cost, they also contribute to your ability to eliminate external parts by making good use of the microcontroller's features.

Microchip provides a line of microcontrollers that builds upon a proven core. The migration path is cohesive with a wide breadth of products that all use the same user-friendly tools. This simple premise provides you with a short design-to-market time frame.

Although this contest could ultimately reward you for your design skills, I believe you have an opportunity for something more important. Microchip is putting a bunch of parts (and all the tools necessary to use them) into your hands. DIP parts will allow you to easily breadboard a circuit. The software tools will allow you to write and simulate code. You will need some hardware to program your code into the microcontroller. I strongly suggest that you consider investing in the ICD2. This in-circuit debugger/programmer will become your best friend. Microchip is including a voucher for 50% off a one-time purchase of select tools. Note that you can get a development PCB and an ICD2 for $99!

No purchase is necessary to submit an entry to the contest. Although Microchip and *Circuit Cellar* are looking forward to your design entry, you will be a winner even if you only use this contest as an opportunity to learn about Microchip's 16-bit microcontrollers (or one of their external peripheral parts). After experiencing the process of developing for one of Microchip's 16-bit microcontrollers, you'll feel comfortable using any microcontroller in its complete line. After all, your comfort level has a lot to do with choosing a device for a future design project.

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.

SOURCES

PIC24FJ64GA002 Microcontroller, dsPIC30F2010 DSP, and dsPIC30F2020 DSP
Microchip Technology, Inc.
www.microchip.com

by George Martin

From "Hello World" to Big Iron

It's time for the "big iron." George shows you how to start writing code and building a project with an NNDK-MOD5234 development kit. He got his project up and running using a serial port interface.

In the mid-1980s, I designed a gate array for a character-recognition project with a 2,000-gate device built on 2-µm technology. I entered the design using text constructions (not graphically or even HDL) via a modem to the engineering VAX at General Electric at Research Triangle Park, North Carolina. The VAX churned for about 8 h, depending on the workload, and then gave me an error report. I soon discovered that if I waited until Duke, North Carolina, or N.C. State were playing basketball, I could have the VAX, an IBM mainframe, and a CRAY all working on my design with a turn around time of 15 min. or less. Now that's what I call BIG IRON!

Gate arrays have come a long way since then. In a previous article, I wrote about putting the NIOS processor into one of Altera's gate arrays (G. Martin, "Designing with the NIOS," Parts 1 and 2, *Circuit Cellar* 167 and 168, June and July 2004). The design tools for that all run on a PC in a reasonable amount of time. The particular array family I'm using is in the 12,000 to 15,000 LE (logic elements, which contain many gate equivalents) range with 256 Kb of RAM. (RAM was a separate IC in the mid-1980s.) My PC is an Intel 3.2-GHz Pentium 4 HT. I bet I'm at least within an order of magnitude in computing power to my 1985 environment, even on game day. The performance of a modern Cray midrange CPU is reported at 2 to 10 GigaFlops. Oh, by the way, the Sony PlayStation 3 is calculated at over 2 TeraFlops. One TeraFlop is 1 trillion (10^{12}) floating-point operations per second! Perhaps that's BIG IRON?

We've already looked at the Texas Instruments MSP430 family in the earlier parts of this project. They cost less than $1, even in small quantities. Now let's turn our attention to a processor away from the low end, but not so far away in price. I'm referring to Freescale Semiconductor's Coldfire family of processors.

NNDK-MOD5234-KIT

The source I'm using for these devices is NetBurner, which has core modules that consist of a CPU and memory. The modules plug into a baseboard that can have a wide variety of different interfaces and development kits, which contain a CPU, a baseboard, and supporting software. I need a unit that can perform MP3 decompression, interface to a memory card (SD type, I think), interface to the Internet, and run the µC/OS real-time operating system. So, I settled on Freescale Semiconductor's NNDK-MOD5234-KIT. There are several to choose from. You can also start with the Freescale CPU, if you've got the resources to build your own boards. It's available in a ball grid array (BGA) package that must be soldered onto a PCB. Freescale has other evaluation boards available and they even list the NetBurner products.

The MOD5234 has a Coldfire CPU with 147 MHz, 2 MB of flash memory, 8 MB of SDRAM, and an Ethernet interface. This is almost 150 million operations per second (MIPS). The NNDK kit contains a baseboard that has RS-232/485 interfaces, a real-time clock, a power supply, a prototype area, and displays. I'm feeling my way through this description since I just received that unit last week. Check out the web site for all the detailed information that comes with the kit. I've left off listing all the software that comes with the kit. As we go through this next series of articles, all the available Netburner software should be covered.

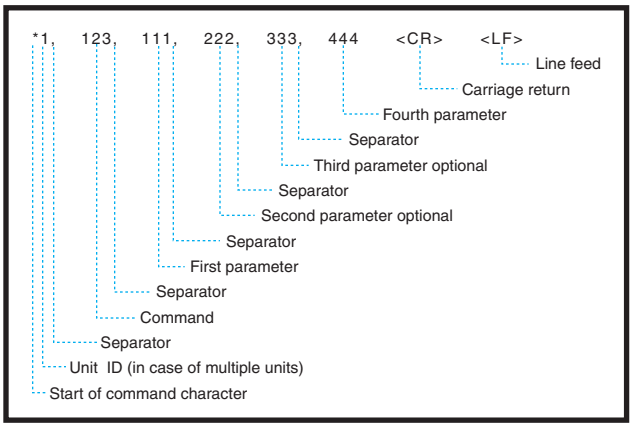


Figure 1—A typical command format using only numerical data.

GETTING STARTED

What's the plan? I would like to get this board up and running using a serial port interface. I have not made up the command set, but my

Serial interrupt routine requirements:
- Receive all characters ignoring <LF> characters
- Buffer received characters
- Keep track of received lines

Command processing routine requirements:
- Check to see if new command is received
- Extract the complete command line
- Parse command line into its following components
  1. Valid start character
  2. Unit ID
  3. First parameter
  4. Second parameter
  3. Third parameter
  6. …
  7. Last parameter
- Execute the command

Figure 2—These are serial input requirements.

commands typically look like what you see in Figure 1.

The command format gives you a lot of flexibility. Keep in mind that you don't yet know the specific commands you intend to implement. The asterisk character (*) marks the beginning of a command and must be a unique character not used as part of any command. Also, the carriage return (<CR>) character marks the end of a command and must be a unique character. I also permit line feed (<LF>) characters. This lets the user create a list of commands in a text file and transfer that file using HyperTerminal (or a similar program) to your embedded system. <LF> characters make the file printable. I just ignore them as they are received.

First, let's consider only numerical parameters for the commands. This will exclude alphanumeric values, such as file names, but we can revisit this decision at a later date. This command definition creates a set of requirements for the serial interrupt and command processing routines (see Figure 2).

Does this set of requirements look like a C routine to you? Don't worry if you don't know how to code that design. With a little practice, it will be second nature. Could you have written these requirement using UML? Sure you could have. The requirements are really simple and UML would do a fine job of capturing them. I cannot urge you enough to start your designs in this manner. It's good practice, and as they get more complicated, you'll find it much easier to

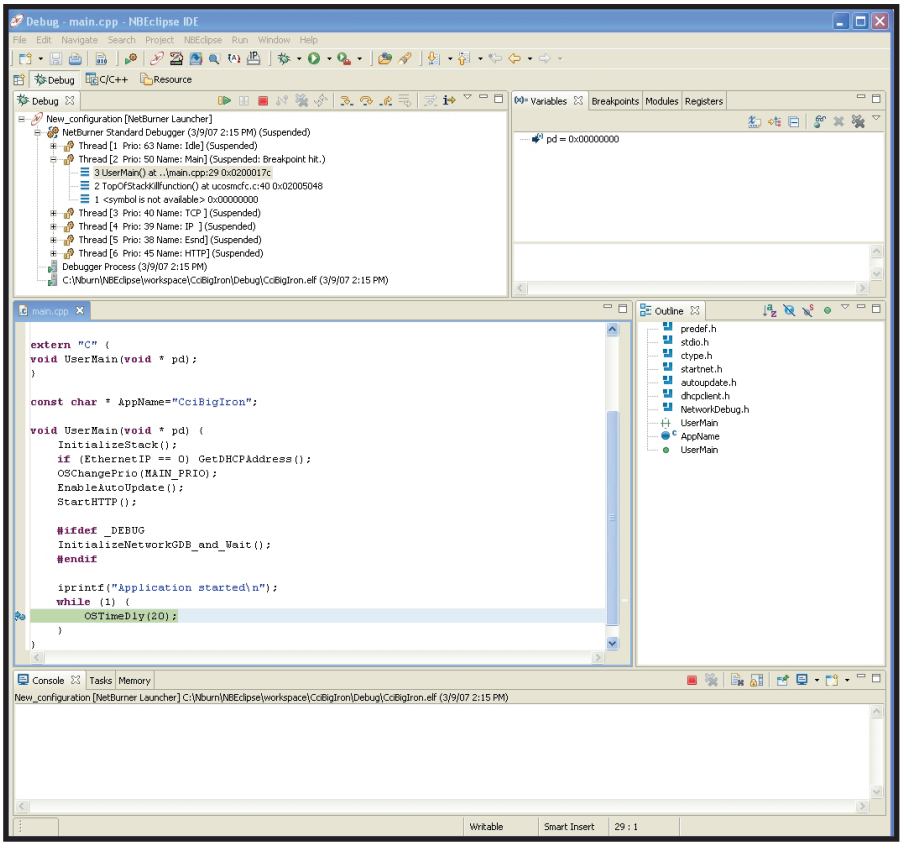


Photo 1—This is a screenshot from the Eclipse debugger. I'm running the default application created by the NetBurner tool set. The CPU is stopped in Thread 2, main.cpp at the `OSTimeDly` instruction.

Listing 1—Here are some straightforward methods to declare and reference character arrays. I have found these useful in small embedded controller applications.

```
INT8 buf[100]; // is a buffer to hold 100 8 bit entities (characters).
buf[0];        // is the first arraymember.
buf[99];       // is the last array member.
INT8 *p;       // is a pointer to an INT8 datatype.
p = &buf[0];   // sets the pointer p pointing to the first member of buf.
p = buf;      //  sets the pointer p pointing to the first member of buf.
buf[0] = 'A'; //  sets the first member of the buf array to the ASCII character A.
buf = "This is a message";      // is an illegal statement!!
strcpy(buf, "This is a message"); // copies the string into the array.
```

work in the requirements area than in the code area. Coding should be the final phase of the process.

JUMP IN

Enough lecturing, let's jump in and do something. I installed the tools from the NetBurner CD and that went on without a hitch. I created desktop icons for three programs that I'll be using frequently: IP Setup Tool, Dev C++, and NetBurner Embedded Eclipse IDE.

The IP Setup Tool does just that. When you plug the NetBurner development board into your network, this tool finds the board and lets you launch a web page contained in the board. The web page is the MOD5234 factory demo program that shows different examples of supported features.

The Dev C++ is an integrated development environment (IDE) distributed under the GNU general public license. The environment is used for code development (editing and compiling). Don't worry about the C++ moniker. I'll only be using C. As you go along, you'll see how that's accomplished.

The NBEclipse is the IDE for debugging. Eclipse was created by IBM as the successor to its VisualAge family of tools. It is now managed by the Eclipse Foundation, an independent not-for-profit consortium of software industry vendors.

I started with a default type of project that just waits for a timer to expire, but I had trouble getting it to work. I had to download an updated version of the tools, which is available at NetBurner's web site under the support tab. To be honest, I was still having problems getting the network debugging up and running. A phone call to support sorted out my difficulties. If you have been following this series of articles, you read about my wrestling with the default settings and file locations for the vendor's IDE. This issue was in that category. I wanted to place all my files in one directory and the tools were looking for them in a different directory (the default directory). Not a big deal, and one that NetBurner is working to correct. I expect to report to you again about this issue as you explore other vendors' offerings.

A screenshot of that initial debug session is in Photo 1. A classic menu and toolbar buttons are across the two top rows of controls. Below that, the Debug perspective is highlighted. "Debug" enables you to look at the project. All the tasks that are running (several) are in the top-left window. In the debug window (top left), I expanded the Main thread. Note that several threads are running. And then, in the code window (lower left), I inserted a break point in my main routine. When an application is stopped at a breakpoint, each task entry can be expanded to display the call stack and line number in the source code from which it was called. The NetBurner development board is running the µC/OS-II real-time operating system (RTOS). You can find articles about that RTOS in past issues of *Circuit Cellar*, where it has been described in detail. It's a popular RTOS and a good one to start with.

The other area of Photo 1 contains information about variables, registers, breakpoints, and more. I will cover these in more detail as the design progresses. I added lines 17 and 32 to insert a variable `i` and to increment that variable each time through my big do-nothing loop. Pressing the green GO arrow runs the CPU until the next breakpoint is encountered, and in this case, the variable `i` has been incremented.

Now I added some serial port code. The NetBurner system comes with a serial cable and the Mtty tool that is used to load code or reload factory default programs when you've got things all goofy. Since this cable is connected to my development PC, I plan to also use this as your serial interface connection. There is a Reset button on the NetBurner board, and I can control the system with the Mtty program. This looks like a setup that will lead to a stable development environment, even if I have written unstable code.

My next decision was whether to use the support provided by µC/OS-II for the serial ports. As I explored example code, I talked to support technicians at NetBurner and compiled code. How to partition the design became clearer.

The files consist of the following: CciBiSerial.cpp (the serial interface routines), CciBiSerial.h (prototypes for the serial interface), CciBiReports.cpp (the reporting routines), and main.cpp (the main (initial) file). Cci stands for Circuit Cellar, Inc. (the customer). Bi stands for Big Iron (my name for this project). The file extensions are cpp, used with C++. The tools defaulted to this with the main file, and I didn't want to change anything because I was just getting started and I was not sure about the exact requirements for the IDE. Don't worry, we are only using the C language.

Look in file main.cpp at about line 24 and you will see:

```
extern "C" { void UserMain(void * pd); }
```

This is telling the C++ compiler that this is C code and to build interfaces accordingly. The file was generated automatically by the IDE's new project command. I added the header files for serial support, the prototypes files for my serial routines, and the calls to my serial routines.

In the UserMain routine, I added lines 55, 57, and 60 to 63. They all initialize the serial code and process serial data as it arrives.

One word of caution: you can't place any breakpoints before the system is up and running. That implies the code must get past line 50 before breakpoints are valid. I added line 55 OSTimeDly(20); just to be sure the OS was happy and my breakpoints would have some significance.

As you look in file CciBiSerial.cpp, you'll see the header files, a structure definition for COM_DETAILS, and the variable int fdDebug. The structure COM_DETAILS is a place to hold information about the serial interface. The NetBurner board has two serial ports, and the processor has support for other types of serial ports in both the hardware and the software. So it's not uncommon to have three or more serial ports in a system today. It would be very tedious to write dedicated code for each port. I plan to create a structure for each port in use and fill the structure with the appropriate values. Then, as data comes in each port, I will process that data using the structure for the port.

If you look in CciBiSerial.h, you will see the definition for COM_DETAILS. That structure holds the latest character received, a command buffer, and a parameter buffer. The parameter buffer is the command data separated into the individual components. You will also see pointers to routines. As each communications port's requirements change, I can insert different routines in their structure. So, each port will call the routine contained in the structure. Routines don't have to be created for each port.

In CciBiSerial.h, I commented out my typedefs for INT8, INT16, and so on (lines 27 to 40). NetBurner also defined these, and I was getting a warning for the duplication, so there was no need to use my typedefs. (Great minds and all that.)

I commented them out using the #ifdef NEVER and #endif pair. C uses a preprocessor to go over the input files before compilation. This is the #define statement that we have used to define constants. A more complete explanation is that the #define is used to substitute one expression for another. It's a good practice to use parentheses to avoid errors.

```
#define  <macro> <replacement name>
#define  LEFT_SHIFT_8(x) ((x)<<8)
#define  FALSE 0
#define  TRUE !FALSE
```

If you have not defined the macro NEVER, then the code between #ifdef NEVER and #endif will not be compiled. A good use of the #ifdef statement is to distinguish between different hardware or compilers. You can write one piece of code and run it on a PC with Turbo C 3.0, Visual C++, or in your embedded system. You can then use the #ifdef statements to change where I/O is located for example.

STRINGS IN C

I need to interrupt this presentation to talk a bit about strings in C. An INT8 is a signed 8-bit entity, but it's really built on the char data type. It's very common to save ASCII characters in 8-bit memory locations. The C language also supports strings.

C strings are character sequences that are stored as one-dimensional character arrays. They are terminated with a null character ('\0'), which is called NULL in ASCII. The name refers to the C programming language using this string representation. Refer to Listing 1 and remember what I said about arrays.

If you look up strcpy, you will see the following description:

```
char *strcpy(char *dst, const char *src);
```

I like to think of the keyword const as a promise you are making with the compiler. In this case, you are agreeing with the compiler that you will not be changing the character string src while the strcpy operation is running. And in the example above, the string "This is a message" is constant for the copy operation. Const has more value than this simple explanation and that's another discussion.

The NNDK user manual contains documentation on the libraries included. The tools use "the Cygnus C Support Library libc 1.4 Full Configuration May 1993" as the foundation, but they also include libraries for hardware found on the prototype board, including the OS. This is where one would go to find out how the int read(int fd, char *buf, int nbytes); procedure would work or what the define FD_ISSET represents. If you study this carefully, you'll gain a great understanding of the support provided with your compiler and system. You'll also see how the experts have implemented various functions.

In main() in main.cpp, after the system is up and running, I call InitDebugComBuf(). This sets up all the variables needed for serial communications. Then I enter a while (1) loop. This is an endless loop that runs forever. In that loop I call DebugProcessCmd(). That routine waits for an entire line of serial characters to come in. The asterisk (*) character signifies the start of a command. The carriage return character signifies the end of a line. The call in DebugProcessCmd() is at line 224 and it reads done = cp->GetLine();. Let's break this down. "cp" is a pointer to a structure of COM_DETAILS. And in the definition of COM_DETAILS (CciBiSerial.h line 63), you see a place to hold the get character routine (INT8 (*GetLine) (void);). This is a pointer to a function that takes no parameters and returns an INT8. In the InitDebugComBuf() routine, you set Debug.GetLine = DebugGetLine;. So, you start to see the power of using structures to encapsulate your information. You could change INT16 DebugProcessCmd(void); to ProcessCmd(struct COM_DETAILS *cp); and not set cp = &Debug; inside the routine. You would then be able to pass different structures to the same code and handle many serial ports without writing unique software for each.

FINAL STEPS

After DebugProcessCmd() has received a complete command line, it calls ExtractParameters();. That routine looks at the command line it just received. If a valid start character is in the command line, the routine

separates all the parameters. It copies each parameter into the `INT8 ParBuf[DBG_COM_PAR_MAX][DBG_COM_CHAR_MAX]` array. The first subscript indexes each parameter while the second subscript indexes each character in that parameter.

After extracting parameters, `DebugProcessCmd()` checks the unit ID portion of the command line that's been separated into the `ParBuf` array. If the unit ID matches the unit's ID, the command is for you. I put in this test in case you are going to have several units connected of a party line like the RS-485 interface. If the command is for this unit, the command value is used to index a table of command numbers and a routine to call for each value. I just step through each entry until I find a match.

Look at CciBiSerial.cpp line 178. That's where I define the structure of the commands and matching routines. Look at lines 180 and 182. The first is the one that's coded for this example. Line 182 is commented out, but it defines a routine that is a pointer to the `COM_DETAILS` structure. Using that definition, you could have data come in on one serial port and output it to another again without customizing the code just by modifying the tables.

Well, we've got a simple command processor up and running on a powerful board. This should give you a peek at how to get started on such a project. Next time, I would like to expand on the processing of strings. We will write a file conversion program that reads pick and place data outputted by the CAD program and writes a file that the pick and place machine can use.

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

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/204.

RESOURCE

NetBurner Inc., "Embedded Eclipse IDE," 2007, www.netburner.com.

SOURCES

NNDK-MOD5234 Network development kit
NetBurner, Inc.
www.netburner.com

MSP430 Microcontroller
Texas Instruments, Inc.
www.ti.com

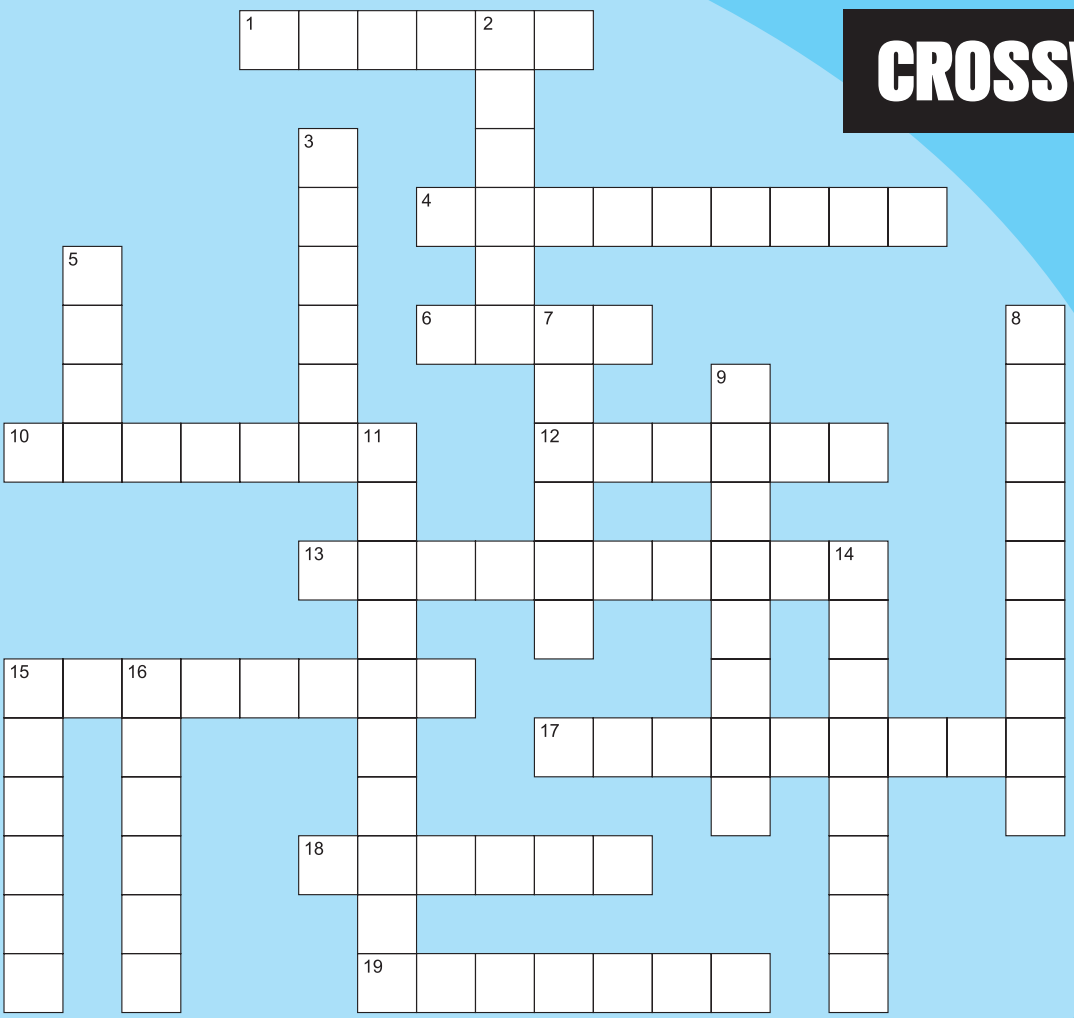
IDEA BOX

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"x 3" FORMAT**. Call for current rate and deadline information. Send your disk and digital submission sheet to: IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066 or e-mail adcopy@circuitcellar.com. For more information call Shannon Barraclough at (860) 872-3064.

The Suppliers Directory at www.circuitcellar.com/suppliers_dir/
is your guide to a variety of engineering products and services.

CROSSWORD



Across

1. A small keyboard often containing just numeric keys and a few symbols
4. Released in 1982, the breadbox-shaped PC used a 1.02-MHz processor
6. A measure of the ability of a circuit to increase the power or voltage
10. Someone who posts chronological, electronic journal entries of their thoughts
12. A composite made by combining two different elements
13. An electronic device that decreases the amplitude or power of a signal without distorting its waveform
15. A social networking site that was originally developed for college students
17. A process, used in calculations or logical operations, that is applied to data
18. Software that converts data being sent to it or received from it into the required format for a peripheral
19. One-way communication where one party is the transmitter and the other party is the receiver

Down

2. A bright glow in the sky caused by the interaction between the Earth's magnetic field and charged particles from the sun
3. A poor design that is often hastily put together
5. The American physicist (1855–1938) who discovered the potential difference on opposite sides of an electrical conductor where an electric current is flowing. An effect, an element, a voltage, and a resistance are named after him.
7. A recently released multimedia device that incorporates an Internet-enabled phone and an MP3 player
8. A large computer that has vast computational power for critical applications
9. An agreed set of rules on the transmission of data between two devices
11. A computer virus that actively seeks out an antivirus program and attacks it
14. A temporary storage area in a central processor that usually has a one-word capacity
15. A cable carrying electricity for power
16. A flashing symbol that indicates where the next typed character will be

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

INDEX OF ADVERTISERS

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

Preview of August Issue 205
Theme: Embedded Development

Embedded Scripting

High-Performance Motor Controller

Text Adventure Gaming

Handheld Multifunction Scope

Electronic Bicycle Design

Build a PHP Components Database

1-Wire in the Real World (Part 1) : The Challenges

THE DARKER SIDE Let's Play with EMI!

ABOVE THE GROUND PLANE Hearing Clearly

FROM THE BENCH Graphical User Interfacing

SILICON UPDATE Silicon Secrets

KEIL™
An ARM® Company

Microcontroller Development Tools

Only 4 Steps...

...are required to generate efficient, reliable applications with the µVision IDE and development tools from Keil.

Step 1. *Select Microcontroller and Specify Target Hardware*

Use the Keil Device Database (**www.keil.com/dd**) to find the optimum microcontroller for your application.

In µVision, select the microcontroller to pre-configure tools and obtain CPU startup code.

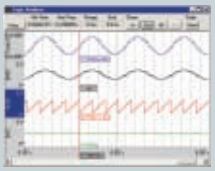
Step 2. *Configure the Device and Create Application Code*

The µVision Configuration Wizard helps you tailor startup code to match your target hardware and application requirements.

Extensive program examples and project templates help you jump-start your designs.

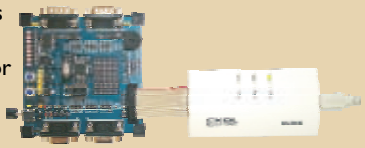
Step 3. *Verify Program Execution with Device Simulation*

High-speed simulation enables testing before hardware is available and helps you with features like instruction trace, code coverage, and logic analysis.

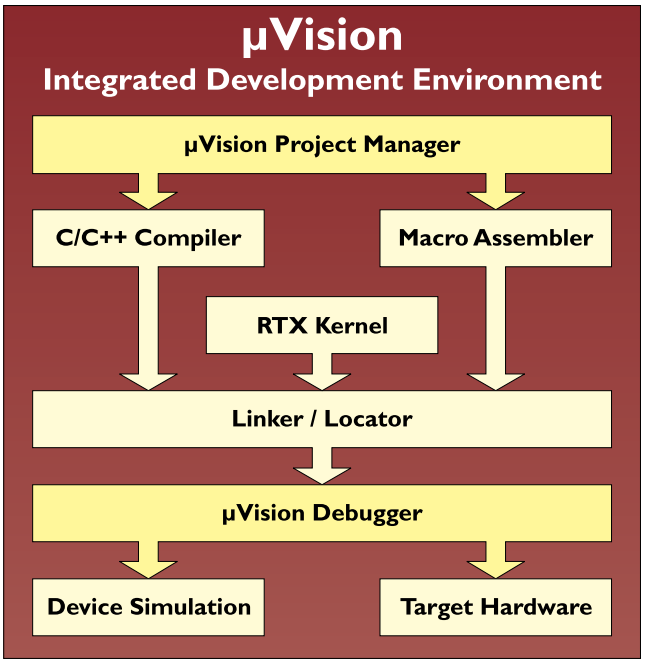


Step 4. *Download to Flash and Test Application*

Once your application is runs in simulation, use the Keil ULINK USB-JTAG Adapter for Flash programming and final application testing.



Keil Microcontroller Development Tools help you create embedded applications quickly and accurately. Keil tools are easy to learn and use, yet powerful enough for the most demanding microcontroller projects.



Components of Keil Microcontroller Development Kits

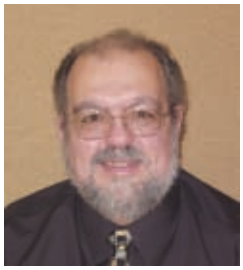
Keil makes C compilers, macro assemblers, real-time kernels, debuggers, simulators, evaluation boards, and emulators.

Over 1,200 MCU devices are supported for:

- **8-bit** - 8051 and extended 8051 variants
- **16-bit** - C166, XC166, and ST10
- **32-bit** - ARM7, ARM9, and Cortex-M3

Download an evaluation version from
www.keil.com/demo

800-348-8051 **www.keil.com**



PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

Keeping the Lights On: Reality Time

OK, I admit it. I'm a tech junkie, and it's been a few years since I was focused on a large project. The truth is that every once in a while I need the excitement of ripping the place apart and building something entirely new.

It's only been a few days since I decided to do this, but if it can be installed at my location (without clear-cutting half the county), I'm going to install a photovoltaic power-generating system. From what I can tell so far, installation costs border on the insane, but the fact that it satisfies my major objective for an independent, secure source of power, it may trump all the mitigating factors. Besides, at $0.20 per kWh in CT right now, I love the idea of selling electricity back to the power company.

I hatched that idea a few months ago when I was 1,000 miles away on vacation. Now that I'm back in CT and have had an opportunity to put my money where my mouth is, I thought I'd give you a status report.

First of all, I need to set the record straight. I'm not a tree-hugger, and I don't believe that either extreme in the environmental debate has the facts straight. I suspect the answers lie considerably more toward the middle, where few dare to reside these days. What I do put my faith in is good engineering. I've been watching PV technology for many years while it seemed the exclusive province of the off-grid crowd and social isolationists. Now that PV and other renewable energy systems are becoming more mainstream and with fewer esoteric anomalies, photovoltaic power is starting to make real engineering sense for me (let's hold the dollars and sense [sic] question 'til later).

Without checking the details of the specific program in CT, my first thought was to simply enhance my existing self-sufficiency by slapping a few PV panels on the roof and generating my own power. The reality of life is that while the architecture of an off-grid battery-backed PV system could accomplish this, the Connecticut Clean Energy Fund doesn't provide rebates toward such a configuration, and it would be a lot more expensive going it alone. I guess my 14-kW diesel generator gets to stay as my main electrical backup for a few more years.

The aim of the CT rebate program is to take some of the load off the present electrical grid instead of simply adding another power plant. The objective is to replace some of the demand with personal renewable energy-generating systems, such as PV. The subsidy here is only for a professionally installed grid-tied PV system where you and the utility share the electricity you produce. Contrary to what I said before (call it media brainwashing), your electric meter doesn't necessarily rotate forward or backward. That really only pertains to older analog electric meters, which, at least in CT, are always digital for grid-tied PV. The two digital readings represent what you use and what you produce. Simple subtraction decides who gets the money.

While I suppose I should be upset that I'm not going to be crawling across a hot roof in July, not personally digging 7′ deep mounting holes, and not lugging 33-lb solar panels up a ladder, there will be lots to document and interfaces to be designed as the installation progresses. The CT rebate program dictates the PV architecture and the approved list of installers. I did the math and I can't even buy the raw materials to do this project for less than what a completely installed system ends up costing with the current CT rebate program. The only choice was to decide how big a system I could afford (no, the magazine isn't paying for this), how much of the county to clear-cut, and who would be the lucky installer to get to deal with me while this mess is being built.

Incidentally, I've always been one of the first to adopt new technology, but this may be setting a new standard even for me. According to the CT Energy Fund, the total number of present residential PV installations in the whole state of Connecticut, population 3.5 million, is 152 (plus 18 commercial installations)! There are another 90 or so residential projects on the books for this year, but any way you look at it, we are definitely at the leading edge of the curve.

Presuming that everything stays on schedule, in a few months, I'll present a couple of articles on how my system was built and add our house to the installed PV list. In case you are interested, my objectives haven't changed in the last few months—only the complexity has changed. The contract has been signed and the rebate documentation has been submitted. If nothing clogs up the works or screws up the rebate, in a few weeks, there should be a bunch of people doing the ladder and roof trick at my house. The contracted system consists of 52 solar panels with a total rating of 10,760 W feeding three inverters. As expected with any project I start these days, it is also far more complex than originally anticipated. It turns out that to get this much capacity, my system will have to have panels on the roof along with two pole-mounted arrays. There better not be any glitches either. A crew of five has been out cutting and removing trees for the last four days. I can hardly hear myself think over all the chainsaw noise. That, and all this blinding sun I've never seen before!

I am putting my money where my mouth is because I believe we are at a point where PV and other renewable-energy systems are starting to make sense. Whether it be a solar heating pump controller, a smart load for a wind turbine, or a net-tied renewable energy monitoring system, etc., etc., I invite other authors to join me in creating regular coverage of this technology in *Circuit Cellar*. It's one thing to simply add it to the interest list in our author's guide. It's quite another when you see us actively pursuing the technology and doing it ourselves. I realize that we are on the cutting edge of some very expensive stuff here and it will take time to evolve as a focus, but just like *Circuit Cellar* provides the premiere source of intensive, exploratory articles about the hardware and software methods for embedded control, it is my intention that we do it just as well for renewable-energy electronics. I invite you to join me.

steve.ciarcia@circuitcellar.com

PC Based Logic Analyzers

NEW Model DV3400

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

from $499.00

DigiView

Professional Hardware Capture + Software Analysis

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

Very flexible, "easy to configure" Advanced triggering.

PICmicro® MCU Programmer

Multi-Function, In-Circuit & Gang Operation

only $199.00

QuickWriter™

See our Memory Emulators from $179.00

TechTools

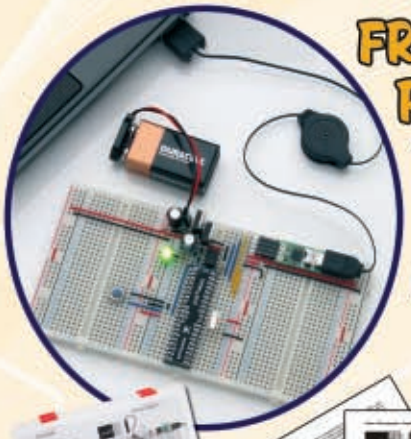
www.tech-tools.com

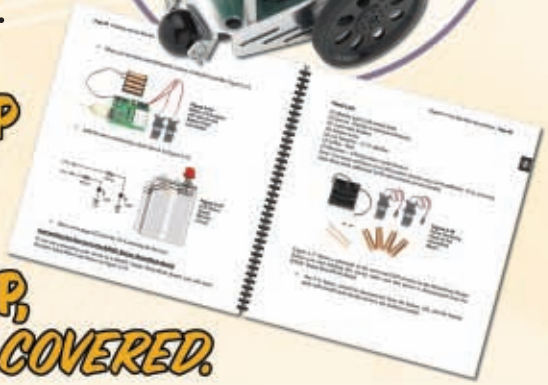
(972) 272-9392 • sales@tech-tools.com

MICROCONTROLLER EDUCATION
WE WROTE THE BOOK!

SINCE 1997

10 years ago Parallax created the Stamps in Class program to address the need for affordable electronics and programming educational resources. We now offer 10 comprehensive tutorials designed to introduce students and educators to electronics, programming, robotics, and sensor management with the easy-to-use BASIC Stamp microcontroller. Our interchangeable hardware platform and our free software, tutorials and sample code downloads make outfitting your classroom affordable. Our BASIC Stamp Educators Courses and broad support base help you get started and keep your program growing. **To learn more about Stamps in Class, visit www.parallax.com/sic.**





FROM BASIC STAMP PROGRAMMING TO THE NEW PROPELLER CHIP, WE'VE GOT YOU COVERED.

In the spirit of Stamps in Class, we have created our Propeller Education Kit and Labs to provide an affordable platform for learning cutting-edge multi-processing microcontroller technology. With 8 processors that run simultaneously to perform independent or cooperative tasks, the Propeller chip is an ideal controller for integrated embedded system design projects. As always, our manual, labs, software and sample code are free downloads. Propeller Education Kit (#32305; $79.95) **Visit www.parallax.com/propeller for details.**

To learn more about Parallax BASIC Stamp education visit www.parallax.com/sic. Quantity discounts are available. Parallax Sales Department: 888-512-1024 (Mon-Fri, 7am-5pm, PDT).

PARALLAX inc

www.parallax.com