

CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

#226 May 2009

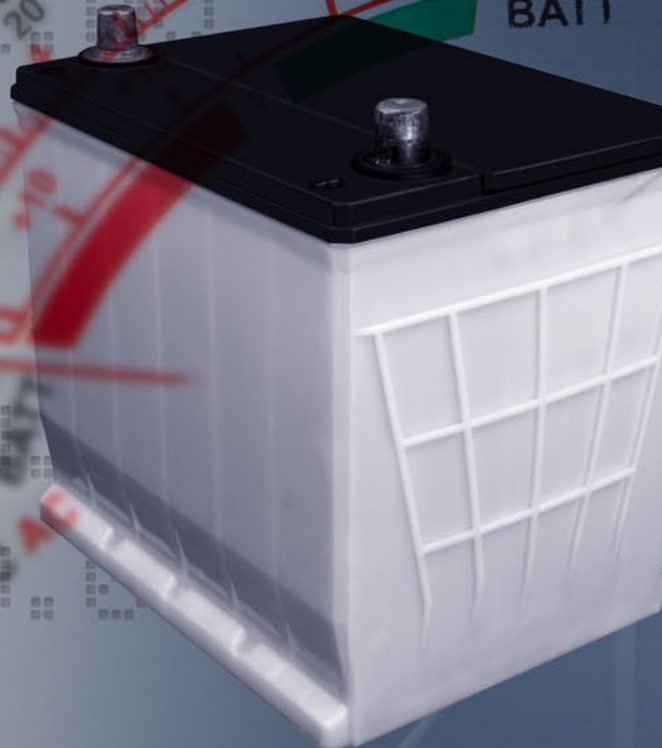
MEASUREMENT & SENSORS

MCU-Based SLA Battery
Measurement

Modern DSP Technology

Transformerless Power
Supply

C Code for a FAT File
System



SECURE SERIAL-TO-ETHERNET SOLUTION



Low-cost



32-bit Performance



SSH/SSL Secured

Need a custom solution?

Customize with the NetBurner SB70LC Development Kit for only \$99.

Customize any aspect of operation including web pages, data filtering, or custom network applications.

Kit includes: platform hardware, ANSI C/C++ compiler, TCP/IP stack, web server, e-mail protocols, RTOS, flash file system, Eclipse IDE, debugger, cables and power supply

Kit enables communication with peripherals that use: SD/MMC Flash Card (including SDHC), SPI, I²C, or the general purpose digital I/O interface

The NetBurner Security Suite Option includes: SSH v1, v2 and SSL support

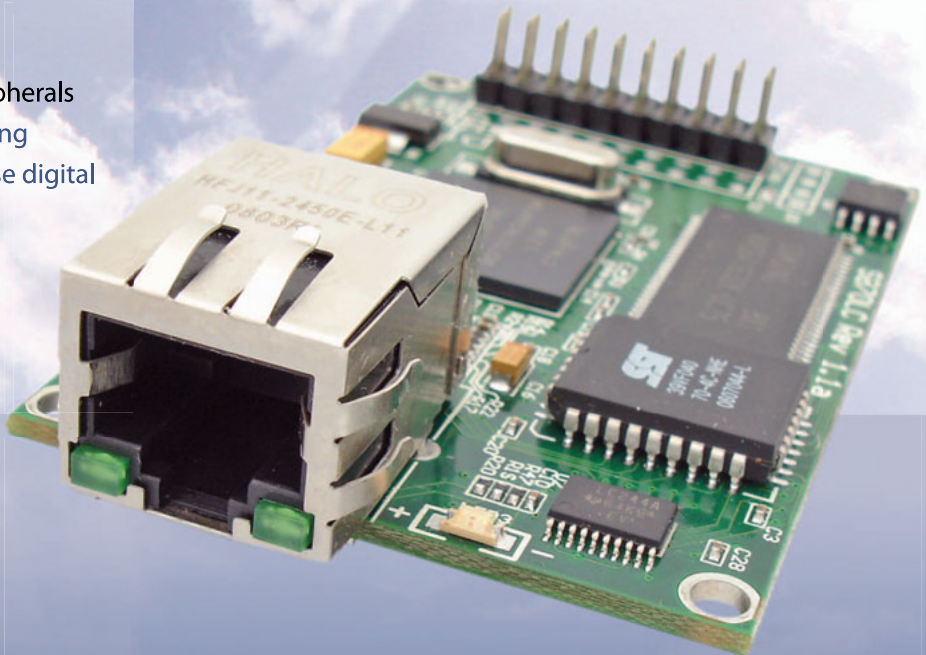
The complete, **secure** hardware and software solution

- Simple Ethernet connectivity for serial devices
- Works out of the box - no programming is required
- Enable data encryption to prevent unauthorized monitoring
- Customize to suit any application with development kit

Features:

- 10/100 Mbps Ethernet
- SSH/SSL/TCP/UDP modes
- DHCP/ Static IP support
- Web-based configuration
- Two TTL serial ports

SB70_{LC}
\$49⁰⁰ Qty. 100



Board Part Number | SB70LC-100CR
Development Kit Part Number | NNDK-SB70LC-KIT
Information and Sales | sales@netburner.com
Web | www.netburner.com
Telephone | 1-800-695-6828



5 Competitive Advantages

Overseas Manufacturing

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

CAM USA

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

Quick-Turn Production

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

Shipping Logistics

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

Significant Price Saving

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

Quick-Turn
Production

CAM USA

Door to Door
Delivery

Significant
Price Saving

STIMULUS
NO INTEREST
ORDER NOW
PAY 50% IN 6 MONTHS
NO PENALTY
PACKAGE

For details please
visit our website
or call us

Capabilities

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

ITAR, ISO 9001 : 2008

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



Imagineering Inc.

847-806-0003

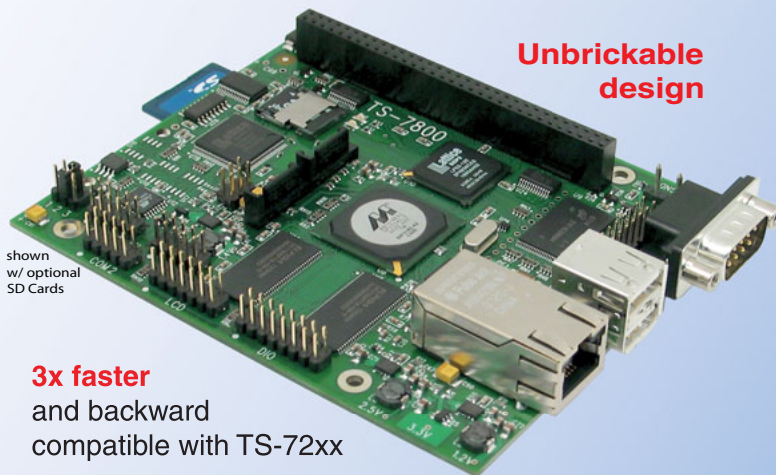
www.PCBnet.com

email: sales@PCBnet.com

24 YEARS IN BUSINESS...AND STILL GOING STRONG

Embedded Single Board Computers

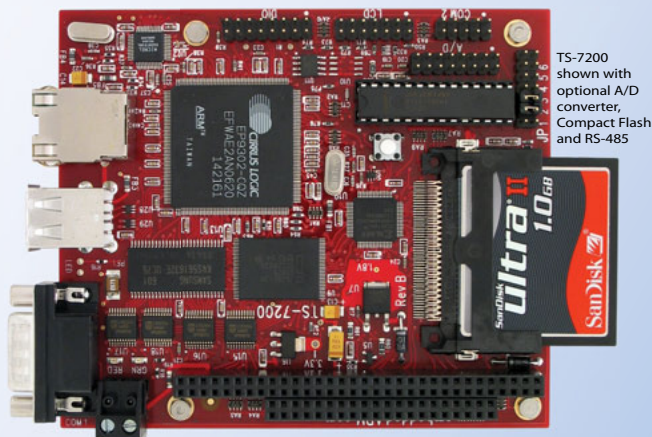
High-End Performance with Embedded Ruggedness



TS-7800 500 MHz ARM9

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

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



200 MHz ARM9 Power as low as 1/4 Watt

▪ options include:

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

- 8 boards, over 2000 configurations **\$99**
as low as
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, Debian
- SD card option
- VGA video
- LCD ready

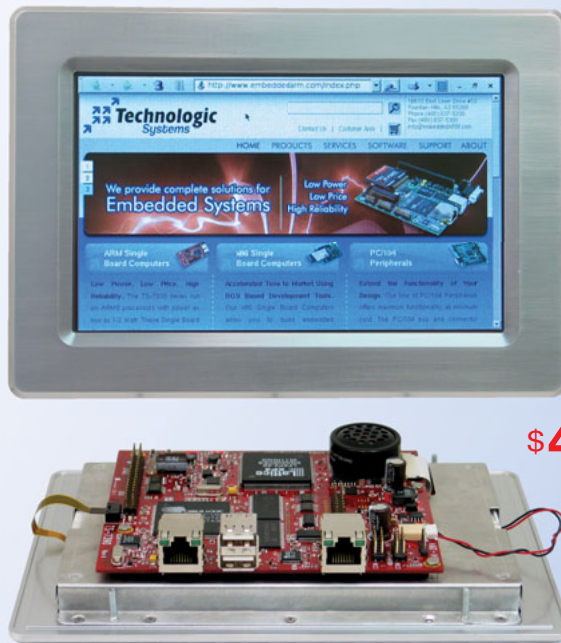
- Over 20 years in business
- Open Source Vision
- Never discontinued a product
- Engineers on Tech Support

- Custom configurations and designs w/ excellent pricing and turn-around time
- Most products stocked and available for next day shipping

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

Featured Products and PC/104 Peripherals

NEW!



\$449
qty 1

TS-TPC-7390

200MHz ARM9 Touch Panel Computer

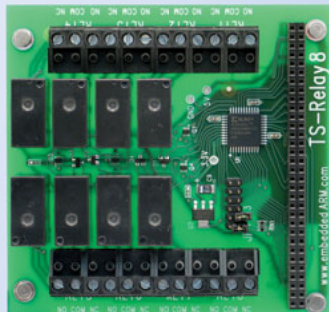
- Low Power, Industrial Quality Design
- Mountable aluminum frame
- 64MB SDRAM (128MB opt)
- 512MB Flash w/ Debian Linux
- Programmable FPGA- 5K LUT
- 7" Color TFT-LCD Touch-Screen
- 800x480 customizable video core
- Dedicated framebuffer- 8MB RAM
- Audio codec with speaker
- Boots Linux 2.6 in about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

NEW!

TS-RELAY8

Eight Software Controlled Relays

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



\$89
qty 1

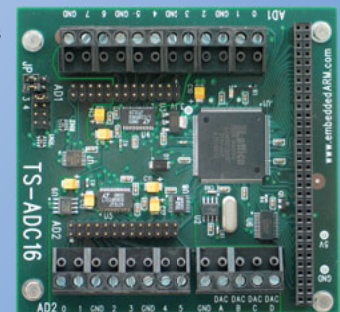
NEW!

TS-ADC16

ADC, DAC and Digital I/O

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

\$169
qty 1



see our website for x86 SBCs, peripherals and option details



We use our stuff.

Visit our TS-7800 powered website at
www.embeddedARM.com



Bonus Video Portal:

Circuit Cellar recurring author Chris Coulston presents the first installment of the DIY guide to building a Goldsprints racing system.

Circuit Cellar video from the 16th annual Trinity College Fire Fighting Home Robot Contest 2009.



Sponsor Events – click to see video

Upcoming Microchip MASTERS 2009



Recent Renesas Devcon



Old Tech, New App

At first glance, this issue looks more like one you would've read in the mid-'90s rather than in mid-2009. The topics of DOS, sealed lead-acid (SLA) batteries, NTSC, C code, digital signal processing, DIY power supplies, and JTAG don't really scream "cutting-edge technology!" But when presented by *Circuit Cellar* authors who spend their days and nights dreaming up innovative embedded apps, they don't scream "old school ideas" either. Yes, we're up to something here.

Basically, we've assembled a group of articles about new ways of developing and tweaking proven older technologies to meet modern design goals. For instance, on page 16, Dale Wheat describes an MCU-based meter he built for SLA battery charge testing. DJ Delorie finishes his USB GPIO pod series with information about downloading a JTAG programming application to program a CPLD circuit (p. 24). In "DOS in the 21st Century," Andrew Mitz and Jon Daley explain how they use DOS as an operating system for embedded applications (p. 36). Turn to Tom Struzik's article on page 44 to learn how he saved money by using a transformerless power supply, rather than expensive transformers or converters, in a recent light switch design project. On page 52, Jeff Bachiochi describes how he uses a Propeller to live in a world without NTSC. After saying farewell to NTSC, check out George Martin's article about C code for the FAT file system (p. 60). Tom Cantrell ends the issue with an article about new ways for tackling DSP apps (p. 65). The part he presents "isn't your father's DSP."

Just so I don't feel left out, let me describe a new twist on an old technology. It is called *Circuit Cellar Digital Plus*, which is the newest version of *Circuit Cellar's* ever-evolving electronic edition. I encourage all *Circuit Cellar* readers to give *Digital Plus* a try! Here's a short intro to *Digital Plus*:

1. *Digital Plus* is a replica of the print magazine in digital form, "plus" it includes digital-only extras like bonus articles, tutorials, videos, photos, advertisements, interviews, and more.
2. You can easily view *Digital Plus* in two ways. One, view it through an online flip book reader, which does not require special software downloads. Click on the "?" icon to get instructions about using the flip reader. Two, download each issue as an easy-to-print PDF. Simply click on the "Adobe Acrobat" icon to download the PDF.
3. An e-mail will notify *Digital Plus* subscribers when a new issue is available. New issues (and some old) are posted at www.circuitcellar.com/DP/.

If you want to subscribe to *Digital Plus*, renew a subscription, or update your information, feel free to visit www.circuitcellar.com/DP at any time. Enjoy!

cj@circuitcellar.com

C. Abate

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FOUNDER/EDITORIAL DIRECTOR

Steve Ciarcia

CHIEF FINANCIAL OFFICER

Jeannette Ciarcia

MANAGING EDITOR

C. J. Abate

MEDIA CONSULTANT

Dan Rodrigues

WEST COAST EDITOR

Tom Cantrell

CUSTOMER SERVICE

Debbie Lavoie

CONTRIBUTING EDITORS

Jeff Bachiochi

Ingo Cyliak

Robert Lacoste

George Martin

Ed Nisley

CONTROLLER

Jeff Yanco

ART DIRECTOR

KC Prescott

NEW PRODUCTS EDITOR

John Gorsky

GRAPHIC DESIGNERS

Grace Chen

Carey Penney

PROJECT EDITORS

Gary Bodley

Ken Davidson

David Tweed

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.rakoczypphoto.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.

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

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

Entire contents copyright © 2009 by Circuit Cellar, Incorporated. All rights reserved. Circuit Cellar is a registered trademark of Circuit Cellar, Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited.



With compilers that cut power consumption

Your MCU can go green!

HI-TECH C[®] PRO
ANSI C Compilers

featuring Omniscent Code Generation™ (OCG)

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

HI-TECH C PRO compilers for Microchip PIC10/12/16/18/32 MCUs and other microcontrollers.

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

HI-TECH
S O F T W A R E

OCG

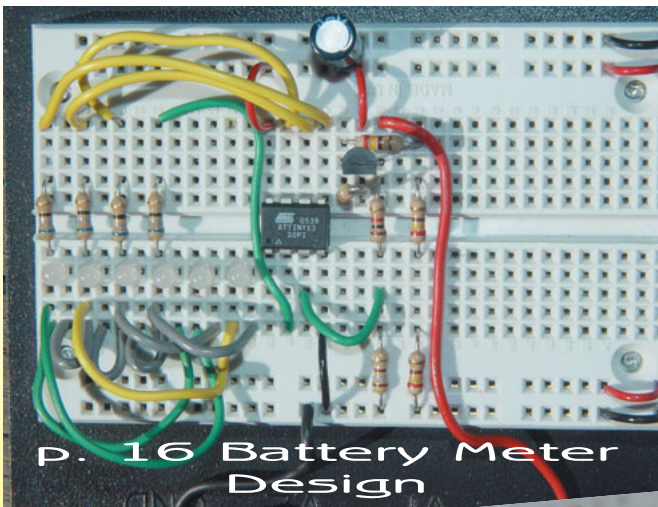
www.htsoft.com • sales@htsoft.com • +61 7 3722 7777

All trademarks are the property of their respective owners.

INSIDE ISSUE

226

May 2009 • Measurement & Sensors

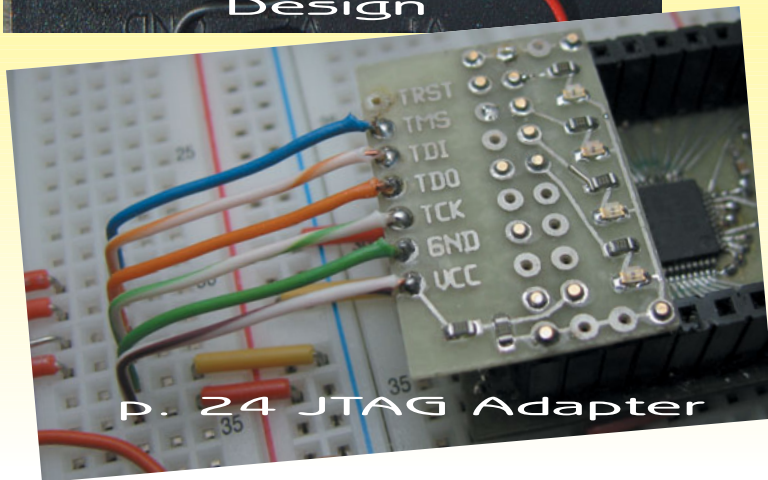


- 16** **Smart Lead-Acid Battery Meter**
An MCU-Based "Gauge" for SLA Batteries
Dale Wheat

- 24** **Construct a USB GPIO Pod (Part 2)**
USB JTAG Module
DJ Delorie

- 36** **DOS in the 21st Century**
A USB Flash Drive Reader for MCUs Works for DOS
Andrew Mitz & Jon Daley

- 44** **Transformerless Power Supply**
Tom Struzik



- 52** **FROM THE BENCH**
A World Without NTSC
Bridge the Gap Between NTSC and VGA
Jeff Bachiochi

- 60** **LESSONS FROM THE TRENCHES**
FAT File System Review (Part 2)
C Code for the File System
George Martin

- 65** **SILICON UPDATE**
Whistle While You Work
A Look at a Modern DSP
Tom Cantrell

- TASK MANAGER** **4**
Old Tech, New App
C. J. Abate

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

- TEST YOUR EQ** **15**

- CROSSWORD** **74**

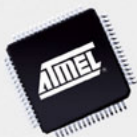
- INDEX OF ADVERTISERS** **79**
June Preview

- PRIORITY INTERRUPT** **80**
It's All About the Content, Stupid!
Steve Ciarcia

Hammer Down Your Power Consumption with picoPower™!



THE Performance Choice of Lowest-Power Microcontrollers



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

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

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

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



NEW SEALED LIMIT SWITCHES

Many limit switches must function in harsh conditions or outdoors—environments which may damage switches and lead to frequent change outs. Even limit switches that are IP67 sealed often require wiring replacements. This could mean changing several meters of cable on outdoor equipment.

In order to facilitate faster maintenance, the newly introduced models **MP730** through **MP760** sealed limit switches with the ability to connect through an M12x1 connector—which maintains IP67 protection on the switch—are now available. The connector can be mounted on the side, but a bottom-mount option is also available.

The MP700 series limit switches are available with a metal housing and a five-pin connector or with a plastic housing and a four-pin connector. Mounting holes can be placed either 20 mm or 25 mm apart for installation flexibility. The switches are additionally supplied with a large variety of actuator options and conform to IEC 947-5-1 and EN 60 947-5-1.

Pricing for the switches starts at around **\$40**.

Microprecision Electronics
www.microprecision.us



NEW OSCILLOSCOPES FOR DEBUGGING

The **WaveAce** series of digital oscilloscopes is a line of portable, affordable, easy-to-use oscilloscopes in the 60- to 300-MHz range. The WaveAce improves troubleshooting and shortens debug time by providing unique features such as long memory, a color display, extensive measurement capabilities, and advanced triggering. A streamlined, time-saving user interface provides quick access to all important controls. With its USB host and device ports, the WaveAce easily connects to a memory stick, PC, or printer. The variety of standard acquisition modes and advanced triggers simplifies capturing even the most complex waveforms, making the WaveAce a valuable tool for design, debug, and troubleshooting.



The new WaveAce is available in two-channel models with bandwidths of 60, 100, 200, and 300 MHz. All models have color displays. With a maximum sample rate of 2 GS/s and up to 8 kpts/ch memory, the WaveAce is a performance leader in this class of portable oscilloscopes. The long memory enables users to capture full sample rate acquisitions that are two to three times longer than the competition.

Improving how a user can understand and analyze waveforms, the WaveAce has 32 built-in automated parameters, including advanced timing parameters for skew, phase, and edge-to-edge measurements between channels. Additional features—such as Pass/Fail testing, user-definable digital filters, and a waveform sequence recorder—all simplify and shorten debug time.

Prices for the WaveAce series start at approximately **\$1,200**.

LeCroy Corp.
www.lecroy.com

NEW PRODUCT NEWS

Edited by John Gorsky

LOW-POWER COMPUTER CAPTURES & TRANSMITS REMOTE DATA

The **DataMover** offers an economical solution to remote data capture and transmission. By integrating an optional GPS receiver, GPRS cell modem, and Sleep mode power controller in a die cast aluminum enclosure, the DataMover is capable of logging and transmitting data from remote or inaccessible locations. With power consumption of less than 1 W during operation and less than 3 mW in Sleep mode, the DataMover can be battery- or solar-powered with minimal expense.

The DataMover single board computer comes standard with RS-232/RS-485 serial ports, conditioned analog-to-digital converter inputs, driver outputs, optically isolated inputs, and simple TTL I/O. The software supplied includes drivers and a read/write file system.

The DataMover controller is **\$142** in 100-piece quantities. Development kits are available for **\$229**.

JK microsystems
www.jkmicro.com



HALL-EFFECT MAGNETIC POSITION SENSORS

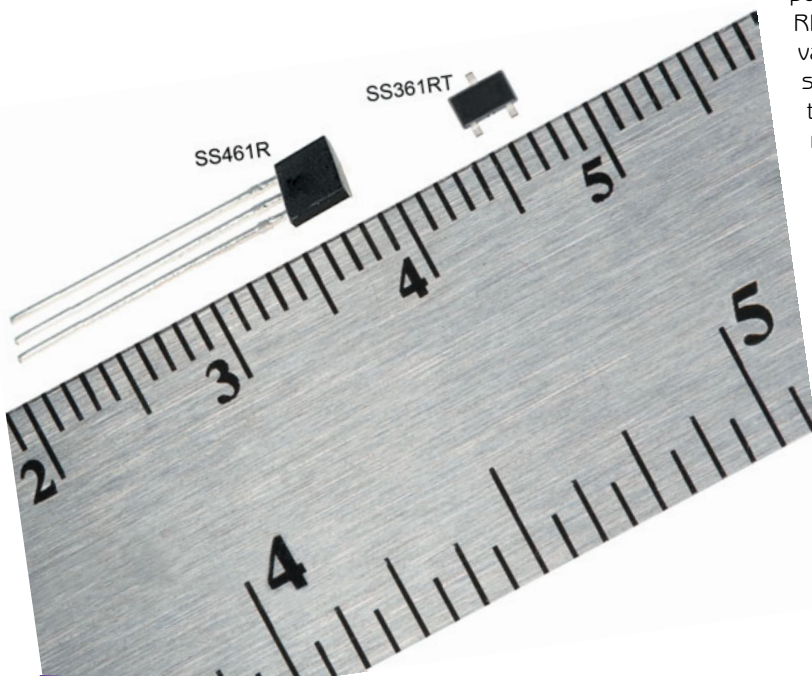
The **SS361RT** and **SS461R** are bipolar latch, Hall-effect magnetic position sensors with enhanced sensitivity, which enables the use of less expensive magnets in some applications. The SS361RT sensor's small footprint takes up less space on the printed circuit board, typically allowing for a reduction in costs. Additionally, the SS361RT sensor is supplied on tape and reel, allowing for automated, high-volume, lower-cost pick-and-place assembly.

The SS361RT and SS461R sensors are designed for potential use in industrial applications (e.g., speed and RPM sensing, tachometer, flow-rate sensing, valve/damper position, brushless dc motors, variable speed drives, motor and fan control, and robotics control), medical applications (e.g., motor assemblies and medication dispense control), and transportation motion control applications (e.g., RPM sensing, tachometer, motor and fan control, electric window lifts, convertible roof positioning, and transmission positioning).

The SS361RT and SS461R sensors feature a 3-V supply voltage capability for low-voltage applications. The sensors are available in two package styles, providing application flexibility. The SS361RT's miniature SOT-23 surface-mount package utilizes a tape and reel format. The SS461R's flat TO-92 style is available in bulk (1,000 units per bag).

Please contact your Honeywell distributor for pricing.

Honeywell International
www.honeywell.com



PROGRAMMABLE SMART CARD

The **BasicCard ZC3.12** is the first programmable smart card with 2 KB of EEPROM for a price less than \$1.30. This micro-processor-based smart card is programmable in Basic. Applications like E-Purse, Identification Card, Medical Card, Gift and Loyalty Card, and so on, can be developed in BASIC and are fully compatible to the ISO 7816 standard. Encryption technology like AES, DES, or ECC is included.

Programming is accomplished through a bidirectional I/O contact. Communication takes place at 9,600 bps or more, according to the T=0 and T=1 protocols defined in ISO/IEC standards 7816-3 and 7816-4. (The latest cards also implement the contactless ISO14443 protocol.) This is completely invisible to the Basic programmer. All you have to do is define a command in the card and program it like an ordinary Basic procedure. Then you can call this command from a ZC-Basic program running on the PC. Again, the command is called as if it was an ordinary procedure.

The BasicCard operating system takes care of all the communications for you. It will even encrypt and decrypt the commands and responds if you ask it to. All you have to do is specify a different two-byte ID for each command that you define.

A development kit for the BasicCard ZC3.12 is available for about \$78. The BasicCard ZC3.12 is priced at \$1.30 in quantities of 10 or more.

ZeitControl cardsystems GmbH
www.basiscard.com



NEW 8-, 16-, AND 32-BIT INDUSTRIAL MCUs

Three new series microcontrollers with embedded flash technology for industrial applications are now available. All are designed to meet the performance and reliability requirements of a wide range of drive applications across a variety of industries.

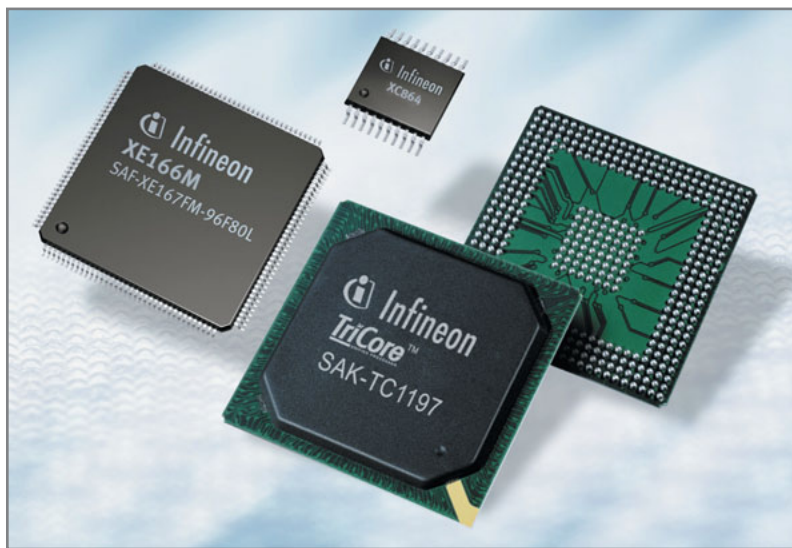
The new **XC864** series is designed for cost-sensitive motor control applications. The XC864 series combines in a small footprint TSSOP-20 package a standard 8051 core with on-chip flash memory of up to 4 KB and powerful on-chip peripherals. The XC864 series features an internal clock source and embedded voltage regulator, supporting single voltage supply of 3.3 or 5.0 V.

The **XE16xM** series of MCUs are designed for use in real-time industrial drive applications, such as servo drives, HVAC compressors and blowers, pumps, advanced sensing and power supplies. The XE16xM series provides additional advanced safety features for SIL3 applications and 64-pin devices (the XE162M series) with a small footprint. The highly configurable serial interface USIC supports multiple protocols, such as Asynchronous and Synchronous Serial Interfaces (ASC, SSC), I²C, LIN, and communication to SD memory cards.

The **TC1167** and **TC1197** are designed for use in demanding real-time industrial applications, such as multi-axis controllers for up to five three-phase complementary PWMs. They support multiple modulation strategies such as Space Vector Control or Direct Torque Control and provide multiprocessor support for reliability and safety.

In quantities of 20,000 units, the 8-bit microcontroller XC864 costs about \$1.05 per unit. The price of the 16-bit XE16xM series starts with the XE162M at about \$7 each. The unit price of the 32-bit microcontroller TC1167 is about \$20.16. The TC1197 is about \$26.46.

Infineon Technologies AG
www.infineon.com



CO GAS SENSOR MODULE

The CO Gas Sensor Module (#27931) is designed to allow a microcontroller to determine when a preset CO gas level has been reached or exceeded. Interfacing with the sensor module is done through a four-pin SIP header and requires two I/O pins from the host microcontroller. The sensor module is mainly intended to provide a means of comparing carbon monoxide sources and being able to set an alarm limit when the source becomes excessive.



The module is built around the MQ-7 CO sensor and features a SIP interface. Included in the kit is the module, a potentiometer adjustment tool, and complete documentation. The module requires 5 VDC at 165 mA (Purge Phase)/50 mA (Sense Phase). The unit is 1.50" H x 1.00" W x 1.00" D and operates over a 0 to 70°C range.

The CO Gas Sensor Module costs \$29.99.

Parallax, Inc.
www.parallax.com

NEW HINGED POWERSTRIP

Samtec has expanded its rugged PowerStrip High Power Interconnect family with a hinged version of its 25-A PowerStrip/25 system. The FMPS/FMPT Series connectors are ideal for applications that call for blind mating or where PCB real estate is tight.

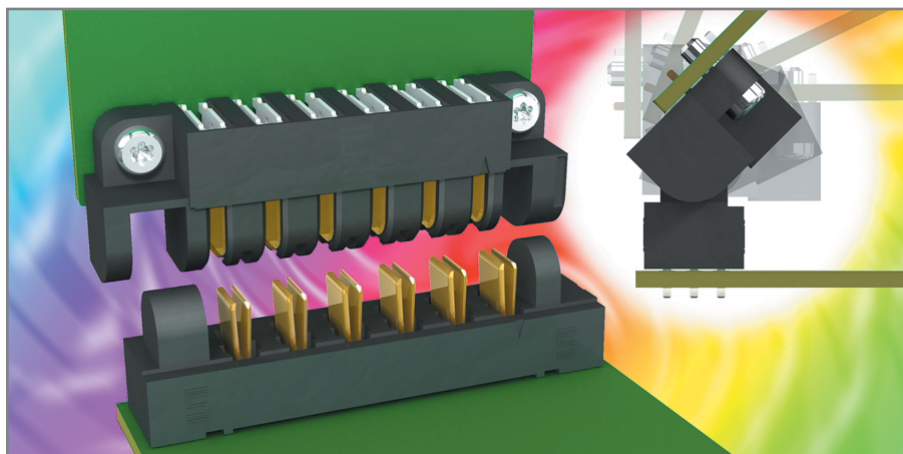
The hinged PowerStrip/25 Socket Terminal system (FMPS/FMPT Series) is a unique system that allows for 90° or horizontal (planar) board-to-board interfaces and a 90° mating radius. The design's flexibility allows the connectors to be mated at an angle convenient for production and then rotated to the final, permanent required orientation. It is available with up to eight power pins on 0.200" (5.08-mm pitch) and can achieve a robust current rating up to 25 A per contact at 75°C. Optional locking clips and a screw-down configuration for the socket are also available.

The hinged PowerStrip/25 joins High Power 35 A PowerStrip/35 connector systems, 25 A PowerStrip/25 connectors, discrete wire cables, and the 10 A PowerStrip/10 modular connector system.

The full line of Rugged/Power solutions also includes Power Mate, Mini Mate and Q Series High Speed/Power connectors and cables, and a variety of Rugged/Power Headers and Sockets.

It costs about \$0.25 per 25-A pin in quantities.

Samtec, Inc.
www.samtec.com



NPN



Software Trials
Available

JTAG (J-Link™)

+++ ARM7/9 Cortex M3 +++

- USB to JTAG
- Fast 720kb/s Download Speed
- Serial Wire Debug (SWD) Support
- Multicore Debugging Support
- Auto JTAG Speed Recognition

The J-Link can be coupled with a number of available software modules to fit your application needs.

J-Flash is a stand-alone application used with the J-Link to program internal and external flash devices.

J-Link RDI permits the use of the J-Link with an RDI compliant debugger.

J-Link GDB Server is a remote server for GDB.

J-Link Flash Download is a module used to download your program into flash even if your debugger does not have a flash loader.

J-Link Flash Breakpoint permits you to set an unlimited number of software breakpoints while debugging in flash.

J-Link SDK is a standard DLL that extends the full functionality of the J-Link to your proprietary application.



We also offer a JTAG isolator which can be used to offer electrical isolation between your target hardware and the J-Link. This is essential when the development tools are not connected to the same ground as the application. It is also useful to protect the development tools from electrical spikes that often occur in some applications, such as motor control applications.

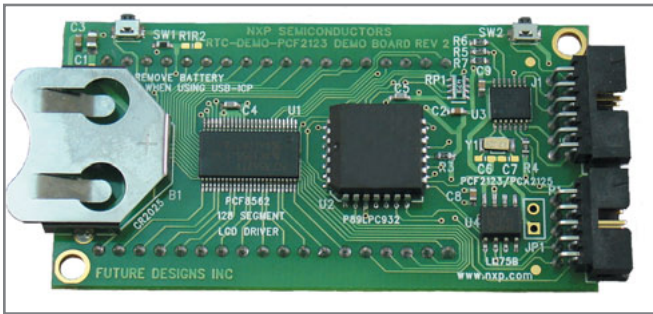


Special Offer

Includes the J-Link,
J-Link GDB Server, and
the J-Link Flash Download.
www.segger-us.com/ncu.html

J-Link Non-Commercial (NCU) Bundle

www.segger.com



PCF2123 DEMO BOARD

The **RTC-DEMO-PCF2123** is a stand-alone demonstration platform for the new NXP SPI-based PCF2123. With a typical standby current of only 100 nA at 2.0 V, the PCF2123 is the lowest power real-time clock currently available. The unit is powered by a single 3-V coin cell battery and includes an eight-character alphanumeric LCD with 128-segment I²C LCD driver and a P89LPC932 8-KB flash microcontroller. Also included are an LM75B I²C temperature sensor and two miniature push buttons for user control. An additional user interface is available via an expansion header and the flash microcontroller is user programmable via a 10-pin ICP header.

The RTC-DEMO-PCF2123 is available from stock for **\$69**.

Future Designs, Inc.
www.teamfdi.com



STRONG, SILENT INDUSTRIAL PC

Ideal for a variety of embedded applications, the **AMOS-3000** is a robust, custom-designed system based on the ultra compact and versatile EPIA-P700 Pico-ITX board. Measuring only 13.5 cm x 4.5 cm x 13.1 cm, the AMOS-3000 is strong, durable, and heat-efficient—yet it is tiny enough to fit in the palm of your hand. The system can be installed easily using simple table, wall and VESA mounting options.

The AMOS-3000 is based on an EPIA-P700 Pico-ITX board, powered by either a 1-GHz C7 processor or an ultra-low-voltage 500-MHz Eden processor coupled with the VX700 unified digital media chipset, supporting up to 1 GB of DDR2 SO-DIMM system memory. The system has a certified operating temperature of -20° to 60°C, vibration tolerance of up to 5 G_{RMS}, and a shock tolerance of up to 50 G.

Storage is provided through a 44-pin IDE interface for Disk on Module Flash drive. An optional storage subsystem expansion chassis offers support for a standard 2.5" SATA drive. An on-board RJ-45 connector provides Gigabit networking, while the VT1708A brings HD audio. Additional features include four USB 2.0 ports, on-board GPIO port, COM port, and programmable watchdog timer. System LED indicators are provided for power and HDD activity.

The AMOS-3000 has a list price of **\$380**.

VIA Technologies, Inc.
www.via.com.tw

THE ORIGINAL SINCE 1994
PCB-POOL.COM
 Beta LAYOUT

Specializing in Quickturn Proto's

- Internet pioneers with 15 years experience
- Instant online Quotations & Ordering
- From Singlesided to 6 layers ML
- Leadtimes from 48 hrs
- Full DRC included on all orders
- High Quality prototypes at LOW cost's

Simply send your layout files
 and order online

www.pcb-pool.com

TollFree USA: 1877 3908541 Email: sales@pcb-pool.com

NPN

SENSOR FILM MEASURES LOW CONTACT PRESSURES

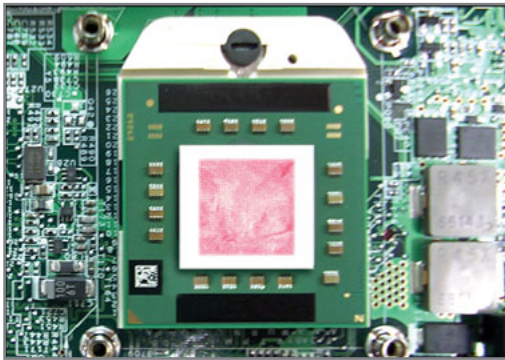
Pressurex Zero is an easy-to-use, tactile pressure-indicating sensor film. Pressurex Zero characterizes contact surface pressure down to an extremely low 7.2 psi (0.5 kg/cm²). It is simply placed between contacting or mating surfaces to instantly and accurately measure and map pressure magnitude and distribution. Variations in contact surface pressure are immediately visible by the impression made on the film. The range for Pressurex Zero is 7.2 to 28 PSI (0.5 to 1.97 kg/cm²).

Pressurex comes in the form of a thin plastic sheet (4 or 8 microns), physically similar in thickness to paper, and is available in eight different pressure ranges. When placed between contacting surfaces, it instantaneously and permanently changes color directly proportional to the actual pressure applied. The precise pressure magnitude (psi or kg/cm²) is easily determined by comparing color variation results to a color correlation chart (conceptually similar to interpreting Litmus paper). Pressurex can also be scanned through one of Sensor Products's optical imaging systems. The film, which is available in eight different pressure ranges, is used in the design, manufacture, calibration, and quality control of many products.

Pressurex is flexible, which enables it to conform to curved spaces. It is ideal for invasive, intolerant environments and tight spaces that are not accessible to conventional electronic transducers.

A 270 mm x 3 m roll of Pressurex costs \$698.

Sensor Products, Inc.
www.sensorprod.com



ULTRASONIC SENSORS FEATURE EXTENDED RANGE AND COMPENSATION

The **U-GAGE T30UX** is an ultrasonic sensor for challenging applications. With its extended functional range and advanced temperature compensation, the T30UX sensor withstands hostile environments, providing superior ultrasonic sensing to solve even the toughest application challenges. The T30UX sensor features a robust U-GAGE housing to resist harsh environmental conditions and provides a variety of model options to meet a broad range of application requirements.

The T30UX ultrasonic sensor is available in a choice of three ranges for reliable sensing from 100 mm to 3 m and delivers high-accuracy performance with built-in temperature compensation across a wide range of ambient temperatures. Temperature compensation enables the sensor to self-correct for the temperature in its environment and maintain the highest accuracy in changing conditions. This robust sensor resists harsh environments with a rugged IP67 (NEMA 6) housing and fully encapsulated electronics, and its 30-mm threaded barrel and a selection of mounting brackets allow simple installation. Additionally, models are available with a single analog or user-configurable discrete output, and all models provide highly visible LED status indicators for power, signal strength, and output.

The sensor starts at \$249 for U.S. orders and depends on the configuration needed.

Banner Engineering Corp.
www.bannerengineering.com



Development Solutions for ARM, 8051 & XE166 Microcontrollers

Microcontroller Development Kits

C and C++ Compilers

Royalty-Free RTX Kernel

µVision Device Database & IDE

µVision Debugger

Complete Device Simulation

Examples and Templates

**Keil PK51, PK166,
and MDK-ARM**
support more than 1,700
microcontrollers
www.keil.com/dd

RTOS and Middleware Components

RTX Kernel Source Code

TCPnet Networking Suite

Flash File System

USB Device Interface

CAN Interface

Examples and Templates

Keil RL-ARM and ARTX-166
highly optimised, royalty-free
middleware suites
www.keil.com/rtos

www.keil.com
1-800-348-8051

NPN

QUICK-CONNECT WIRING SYSTEM

The new M16 powerfast wiring system is specifically designed for machine power distribution and motor control. The quick-connect M16 powerfast system provides a time- and cost-saving replacement for traditional hard-wiring installations and complies with NFPA 79: Electrical Standard for Industrial Machinery. These two-, three-, and four-pin connectors and tees provide up to 18 A in a compact form factor.

The cordsets are offered with Tray Rated, exposed run PVC flexlife cable. All connectors deliver IEC IP 67 protection and are rated for 600 V and up to 18 A. Tees are available with simple connectors or with branches.

Like most TURCK cordsets, the M16 powerfast line offers male or female options, straight connectors, standard and custom lengths, and pigtails or extensions. To complete the system, fully encapsulated mating receptacles with nickel-plated brass housing and 0.5" to 14 NPT, 0.375" to 18 NPT, M18, and M20 mounting threads are available.

Prices are dependent on the configuration, please contact TURCK directly.



TURCK, Inc.
www.turck-usa.com

NPN



Embedded & Network Computing Technologies

Tiny, Light & Powerful All In One!



Embedded Computer
Tinycore form-factor (36 x 41 mm)

ATMEL AT91SAM9G20 @ 400MHz

256MB NAND Flash (8bits),
64MB SDRAM (32bits @ 133 MHz)
USB Device, Serial DBGU & 2 Expansion Ports.

www.calao-systems.com

System on Module Internet Appliance Engine

SoM-9307

- EP9307 ARM9 200Mhz CPU
- 3 Serial Ports & 2 SPIs
- Up to 40 Digital GPIOs
- 3 USB 2.0 Host Ports
- I2S Audio Interface
- 10/100 BaseT Fast Ethernet
- SD/MMC Flash Card Interface
- Up to 64 MB Flash & 128 MB RAM
- Graphic LCD Interface with 2D Acceleration
- Linux with Eclipse IDE & WinCE 6.0
- 8 12-Bit A/Ds & 4 16-Bit Timer/Counter
- Small, 144 pin SODIMM form factor (2.66 x 2.38")



2.6 KERNEL



The SoM 9307 uses the same small SODIMM form factor utilized by other EMAC SoM modules, and is the ideal processor engine for your next design. All of the ARM9 processor core is included on this tiny board including: Touchscreen Interface, Flash, Memory, Serial Ports, Ethernet, I2S Audio Interface, PWMs, Timer/Counter, A/D, Digital I/O lines, and more. Like other modules in EMAC's SoM product line, the SoM 9307 is designed to plug into a custom or off the shelf Carrier board containing all the connectors and any additional I/O components that may be required. The SoM approach provides the flexibility of a fully customized product at a greatly reduced cost. Single unit pricing starts at \$150.

<http://www.emacinc.com/som/som9307.htm>

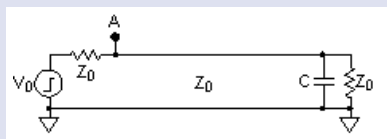
Since 1985
OVER
24
YEARS OF
SINGLE BOARD
SOLUTIONS

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

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

Robert Lacoste recently wrote about transmission lines and time-domain reflectometry (TDR) in Circuit Cellar issues 224 and 225. Let's explore that topic a little further.

Problem 1—Suppose we use a step waveform (instead of a narrow pulse) to excite the transmission line circuit shown below. What sort of waveform would you expect to see at point A?



Problem 2—Can you reconcile your answer to the previous question with the waveform created by pulse excitation that Robert Lacoste discussed in Circuit Cellar 225?

Problem 3—Suppose you have a circuit that is capable of producing step waveforms with very fast rise times. How can you use a transmission line to convert the steps to narrow pulses?

Problem 4—Joe is a ham radio operator, and he likes to operate on the 6-m band (50 to 54 MHz). One day Bill is visiting Joe's "shack," and notices that Joe has a tee adapter on the transmitter's antenna jack, and in addition to the cable to the antenna, there is a second piece of coax attached to the tee that doesn't seem to go anywhere. It appears to be about a foot long.

Bill asks Joe about it, who laughs and says, "Oh, that helps keep me legal!" What's going on here?

David Tweed contributed the four problems and answers in this issue.

What's your EQ?—The answers are posted at www.circuitcellar.com/eq/
You may contact the quizmasters at eq@circuitcellar.com

FRONT PANELS & ENCLOSURES

Customized front panels can be easily designed with our free software
Front Panel Designer

- Cost-effective prototypes and production runs
- Wide range of materials or customization of provided material
- Automatic price calculation
- Fabrication in 1, 3 or 7 days

Sample price:
\$43.78 plus S&H

FRONT PANEL EXPRESS

www.frontpanelexpress.com
(206) 768-0602

RF Specialists

RF Modules

From Part 15 to Part 90 Compliant
Narrow Band FM, UHF Multi-Channel

GSM/GPRS

M2M Solutions
GSM/GPRS modules and modem series

Industrial Bluetooth

OEM, Modules, Wireless Device Servers, RS-232
Long range options, low cost

Data Loggers

Stand Alone and
Wireless Mesh Networking Logger

GPS

OEM Modules and USB ZigBee Sticks,
Mesh Networks

ZigBee Pro

OEM Modules and USB ZigBee Sticks,
Mesh Networks

Wi-Fi

RS232 / 422 / 485 to Wi-Fi Adapter
Connect Data Acquisition Equipment through Serial Port to Wi-Fi network

LEMOS INTERNATIONAL

www.lemosint.com
866.345.3667
sales@lemosint.com

Smart Lead-Acid Battery Meter

An MCU-Based “Gauge” for SLA Batteries

Think sealed lead-acid (SLA) batteries are a thing of the past? Think again. They are bound to be around for years to come. Dale designed a state-of-charge meter for his SLA batteries. His multicolored LED meter simplifies battery charge testing.

Rechargeable batteries are found in almost all portable, mobile, and mission-critical electronic equipment. New technologies are storing more energy in less space than ever before. Sealed lead-acid (SLA) batteries are far from the leading edge of available options: they are heavy, poisonous, and lack energy density compared with newer options. Despite these apparent drawbacks, they are *everywhere*. Like COBOL, they will be with us for some time.

SLA batteries are especially popular in hobby robotics as well as hobby electronics in general. Their ubiquity in surplus channels makes them an easy choice for beginning roboticists. Low initial cost and dependable, well-documented performance endear them to novices and seasoned professionals alike. It also doesn't hurt to find that many standardized sizes and capacities are still being produced all over the world and will be for the foreseeable future.

Charging characteristics increase their appeal: A constant voltage charger is often a good first project for the curious electronics

hacker. Off-the-shelf charging solutions abound and remain competitively priced.

A FUEL GAUGE

How much charge is left in the battery? I'm glad you asked! There are several methods available for determining the “state of charge” (SoC) or, conversely, the “depth of discharge” (DoD). I think I've seen those acronyms before, somewhere.

The most direct method is to simply measure the battery voltage and say, “Lo, here is the remaining charge.” Fortunately for battery users and unfortunately for us battery measurers, most batteries have a relatively flat discharge curve. Lithium-based chemistries are the most

Terminal voltage	State of charge
12.65	100%
12.45	75%
12.24	50%
12.06	25%
11.89 or less	Discharged

Table 1—This is the state of charge versus voltage for a sealed lead-acid battery at rest. These values were taken from “Battery University,” a series of informative, on-line articles about rechargeable batteries by Isidor Buchmann and sponsored by Cadex Electronics.

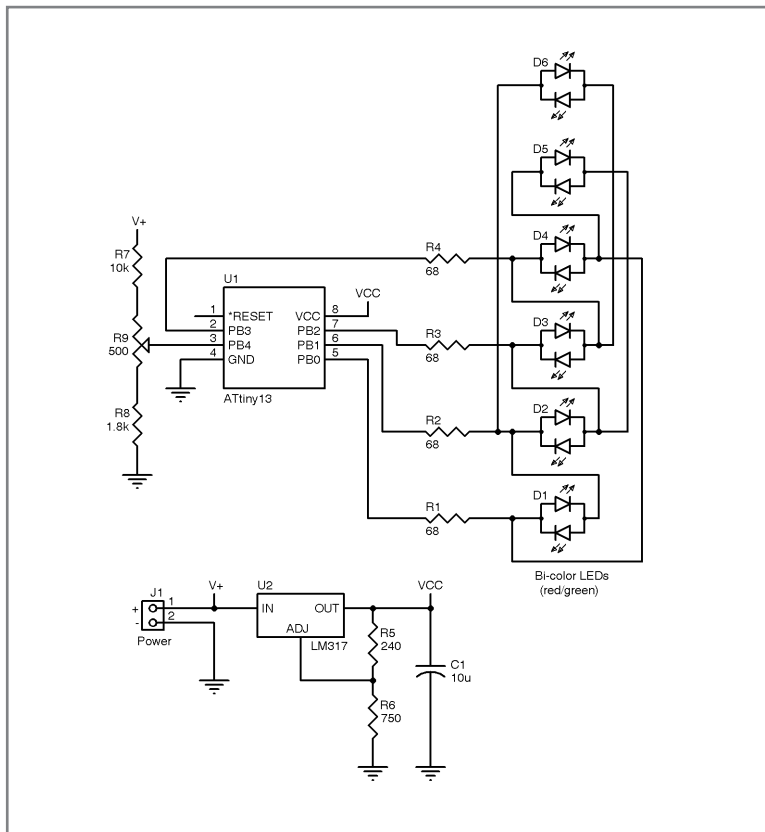


Figure 1—This is the smart SLA battery meter. I originally drew this using Advanced Circuits's free/gratis PCB Artist software. It enabled me to enter the schematic, lay out the PCB, then quote and order PCBs, all from the same program. This is a redrawn version.

notorious for steady, stable voltage output. Curse them and their dependable, high-energy densities! We will eventually use a variation of this method to determine

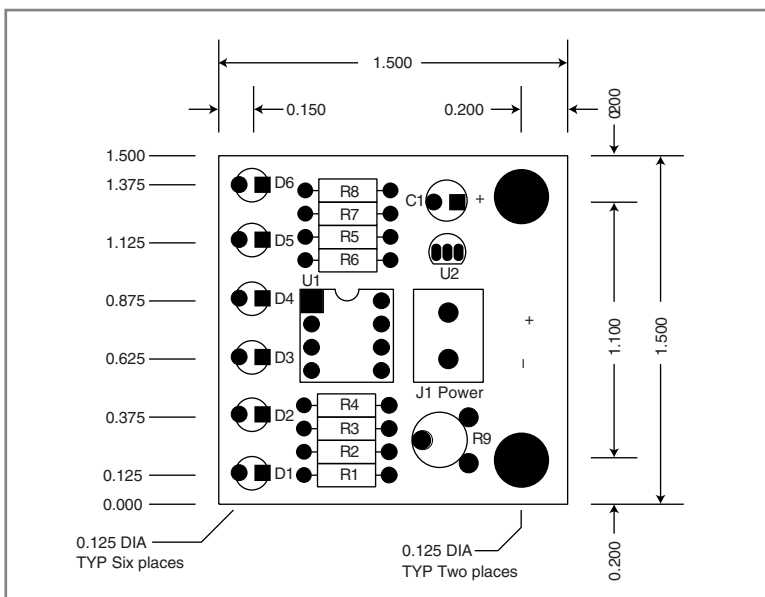


Figure 2—Here are the mechanical dimensions of the prototype PCBs. This will allow you to have a custom panel machined that will fit like a glove. The production version of the PCB will have the same dimensions but will have rounded corners.

the state of charge, after combining it with some other knowledge we have that is specific to lead-acid batteries.

Once upon a time, before SLA batteries were sealed, the specific gravity of the electrolyte could be measured and used as a good indication of the state of charge. You dipped an industrial-looking turkey baster containing floaties into the different cells and slurped up a sample electrolyte using the high-tech squeeze bulb on the end. If you had a steady hand and good lighting, you probably didn't spray battery acid on your clothes and skin. Unfortunately, I often did. I don't test those kinds of batteries any more.

A more detailed method to determine SoC is to measure how much energy has gone into the battery during the charge cycle and then keep track of how much energy is then taken out during usage. "Counting coulombs" requires both current and voltage measurements to be integrated over time and must take into consideration temperature and age of the battery. In reality, temperature and age really have to be considered in *any* comprehensive methodology. Several "coulomb counter" circuits are available from Maxim Integrated Products, Linear Technology, Texas Instruments, and others. Also, if you can count individual coulombs per second, you know how many amps are flowing in a circuit, because that is the textbook definition—but you knew that.

Although there is no *linear* relationship between voltage and state of charge, there are well-known waypoints down the discharge curve for lead-acid batteries. These voltage levels are only truly representative of the state of charge after the battery has been allowed to rest for at least 8 hours. They are summarized in [Table 1](#).

MEASUREMENT CIRCUITRY

[Figure 1](#) shows a simple data acquisition circuit that can be used to approximate the state of charge of an SLA battery. The input signal powers the entire circuit. The battery voltage is brought in via J1. From there, it is split into two paths. The first goes to an LM317L adjustable, positive-voltage regulator that is set to approximately 5 V using R5 and R6. I chose this for its wide input range, excellent output stability, small size, abundant availability, and low cost. The regulator's output is filtered by C1. Overall, this is a mundane power supply circuit. Its only special feature is that it will withstand a high input voltage and automatically shut itself down if it overheats.

The other path for the incoming power signal is to a voltage divider composed of R7, R8, and R9. R9 is a potentiometer allowing the calibration of the incoming signal to a known value.

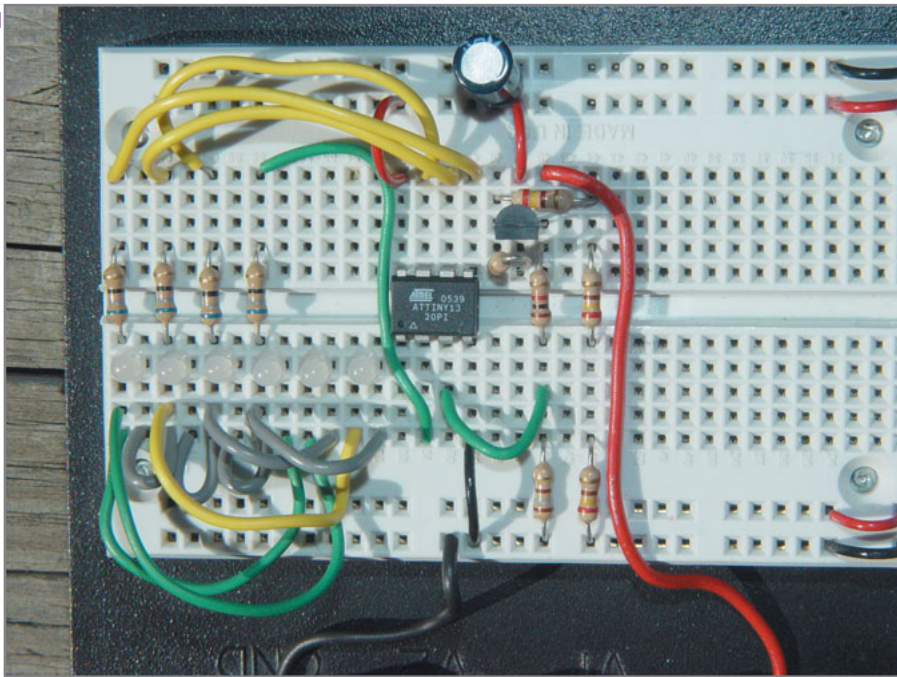


Photo 1—This is the first prototype of the battery meter built on a solderless breadboard. The extra resistors represent doing it the hard way when you don't have an exact value in stock when inspiration strikes.

This is handled in software to allow quick and reliable calibration of the battery meter in production (see [Figure 2](#)). I will describe more about the calibration procedure when I cover the firmware in detail.

The wiper pin of R9 leads into pin 3 of U1, the main brain of the circuit. I used one of my favorite Atmel AVR's, the ATtiny13, for its small size, ease of programming, versatile peripheral set, and low cost. The primary features that helped me select this part for the design were an on-chip ADC with up to 10 bits of accuracy, an internal clock, low current requirements, and reprogrammable I/O lines.

TELL THE WORLD

The AVR is happy to measure the incoming voltage and store the answer in a register or RAM location. That's not interesting at all. I wanted to add a simple but effective display using a minimum number of parts. I also wanted the display to produce "meaningful" information even with no user manual handy. A single

line of LEDs arranged as a bar graph would give a good report with a single glance; the more LEDs lit, the more power was left in the battery. Somewhere along the line I decided to spice it up and use multicolor LEDs to reinforce the message: green was good, yellow foretold doom, and red would announce failure. I suppose "multicolor" is a bit of a reach, as the LEDs I finally included were technically "bipolar" LEDs: a red

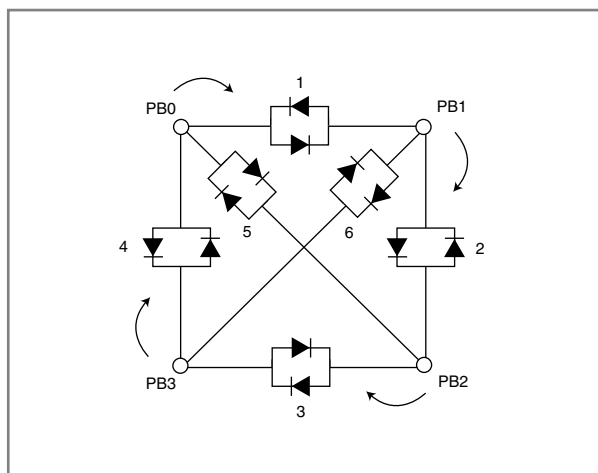


Figure 3—This is the geometric version of the LED array schematic, showing all the possible combinations of connections between the four I/O lines. That's how you light up 12 LEDs with only four I/O lines!

and a green LED die mounted inside the same package. Current applied in one direction would course through one die and illuminate it while reversing the current would light up the other side. Alternating the current rapidly would produce the illusion of both LEDs being lit simultaneously. This is *alleged* to be yellow, but it only looks yellow if you look at it quickly and sideways at the same time. It is yellow-esque, at best. Five or six multicolor LEDs would give enough of a range to make a useful display. Because each LED package is two distinct LEDs, I would have to come up with a way to drive a dozen or so LEDs with a small microcontroller.

FANCY LED MULTIPLEXING

Normally, 12 individual LEDs would require, at a minimum, 12 dedicated output driver lines to be able to individually and randomly address any LED. The ATtiny13 has six I/O lines available. Using a multiplexing technique sometimes referred to as "charlieplexing," I was able to drive the 12 LEDs using only four output pins. Maxim application note 1880 credits Charlie Allen with championing this technique within the organization. Maxim Integrated Products offers several devices that will perform that trick for you. Don Lancaster wrote about it in his "Tech Musings" column in August

2001 and gave several examples (www.tinaja.com). I'm sure I thought of it long before, but I must have neglected to write it down in my journal. Dang.

The number of LEDs that can be addressed depends on the number of I/O lines that you're willing to throw at it. The formula is $n(n - 1)$, where n is the number of I/O lines. So, four I/O lines would give you 12 LEDs (i.e., 4×3). That translates to six bipolar LEDs. The next step up would be five I/O lines and 20 normal LEDs or 10 bipolar LEDs. That's too many. The next step down

Sweet!

Introducing the MiniCore™ Series of Networking Modules

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

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

Wi-Fi and
Ethernet
Versions



MiniCore Module Development Kits

**From
\$49** Limited
time offer.



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

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



would be three I/O lines and six normal LEDs or three bipolar LEDs. That's too few. Six LEDs is just right.

Imagine the four I/O pins are wired to the four corners of a square. Now each square corner is connected by a bidirectional LED to every other corner. This makes four LEDs around the periphery of the square as well as two across the diagonals. My schematic skills omit the symmetrical beauty of this circuit and hint only at the possibilities.

We have 12 LEDs: six red and six green. They look just like regular LEDs when they are not illuminated. Some bipolar LEDs have three legs: one for each die and a common terminal. I used the two-legged variety of the 3-mm persuasion where each leg is attached to both the anode of one LED and the cathode of the other.

Instead of wiring a current-limiting resistor to each LED, I wired them to the I/O lines. This way there are only four resistors for all 12 LEDs. This works out well as there is only one LED on at a time, even when it looks like they are all illuminated. Because the current through any LED must flow through two resistors, the value of each resistor is half of what is required to limit the current through the LED. I calculated a maximum of 20 mA through each LED. I could probably have upped that number as the duty cycle is at most 1/6.

To illuminate any single LED, the I/O pin that is connected to its anode is programmed to be an output and to drive that output high. The corresponding cathode pin is also programmed to be an output and to drive its output low. The other two I/O lines are told to sit down and shut up. A more technical and accurate way to describe it would be to say that they are programmed to become inputs and not activate their internal pull-up resistors. They

effectively become tristated at this point and do not contribute to the current flow in the LED array. Each LED has its own pattern of ins, outs, highs, and lows.

WRITE THE SOFTWARE

I built a prototype of the circuit using a solderless breadboard (see [Photo 1](#)). I used an Atmel STK-500 to program the ATtiny13. Being an eight-pin DIP, it was relatively easy to pop it in and out of the prototype for programming. Anything larger would have wanted its own programming cable attached. The AVR devices have the nice capability of "in-system programming," or ISP. This allows the part to be programmed and reprogrammed *in situ*, without having to remove the part from the circuit and placed in a device programmer.

I wrote the code in C. I used the WinAVR port of the GCC compiler collection, which works well with Atmel's AVR Studio. These are all free programs, in various senses of the word (GCC is *libre*, or free as in speech, while AVR Studio is *gratis*, or free as in beer).

The first task was, as usual in embedded development, to light a single LED. This was a little more complex than the normal embedded "Hello, world!" monochrome monapixel. I had to constantly refer

to my scribbled notes as to pin placement and LED array wiring (see [Figure 3](#)).

I played with various combinations of highs and lows and ins and outs. Once I could light up individual LEDs to my satisfaction, I wrote a routine to scan through a memory map of LEDs and light them or not, over and over, too quickly for the eye to detect. This produces the illusion of multiple LEDs being illuminated at the same time. This became the main loop of the final firmware.

Now I had a medium waiting for a message. The next step was to read the ADC peripheral and determine what voltage was being presented for analysis. Naturally, reading it once was not going to be adequate, due to sampling noise and other error terms, so I set up the on-chip Timer/Counter peripheral to interrupt periodically and in the interrupt service routine set about to oversample the ADC input. Eight successive samples are read and summed then divided by eight to give a rough arithmetic average. This number is compared to a series of preset values to determine how many LEDs of each color to light up. This is where the values from Table 1 come into play.

I broke down the basic thresholds from Table 1 and added a few more of my own devising because my

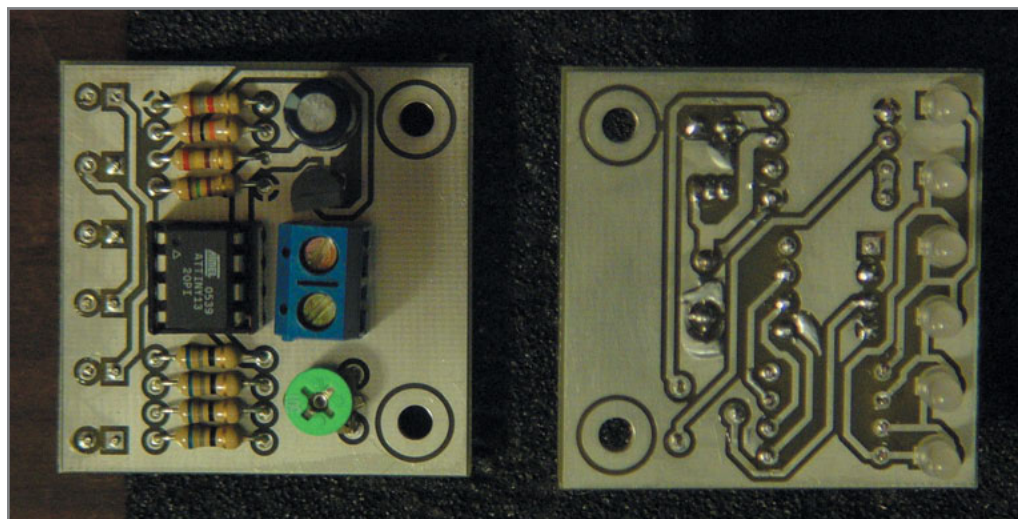


Photo 2—This is the assembled PCB. These boards made by Advanced Circuits are their BareBones special: double-sided, plated-through holes, no soldermask, no silkscreen. Believe it or not, they were built in under 24 hours! Every one perfect, every time.

The Newest Optoelectronics

EVERLIGHT

PANJIT TOUCH SCREENS

VISHAY

LEDENGIN

OPTEK

KINGBRIGHT

LITE-ON

AVAGO

LUMEX

LAMINA

SHARP

NORITAKE

TAOS

BIVAR



Kingbright

www.kingbrightusa.com

High Brightness LEDs
www.mouser.com/kingbright/a

PACIFIC SILICON SENSORS

SEOUL SEMICONDUCTOR

3M TOUCH SYSTEMS

NEWHAVEN DISPLAY



www.mouser.com

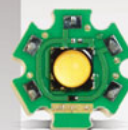
Over A Million Products Online

New Products from:



Avago
TECHNOLOGIES

PCB PolyLEDs
www.mouser.com/avagotechnologies/a



LedEngin, Inc.

LZ4-4xxx10 Serially
Connected MCPB
www.mouser.com/ledengin/a



SHARP
MICROELECTRONICS
OF THE AMERICAS

GM5BW High Power White LEDs
www.mouser.com/Sharpma/a

The ONLY New Catalog Every 90 Days

Experience Mouser's time-to-market advantage with no minimums and same-day shipping of the newest products from more than 390 leading suppliers.



a tti company

**The Newest Products
For Your Newest Designs**

(800) 346-6873

multicolored bar graph could easily express more than the five original charge states.

If the input voltage is above 13.5 V, I assume that the battery is being charged. To announce this, I have the LEDs do the happy dance, where an upward sequence of green LEDs is marched up the bar graph. This is similar to what you may have seen on many cellular phones during their recharge period, except that mine is greener and happier.

If the voltage is between 12.65 and 13.5 V, the graph shows six of six bars, or green LEDs in this case. This represents a fully charged battery with 100% of its capacity available. You have to remember that in the real world, a battery has 100% of its rated capacity only once, if you're lucky, early in its life, and that it slowly degrades over time.

When the voltage is between 12.45 and 12.65 V, it is estimated that approximately 75% or more of the power is still available. Five green dots out of six indicate this stage.

Note that 12.24 V marks the 50% line. Only four green dots are displayed at this point. When 12.06 V is measured, it tells you that only about 25% of the charge remains, and three yellow LEDs are displayed, notifying you that it might be time to start thinking about finding the charger.

When the measured voltage drops further but is still above 12 V, two yellow LEDs describe this sad state of affairs. The battery is effectively flat at this point.

While still above 11.89 V, a single red LED will be lit to announce impending loss of battery power. Anything below this is considered a battery failure and will be displayed as a blinking red light. Below 11.5 V, the unit stops displaying anything in an effort to save what little power might be left.

DUAL POWER

One of the original requirements for the meter was to measure both 12- and 24-V battery systems, because these were the most common for SLA batteries. I had originally thought of

using different resistors in the inbound voltage divider, but this would have ultimately created two meters: one for 12-V systems and another for 24-V systems. My first thought to improve this was to use a jumper to select between the two ranges and have both sets of resistors installed, just in case. I finally decided to have the required duplication in software where it would weigh less and take up less shelf space. The firmware is actually checking the various waypoints down the discharge curve and also exactly double those voltages. This makes the meter "autoranging" within an admittedly small set of ranges. It also makes it hard for the end user to order the wrong meter or to configure it incorrectly. Note that I didn't say "impossible." Those of you with customers of your own will understand what I mean.

I added an overvoltage condition test for inputs of over 30 V. The

That's it!

The final version of the firmware is contained in a single file of approximately 300 lines of C code and takes up 99.4% of the ATtiny13's 1-KB flash memory. The file and its corresponding object file in Intel HEX format are posted on the *Circuit Cellar* FTP site.

SHOW & TELL

Once I had a working prototype of my new meter, I just had to show it off to some fellow robot builders. Last summer, I brought the battery meter, along with some other blinky toys that I had been working on, to an R2 Build Day hosted by Jerry Chevalier of the R2-D2 Builders Club. Several of the robot builders expressed interest in the meter and that got me thinking about making a real PCB and offering it for sale.

I had been playing with Advanced Circuits's PCB Artist software for several months but had not really

66

Several of the robot builders expressed interest in the meter and that got me thinking about making a real PCB and offering it for sale.

99

LEDs light up in an alternating red wig-wag pattern that screams out "uh oh, uh oh" to let you know you've hooked up the meter to the wrong wires.

One additional "waypoint" is coded into the firmware to allow quick and easy calibration of the meter. Then exactly 7.50 V is detected, which is below the normal operating range but still high enough to keep the voltage regulator happy, a special pattern of two yellow LEDs is lit on the meter. To calibrate, attach an accurate 7.50-V source (I happen to have one) and adjust the potentiometer R9 until the LEDs light up with the right pattern.

gotten the hang of it yet. I would tell myself to start out simply and work my way through a complete, if trivial, example. It never failed that I would eventually start trying to get all fancy and do a lot of things at once and would inevitably work myself into some indescribable corner where I would get frustrated and just end up walking away. It's ironic because Advanced Circuits's Drew Peterson lives to help people with PCB Artist questions. The trick is to be able to *describe* the problem you're having. If you can describe it, Drew can help you out. If you can't, other than "It is not doing what I want it to do," then you're kinda stuck.

So I started over again, this time on a schematic-only design. I had built a small, five-LED "Cylon" scanner and needed a schematic to put up on my web page. It just needed to be lines, dots, and text. I had already laid out a PCB without the benefit of a schematic (because it was that trivial). This proved a good exercise in learning the user interface of PCB Artist.

The next step in PCB-fu was to learn to make my own symbols. In PCB Artist, all components are symbols, with a schematic view, a PCB footprint, and a link associating the two. Once you can build your own symbols effectively, the sky's the limit. I'm still a long way from being an expert, but I can rough out a board in 1 hour or so and have production-quality work in a few hours more. As I mentioned before, the software also enables you to order PCBs online. I am constantly amazed by this wonderful world of ours.

The PCBs arrived the same day as the last of the components, so I got busy soldering. I used through-hole parts exclusively on this design and the boards were straightforward to assemble. The finished PCB is 1.500" x 1.500" (see [Photo 2](#)). The components are mounted on one side of the board, and the LEDs are mounted on the other side so that the meter can be flush-mounted in an enclosure, if desired.

The meters are available for sale on my web site (www.dalewheat.com). They are also available on the sites of the other vendors.

If you have any questions about this little circuit, please don't hesitate to contact me. ☒

Dale Wheat is a full-time freelance writer working primarily with embedded systems and shiny things that blink or beep. Dale is married and the father of two adult children. He lives near Dallas, where he enjoys mowing two acres of grass in the summer and not mowing it in the winter. To find out what he has been up to, visit his personal web site dalewheat.com.

PROJECT FILES

To download code, go to [ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/226](http://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/226).

RESOURCES

I. Buchmann, "Charging the lead-acid battery," Battery University, www.batteryuniversity.com/partone-13.htm.

The Electropaedia, "State of Charge (SOC) Determination," www.mpoweruk.com/soc.htm.

D. Lancaster, "Tech Musings," www.tinaja.com/glib/muse152.pdf.

Maxim Integrated Products, Inc., "Application Note 1798: Frequency Under-sampling in Coulomb-Counting: Measuring Current Flow in Battery Applications," 2002, www.maxim-ic.com/appnotes.cfm/appnote_number/1798.

———, "Application Note 1880: Charlieplexing—Reduced Pin-Count LED Display Multiplexing," 2003, www.maxim-ic.com/appnotes.cfm/appnote_number/1880.

R2-D2 Builders Club, <http://astromech.net>.

D. Wheat, <http://dalewheat.com>.

SOURCES

PCB Artist software

Advanced Circuits | www.4pcb.com

ATTiny13 and STK-500

Atmel Corp. | www.atmel.com

Windows CE Touch Controller

CUWIN™



CUWIN4300K ▶

\$499 / Qty. 1

- 7" / 10.2" wide color TFT display
- 800 x 480 resolution, 260K colors
- SD card & Ethernet support
- RS232 x 2 / RS485 x 1 or RS232 x 3
- Mono Speaker and Stereo Jack
- Real time clock (Battery backup)
- Visual Basic and Visual C++ support
- USB I/F (ActiveSync)
- ARM9 32bit 266MHz processor
- Keyboard or Mouse support

CUWIN3200 ▶

\$399 / Qty. 1

CUWIN3200

7"

CUWIN3500

7"

CUWIN4300K

10.2"

CUWIN4300S

10.2"

* CUWIN3200 : 7" Bezel type case

* CUWIN3500 : 7" Cover Case (Front waterproof)

* CUWIN4300K : 10.2" Black Bezel type case

* CUWIN4300S : 10.2" Silver Bezel type case

COMFILE TECHNOLOGY

www.cubloc.com

Toll-Free: 1.888.928.2562

Construct a USB GPIO Pod (Part 2)

USB JTAG Module

In the first part of this article series, DJ presented a general-purpose input/output (GPIO) pod that can plug into a USB port. Now he describes how to download a JTAG programming application and program a CPLD circuit.

As I explained in the first part of this article series, you can use a general-purpose input/output (GPIO) module that plugs into a USB port on your computer to make up for the lack of a parallel port. While the flexibility of having a microcontroller in this pod enables you to do many things, the real advantage is that you can put a lot of logic—even entire applications—in the microcontroller. This month, I'll explain how to download an entire JTAG programming application into the module. I'll then describe how to use it to program a simple CPLD circuit from a Verilog program from my Linux workstation.

There are commercial devices that let you program JTAG devices over USB from Linux. Then again, there are devices that do a lot of the things you learn to do in *Circuit*

Cellar, but it's no fun if you don't do it yourself, right? The module I built is a pure do-it-yourself solution. You can use it for more than just this one task, which saves money in the long run, too.

JTAG OVERVIEW

The Joint Test Action Group (JTAG) is a specification created for testing densely populated circuit boards. In addition to testing, many chips have extended their JTAG interface to include other operations, such as programming or debugging. The JTAG interface consists of five logic signals: Reset (optional), Clock, Mode Select, Data In, and Data Out. The signals are abbreviated TRST, TCK, TMS, TDI, and TDO. There are three types of operations, which,

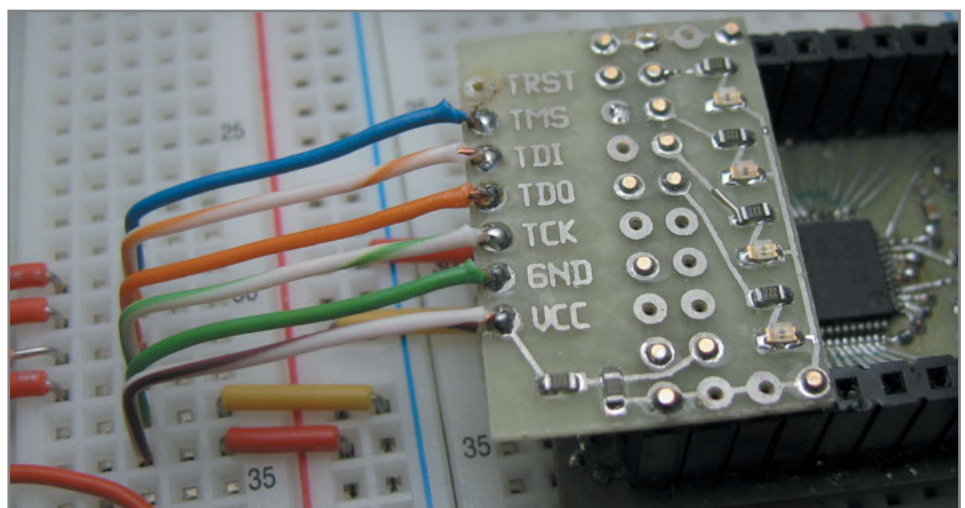


Photo 1—A closeup of the JTAG adapter. Most of the complexity is in the software; the adapter serves to manage the wiring itself. My project doesn't use TRST, so it is left unconnected.

in combination, yield all the operations the chip defines.

Not all chips include a TRST pin because it isn't always needed. When it is included, you can reset the chip's JTAG module by pulsing TRST low. When it is not present, the JTAG module can be reset by clocking in a reset command to the TMS port. Note that this doesn't reset the chip, just the JTAG interface.

The TMS pin controls a state machine, which selects the mode in which the interface operates. Chips that support JTAG use the same state machine, so a project with multiple JTAG chips can tie all the JTAG interfaces together—TRST, TCK, and TMS (in parallel) and TDI and TDO (chained in series)—and control all the chips at once. The state machine is designed so that a reset sequence of clocking five 1 bits on TMS always returns it to the idle state, regardless of where it started. Other combinations of clocked 1 and 0 bits maneuver the state machine into other states, where it can stay as long as the TMS

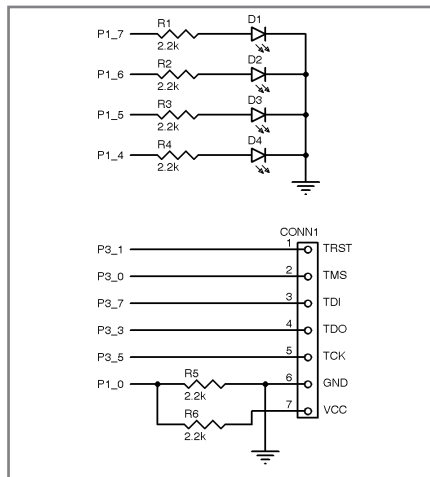


Figure 1—The JTAG adapter module is simple. R5 and R6 feed half the target VCC to P1.0, which is an ADC input on the pod. The rest is just blinky lights and connections.

bit remains low. Two of these states are key. One lets you shift bits into the instruction register via TDI (and shift its previous value out via TDO). The other lets you shift bits through the data register. In most chips, the instruction register selects which data

registers (out of many) are used and allows a wide range of functionality.

After a register is selected (either the instruction register or one of the data registers), you can access it through the TDI and TDO pins. The state machine first loads the register with data from elsewhere on the chip. Then, much like a SPI port, the old data is clocked out of TDO while new data is clocked into TDI. Once this is done, the state machine stores the register's data elsewhere in the chip, performing whichever action is indicated.

Aside from a few well-defined common functions, each chip has its own set of registers and functions that it documents. However, they all follow the standard JTAG interface for accessing those registers and functions. Chip manufacturers will often provide a configuration file that describes these functions. Software that uses JTAG to talk to these chips will often read these configuration files to understand how to access these functions. When multiple devices are in the JTAG chain (TDO

Expand Your I/O with Rabbit® RIO

Versatile Device with I/O Options

- Add functionality without costly processor changes
- Multiple Processor Interfaces
- Add 38 I/O
- Configure I/O for PWM, TRIAC, input capture, or decoder



**Rabbit RIO™ Programmable I/O
Application Kit for \$299**



**Order Online At
rsappkits.com**

08033



AP CIRCUITS

PCB Fabrication Since 1984

As low as...

\$9.95

each!

Two Boards
Two Layers
Two Masks
One Legend

Unmasked boards ship next day!

www.apcircuits.com







chained to TDI), programming utilities can use the configuration for the chips they're not programming to learn how to bypass those chips and access the one they're interested in.

Do you really need to know all this to use JTAG? Not really. Aside from telling inputs from outputs, you don't even need to know this to port the programming application to the pod. However, it helps you understand that there is a lot of bit twiddling going on.

JTAG POD MODULE

To simplify connections to the project, I created a JTAG programming module for the USB-GPIO pod (see [Photo 1](#)). Aside from bringing out the specific JTAG signals I needed, there are two other functions this module provides.

First, I added a couple of LEDs to provide some visual feedback of what the JTAG protocol is doing. After all, I can't have a project without blinky lights, and the LED on the pod itself is used to give feedback about the hardware flow control used by the serial port software. Second, I needed to be able to monitor the target circuit's supply voltage (see [Figure 1](#)).

Because the pod can run at either 3.3 or 5 V, and the target circuit can run at any voltage, I used one of the pod's ADCs to compare the target's voltage with the pod's voltage. The reference voltage is set by the pod to its own supply voltage. I used a pair of resistors to divide the target's supply voltage in half and feed it into the ADC. If the voltages match, the resulting ADC value should be about 128 (on a scale of 0–255, or an 8-bit conversion). If the pod is running at 3.3 V and the target at 5 V, the result will be 5/3.3 times higher, or around 194. If the voltages are the other way, the result will be 3.3/5 times as high, or around 84. If the target is not yet powered up, the result will be zero. While the R8C/20 does not have 5-V-tolerant inputs when running at 3.3 V, this at least enables me to find out quickly if the voltages are mismatched. Mostly, it just checks to see if the target is powered up at all, protecting the target from damage.

The pod's software is a port of the JTAG programming application, as provided in Xilinx's 2007 application

Listing 1—This is the top module for my CPLD project. The signals named on the line with module top correspond to pins. Each always block corresponds to some logic gates, and other logic modules are brought in by reference.

```
module top(ibin, nen, en, blank, lzblank, polarity, oseg2, oseg1, oseg0);
    input [7:0] ibin;
    input nen;
    input en;
    input blank;
    input lzblank;
    input polarity;
    output [6:0] oseg2;
    output [6:0] oseg1;
    output [6:0] oseg0;

    wire nen;
    wire en;
    wire blank;
    wire lzblank;
    wire polarity;

    wire [1:0] bcd2;
    wire [3:0] bcd1;
    wire [3:0] bcd0;

    reg [7:0] ibinh;

    reg blank2;
    reg blank1;
    reg blank0;
    reg lz2;
    reg lz1;

    wire [6:0] oseg2t;
    wire [6:0] oseg1t;
    wire [6:0] oseg0t;

    // Latch in the input value
    always @ (ibin, nen, en)
    begin
        if (en & ~ nen)
            ibinh <= ibin;
        else
            ibinh <= ibinh;
    end

    // Convert it to BCD
    bcd b0 (ibinh, bcd2, bcd1, bcd0);

    // Calculate leading zero suppression
    always @ (bcd2, bcd1)
    begin
        lz2 <= (bcd2 == 0) ? 1 : 0;
        lz1 <= (bcd1 == 0) ? lz2 : 0;
    end

    // Calculate blanking
    always @ (lz2, lz1, lzblank, blank)
    begin
        blank2 <= (lzblank & lz2) | blank;
        blank1 <= (lzblank & lz1) | blank;
        blank0 <= blank;
    end

    // Convert to seven segment format
    sevenseg012 s2 (bcd2, oseg2t);
    sevenseg s1 (bcd1, oseg1t);
    sevenseg s0 (bcd0, oseg0t);

    // Apply blanking and polarity, and output
    blanker digit2 (oseg2t, polarity, blank2, oseg2);
    blanker digit1 (oseg1t, polarity, blank1, oseg1);
    blanker digit0 (oseg0t, polarity, blank0, oseg0);

endmodule // top
```

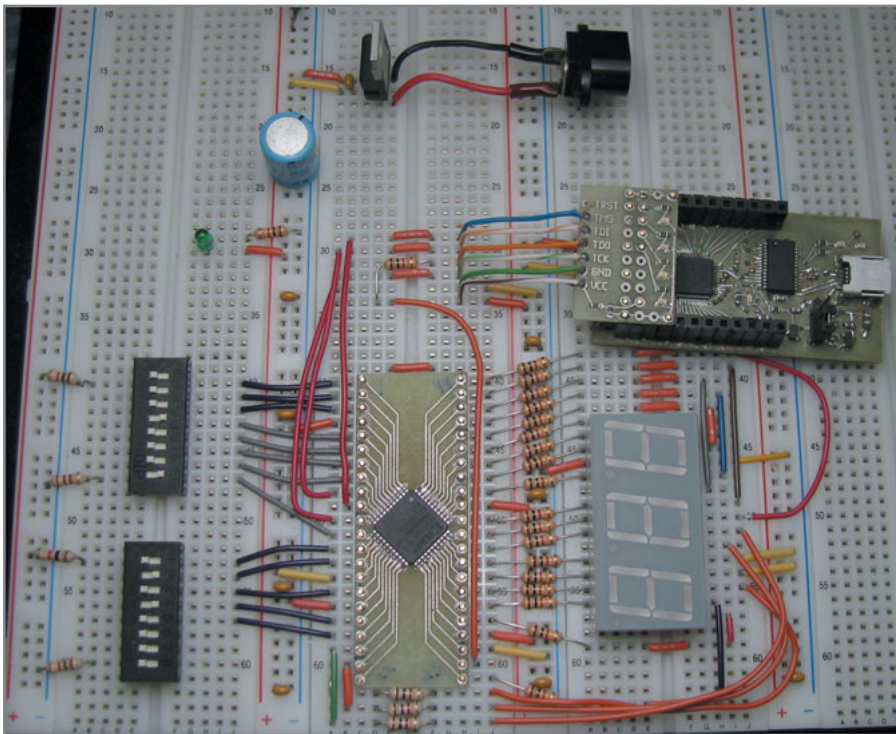


Photo 2—My CPLD prototype. At the top is a 3.3-V power supply. On the left are two DIP switches to provide manual input to the CPLD, shown center-mounted on a DIP adapter. At the right is the LED display, and above that is the pod with the JTAG adapter module. Having programmable pins on the CPLD means wiring up the circuit the easiest way, which can be seen here as short, neat connections.

note titled “Xilinx In-System Programming Using an Embedded Microcontroller” (XAPP058). I used the core files as-is, but wrote my own version of ports.h and ports.c to correspond to my pod’s hardware. I had to provide two interfaces.

First, I had to provide a source for the CPLD data. This data was created by the Xilinx WebPack utility, which produces (among other things) a Xilinx Serial Vector Format (XSVF) file. The file is a set of binary commands describing how to program the CPLD. The host application feeds this data across the USB link, so the port on the pod has a serial UART driver that reads this data.

The second interface is the I/O for the JTAG signals. I simply mapped these to GPIO signals on port 3. Conveniently, I’d used the SPI pins to allow for future use of the SPI peripherals on the R8C to drive the JTAG interface even faster.

At first, I ported XAPP058 to run on the host, using a simple serial protocol to toggle the output bits and read the



We add value to PCBs when others just sell it.

- One Stop Manufacturing Service
- Free Electronics Components
- Free Prototyping Assembly
- Professional Consultant

Design

Prototype

Production


Designing Service

3D Enclosure Designing Virtual Assembly PCB Design



PCB



Assembly



Quick Prototype



Component



FPC



Keypad



Components FPCs Keypads



www.EzPCB.com **Email: sales@ezpcb.com**

input bits. It was slow, clocking at only a few kilohertz (one serial byte per logic pin change). By running the application on the pod and sending only the XSVF data across the USB link, I avoided the overhead. The serial port was configured to support full hardware flow control, so no special host application was required. I could use any terminal emulator with a “send raw file” option. However, I wrote a small utility to automate the communication. It resets the pod, which gives control to the pod’s initialization function. That let me check that the power supplies match. Once they matched, I sent a command to turn control over to the XSVF application, and started sending the .xsvf file across. When the programming was done, control returned to my pod function and it sent the completion status to the host utility.

How much of a difference does it make? Running the application on the host and sending each change output or read input over the USB interface turns out to be expensive. Programming the

Listing 2—This is a sample session of building and downloading a pod application.

```
$ cd pod/xsvf ; make

make: Nothing to be done for `all'.
$ sudo ../../host/uflash/uflash xsvf-pod.elf
version = "VER.1.00"
Status: Seq: Ready   Erase: Normal   Program: Normal   ID: ID'd
Status: Seq: Ready   Erase: Normal   Program: Normal   ID: ID'd
00ff00
done!
```

XC9500XL, which has only a 28-KB bit file, takes almost 8 minutes! Varying the pod’s CPU clock, or the data rate between the FT232R and the R8C, has almost no effect on this time. It’s all USB packet overhead for all those bit changes. Moving the application to the pod’s CPU makes the bit twiddling fast, especially if I inline the pod-specific instructions, which saves even the function call overhead. By avoiding the overhead, the programming time is reduced to 12 s, about 40 times faster.

SAMPLE CPLD CIRCUIT

I got the idea for the circuit from a

Usenet posting. The original poster was asking for the best way to turn an 8-bit binary number (0–255) into three seven-segment digits (000 through 256). Although there were other suggestions involving ROMs or special TTL chips, I suggested an inexpensive CPLD. At that point, it occurred to me that such a project would be a useful way to teach myself about CPLDs and Verilog. The project was well defined—eight binary inputs representing a number from 0 to 255, and 21 logic outputs to directly drive three seven-segment LED displays. Other pins would be used for other features if I had space in the chip.

WIRELESS MADE SIMPLE®

BRING YOUR PRODUCT QUICKLY AND LEGALLY TO MARKET

RF Modules

Add **INSTANT** wireless analog / digital capability to your product.

Low-Cost TX, RX & TRX Modules

Long-Range Modules

Multi-Channel Modules

OEM Products

FCC PRE-CERTIFIED & ready to customize for your application.

Handheld TXs

Keyfob TXs

Function Modules

Feature Products

A closer look at Linx innovation

LOW-COST • LONG-RANGE TRANSCIVER


- Direct serial interface
- Low power consumption
- PLL-synthesized architecture
- RSSI and power-down functions
- Compact surface-mount package
- No external RF components (except antenna)

REMOTE CONTROL TRANSCODER IC

- Up to 8 inputs
- Bi-directional control
- Transmitter ID output
- Automatic confirmation
- Secure 2²⁴ possible addresses
- Latched and/or momentary outputs



800-736-6677
159 Ort Lane • Merlin, OR 97532
www.linxtechnologies.com



CIRCUIT CELLAR®

FOR COMPUTER APPLICATIONS

Make sure you're signed up to receive Circuit Cellar's monthly electronic newsletter. *News Notes* will keep you up to date on Circuit Cellar happenings. Stay in the loop!

Register now. It's fast. It's free.
www.circuitcellar.com/newsletter/

After some searching, I chose the Xilinx XC9500 family of CPLD chips. They were available with a variety of voltage requirements, packages, and logic cell counts. Also, they were inexpensive. The smallest member of the family was only \$1 per chip and available online. It turned out that the smallest chip wasn't big enough for this project, so I chose the second smallest—the XC9572XL. It is a 3.3-V part with 72 logic cells that's available in a 44-pin TQFP. While the definition of logic cell varies from manufacturer to manufacturer, in general each cell includes some form of flip flop and a wide range of combinatorial logic (AND and OR gates, multiplexers, and so on). The program you download into the chip decides which gates and other signals are connected to each other. Another advantage of this family is that the program is stored in the chip itself, so no other supporting logic is required once the chip is programmed.

I chose Verilog to describe the circuit I wanted, again, as a way to learn Verilog. Alternatives to Verilog are VHDL and schematics, but Verilog was simpler for this project than VHDL and more powerful than schematics. The full Verilog sources for this project are posted on the *Circuit Cellar* FTP site. There are three source files, which correspond to two independent modules and the logic that ties them together. Each source file defines one or more logic blocks in the form of modules. Like software functions, each logic block has a set of parameters that represent its inputs and outputs. Unlike software functions, these inputs and outputs are electrical connections, representing wires. To use one logic block in another module, you instantiate a copy of the block in your own logic. When you do this, you give each instance its own name and specify which signals in your module will connect to which inputs and outputs in that instance. Think of modules as a definition of a chip, and instantiating one as buying a chip to use.

First, I needed a way of decoding the binary number into a three-digit decimal number. In *bcd.v* there is a large


case table, with one entry for each input combination. Each case describes the three output digits. I actually used a script to generate this, rather than type it all in by hand, but at least I didn't need to try to figure out the logic behind each output bit—the Xilinx tools did it all for me.

Next, I needed a way to map the binary encoding of a digit into a seven-segment encoding for that digit. There are two of these in *sevenseg.v*, as the hundreds digit has only three

possible values. Adding a smaller table for that digit reduces the amount of internal logic needed. Again, I used a case table to describe each combination, with default entries (not all binary inputs are valid) of "don't care" to help reduce the logic needed.

The last module is the top-level module for this chip, unimaginatively named *top.v* (see [Listing 1](#)). The other two modules had inputs and outputs, but they existed only for referencing from other modules. For the top-level

"Producing prototypes on-the-fly allows me to be more Creative"



Electrical engineers agree: with a Protomat S-Series prototyping machine at your side, you'll arrive at the best solutions, fast. These highly accurate benchtop PCB milling machines eliminate bread-boarding and allow you to create real, repeatable test circuits—including plated vias—in minutes, not days.

- Declare your independence from board houses
- Affordable, entry-level price tag
- The best milling speed, resolution, and accuracy in the industry
- Single-sided, double-sided, and multilayered machining without hazardous chemicals
- Optional vacuum table and autosearch camera for layer alignment



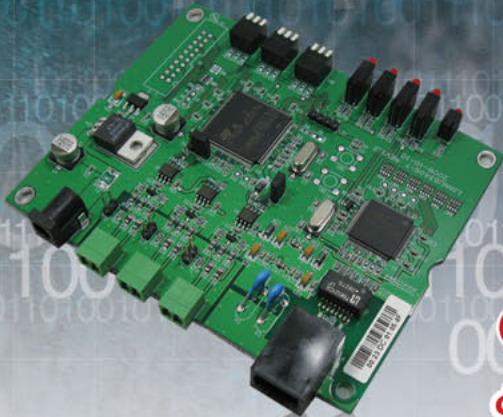
**ProtoMat® S-Series
PCB Milling Machines**

LPKF®
Laser & Electronics

For complete details visit:
www.lpkfusa.com
or call:
1-800-345-LPKF

3 PORT INTERFACE

RS-485 to Ethernet Converter



Only
\$170



RS-485 to Ethernet Converter

Powerful feature

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

Specification

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

MP3P DIY KIT, Do it yourself

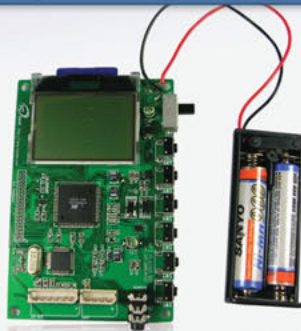
(Include Firmware Full source Code, Schematic)

• myPIC



Only
\$160 \$220
qty 100 qty 1

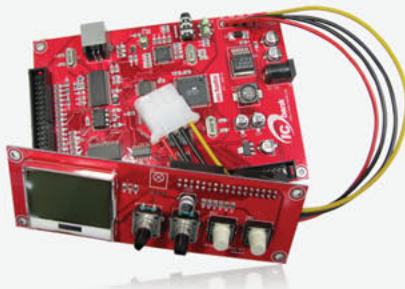
• myWave (MP3 DIY KIT SD card Interface)



Only
\$150
qty 100
\$200
qty 1

• myAudio (MP3 DIY KIT IDE)

Only
\$180
qty 100
\$220
qty 1



Powerful feature

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

Specification

Microchip dsPIC33FJ256GP710 / 16-bit, 40MIPS DSC
VLSI Solution VS1033 MP3 CODEC
NXP UDA1330 Stereo Audio DAC
Texas Instrument TPA6110A2 Headphone Amp(150mW)
320x240 TFT LCD
Touch screen
SD/SDHC/MMC Card
External extension port (UART, SPI, I2C, I2S)

Powerful feature

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

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

Powerful feature

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

module, the inputs and outputs are the pins on the chip. There are eight inputs for the binary number, and 21 outputs for the LED segments. There were enough pins and logic cells left over to add a pair of latch enable pins, two types of blanking (display and leading zero), and an output polarity control pin.

While Verilog looks like generic software programming code, it's describing hardware. You can follow the interconnections in `top.v`, as I coded the logic blocks from the input pins through to the output pins. First, I latched the input pins into a register called `ibinh` (input binary, held). I connected this register to the BCD decoder `b0`, returning the three digits as `bcd2`, `bcd1`, and `bcd0`. Next, I added logic to detect when leading zeros were present (`lz2`, `lz1`—the third digit is always on) and to compute the blanking signals `blank2` through `blank0`.

I also connected (note that I write "connected" instead of "passed"—I'm talking about wires here) the three

BCD values to three instances of the seven-segment decoder. I used two of the full decoders and the one partial one for the hundreds digit. The results of these decoders each have to pass through modules to determine if they're inverted, noninverted, or tristated (blanked).

The results of these last modules are connected to the chip's output pins. Programmable logic uses a constraint file to determine which internal signals go to which pins. In this case, `bin7.ucf` maps all the pins to their signals. But if your layout is flexible, you can let the tools choose which pins get which signals based on the internal layout of the design and how it relates to the available pins. Some pins, for example, are connected to the chip's internal clock grid, and the I/O pins each have an associated logic cell block that they prefer to be connected to. As long as the chip permits it, you can even fix board layout problems by going back to the chip and moving or redefining pins. An added bonus is that you can migrate the design to a different

package just by writing a new constraint file and rebuilding.

PROGRAMMING DEMO


Now that I've covered all the parts, let's focus on how they fit together. I built the CPLD circuit ("the circuit") on some protoboard (see [Photo 2](#)). I plugged the JTAG adapter module into the USB pod and wired it into the protoboard, matching TDI to TDI, TCK to TCK, and so on. I labeled mine to match the pin names, but keep in mind that if you have to chain two or more devices together, you connect TDO to TDI, with the pod's TDI connected to the first device in the chain, and its TDO to the last.

While the JTAG module connects to the circuit's V_{DD} bus, it neither powers the circuit nor draws power from it. The V_{DD} pin on the module only measures the V_{DD} through a resistor divider and an ADC. Our CPLD does not use the TRST pin, so I have not connected it to anything.

The R8C tristates all I/O pins at reset, so it's relatively safe to connect

Cellular and GPS capable

Data Mover



- Flash file System
- 4 Chan 12-Bit A/D
- 1MB SRAM
- 512KB FLASH
- 4 Isolated Inputs
- 4 Hi Current Outputs
- 2 External RS-232
- 2 Internal RS-232
- Bat Backed Clk/Cal
- Cell Modem Option
- Internal GPS Option
- Metal Case Option
- 1ma Standby Option

It's easy and cost-effective to do mobile or solar-powered data collection and asset monitoring with the JK micro's **Data Mover**.

With the ability to integrate a Cellular Modem, GPS and DOS-based embedded controller in a single rugged enclosure, you can capture and transmit your data quickly, easily and at low cost. Inexpensive development kits including Borland C/C++ and PowerBasic are available now. Call or email us for more details.

Call **530-297-6073** Email sales@jkmicro.com
On the web at www.jkmicro.com

JK microsystems


FlashPro430

FlashPro-CC

FlashPro2000

GangPro430


GangPro-CC




USB Flash Programmers for Texas Instruments' MCUs
MSP430, Chipcon CCxx, C2000 DSPs

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

- * can assign unique serial number
- * up to eight programmers can be connected to one PC and program target devices simultaneously





www.elprotrotronic.com

USBee Test Pods

Programmable Multifunction Logic Analyzers,
Oscilloscopes, Signal Generators
and Protocol Analyzers

Inspired by engineers

Feared by bugs



Actual Size

USBee Suite

For the USBee SX, AX,
ZX and DX



USBee Test Pods
starting at \$139

www.USBee.com
Powerful Debugging - Small and Portable

USBeeTM
USB based Electrical Engineer

the pod to the circuit before you program the pod. If you haven't done so already, build the JTAG application and program it into the pod. This only has to be done once, unless you put some other application into the pod (see Listing 2). If you haven't built the host-side application, build that too. This builds a small host-side program called `xsvf` that does nothing but send the CPLD bitstream to the pod:

```
$ cd ../../host/xsvf ; make
```

Many of the pod applications will be built this way—some software that runs on the pod, and some software that runs on the host. As this example shows, minimizing the amount of data that needs to go over the USB bus can yield amazing performance gains.

To build the CPLD image, you'll need to download and install the Xilinx WebPACK application. This includes everything you'll need to program CPLDs. I used the GUI to do the initial design, and then wrote a Makefile to automate it. There are additional

instructions on the FTP site; but in a nutshell, compile the Verilog into a standard intermediate form, compile that file to the specific chip, and then convert that file to the format needed for programming. In my case, I used the XSVF format. SVF is a standard vector format, but it's a text file. XSVF is a binary format that's specifically designed to be easily interpreted by a small application. That small application is the one I put in my pod. Building the XSVF file is simple with makefiles:

```
$ cd ../../cpld ; make
```

This created a bitstream file `bin7.xsvf` with binary-encoded instructions on how to program the CPLD to do what I wanted it to do. After all the parts were built and the pod was programmed, I could use it to program the CPLD:

```
$ sudo ../host/xsvf/xsvf bin7.xsvf
```

What exactly happened here? Well,

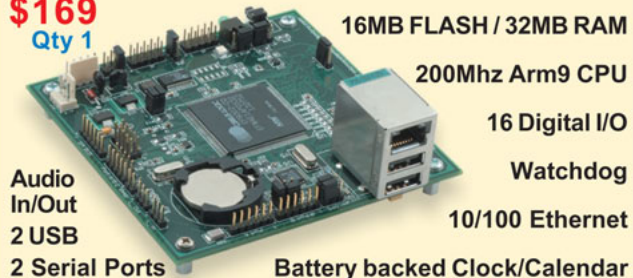
the `sudo` command provided access to the USB hardware, needed by the FTDI libraries. The `xsvf` host program used the FTDI libraries to open a channel to the pod and take it out of reset, which started the pod's program. The host sent a few commands to tell the pod to check the circuit's V_{DD} and verify that it matched the pod's V_{DD} . The host then read the XSVF bistream file and sent it across the USB line to the pod, which interpreted those instructions so as to use the JTAG lines to program the CPLD. Note that the pod and host used hardware flow control to control the datastream, and the pod had a 256-byte buffer to hold incoming data.

Once the programming was done, the host program read any diagnostic text from the pod and printed it, and the pod released the CPLD. The CPLD, of course, started running the logic design programmed into it. Once programmed, I could disconnect the pod from the circuit. Done!

The first time you do all this, there's a lot of set up. However, edits

Easy Embedded Linux

\$169
Qty 1



Audio
In/Out
2 USB
2 Serial Ports

16MB FLASH / 32MB RAM

200Mhz Arm9 CPU

16 Digital I/O

Watchdog

10/100 Ethernet

Battery backed Clock/Calendar

We brought you the world's easiest to use DOS controllers and now we've done it again with Linux. The **OmniFlash** controller comes preloaded with Linux and our development kit includes all the tools you need to get your project up and running fast.

Out-of-the-box kernel support for USB mass storage and 802.11b wireless, along with a fully integrated Clock/Calendar puts the **OmniFlash** ahead of the competition.

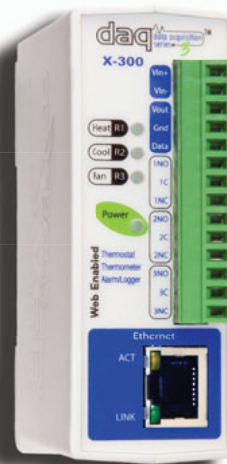
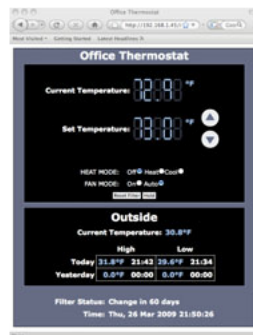
Call **530-297-6073** Email **sales@jkmicro.com**

On the web at **www.jkmicro.com**

JK microsystems

Introducing X-300

**WEB ENABLED
THERMOSTAT
TEMPERATURE LOGGER
THERMOMETER**



(435) 750-5999
sales@ControlByWeb.com

**CONTROL by
WEB™**
www.ControlByWeb.com

to the Verilog need only a make flash command to have the Makefile build a new bitstream and send it to the CPLD chip. The Makefile rebuilds whatever is needed and runs the host xsvf program to send it to the CPLD.

MORE USES

You now know how to use the CPU in the USB GPIO module to offload processing in order to optimize performance for a task-oriented purpose. More than just fiddling bits, it enables you to put intelligence close to your circuit in order to take advantage of the R8C's speed and peripherals. I also showed you how the host and pod programs work together to create smart peripheral modules, like the JTAG controller. And last, but not least, I described an example of how a smart pod can be used in a real-life circuit—programming a CPLD chip.

Can you think of another use for the pod? Experimenting with LCD interfaces, testing SPI chips, preprogramming PC EEPROMs—the possibilities are limited only by your needs! 📷

DJ Delorie (dj@delorie.com), who has been designing electronic circuits since high school, earned an ECE degree at Clarkson University. After holding jobs designing PC motherboards and network management software, he now writes embedded development tools for Red Hat. DJ is also the creator of DJGPP and one of the contributors to the gEDA project.

RESOURCES

Bin7 Project Page, www.delorie.com/electronics/bin2seven/.

Xilinx, Inc., XAPP058 Source Files, <ftp://ftp.xilinx.com/pub/applications/xapp/xapp058.zip>.

——, "Xilinx In-System Programming Using an Embedded Microcontroller," XAPP058, 2007, www.xilinx.com/support/documentation/application_notes/xapp058.pdf.

SOURCE

R8C Microcontroller

Renesas Technology Corp. | www.renesas.com/en/r8ctiny

WebPack Software and XC9500 family of CPLDs

Xilinx, Inc. | www.xilinx.com


2B2C
 Factory on Your Fingertip



From Idea to Reality

- > Design
- > Prototyping
- > Production

See Our Differences in

- > Quality
- > Service
- > Price

*Milling *Turning *Grinding *CNC
 *Wire Cutting *Laser *Plasma
 *Water Jet *Plastic Injection
 *Sheet Metal *Gear *SLA
 *FDM *SLS *LOM




CLICK TO MAKE

MachinePIER

Tel: 408-421-9840 Email: sales@machinepier.com
 Website: <http://www.machinepier.com>

Got Serial, Need Network?




Bluetooth
Qty 1
\$145

Ethernet
Qty 1
\$99

Wireless
Qty 1
\$199

Volume Discounts Available


gridconnect™
www.gridconnect.com
+1 800 975-4743

DOS in the 21st Century

A USB Flash Drive Reader for MCUs Works for DOS

With a little effort, you can turn DOS into a handy real-time operating system. As Andrew and Jon explain, it can be the perfect fit for embedded applications that may require too many resources for a single-board microcontroller. Read on to learn how this USB flash drive reader for DOS can enhance your future embedded applications.

Even the smallest embedded project can read and write USB flash memory drives thanks to the Vinculum VDRIVE2 flash memory drive reader module from Future Technology Devices International. The VDRIVE2 module, with its built-in USB socket, snaps into the front panel of your project and talks to a microprocessor through either a SPI port or a serial port (see [Photo 1](#)). All of the complexities of talking to a flash memory drive (attachment, FAT16 support, and so on) are handled by its FTDI VNC1L flash memory drive host controller chip, which provides simple commands for directory listings, file transfers, and other operations.

You can get a VDRIVE2 off-the-shelf from Mouser Electronics for about \$25. Although the VDRIVE2 was intended for microcontroller use, our embedded application uses DOS. Much of this article will apply to both.

WHY DOS?

DOS may be dead in the desktop computing world, but it lives on as an important operating system for embedded applications. One of the great advantages of DOS is that small tweaks can turn it into a true real-time operating system that is perfect for low-volume, cost-sensitive applications that require too many resources for a microcontroller. At the National Institute of Mental

Health (part of the NIH), Andrew uses an open-source data acquisition and control program called NIMH Cortex. It is a real-time DOS application developed for behavioral brain research. It supports low-speed analog channels (each with a 1 ksp/s conversion rate), lots of digital I/O, and a few specialized interfaces (e.g., touchscreens), and it comes with a companion program for near



Photo 1—The Vinculum VDRIVE2 USB flash memory drive reader snaps easily into a front panel. It has a built-in microprocessor that manages the flash memory file system with simple serial or SPI port commands, so just about any embedded controller can access a flash memory drive.

real-time display of simple graphics objects on a separate computer. You program NIMH Cortex in C. Once you get past the learning curve, it is a powerful tool for automating experiments. One thing we really miss on our DOS system, however, is a USB flash memory drive reader. USB disk support can run under DOS (refer to www.bootdisk.com/usb.htm for examples), but large applications like NIMH Cortex don't tolerate the terminate and stay resident (TSR) and other drivers that eat up scarce DOS resources. We have circumvented the DOS memory problem by using the VDRIVE2. In this article, we will describe our mixed hardware/software solution for a DOS flash memory drive reader/writer that can be assembled for less than \$50 and doesn't use resident memory.

HARDWARE

The VNC1L chip has a built-in UART, but the chip provides only 5-V logic signals. RS-232 serial ports require ± 5 to ± 15 V, so a level converter is needed for the VDRIVE2. A Maxim Integrated Products MAX232A is the most common chip for this job. It needs only four 0.1- μ F capacitors and a decoupling capacitor to provide buffering and bipolar voltage boost. We designed a small PCB using the free layout tools from ExpressPCB. Two copies of the circuit board layout fit onto a single 2.5" \times 3.8" board, the size used for the ExpressPCB MiniBoard Service. Using our layout file, you can electronically order three boards (six copies of the circuit) for about \$60. The layout file is posted on the *Circuit Cellar* FTP site. Alternatively, you can delete one copy of our circuit from the layout and lay out another project. Then your \$60 will get you three converter boards and three of your design. Just don't forget to leave space between the layouts for cutting each board. If you plan to use the VDRIVE2 with a SPI port, a level converter is not necessary. However, a level converter will enable you to use

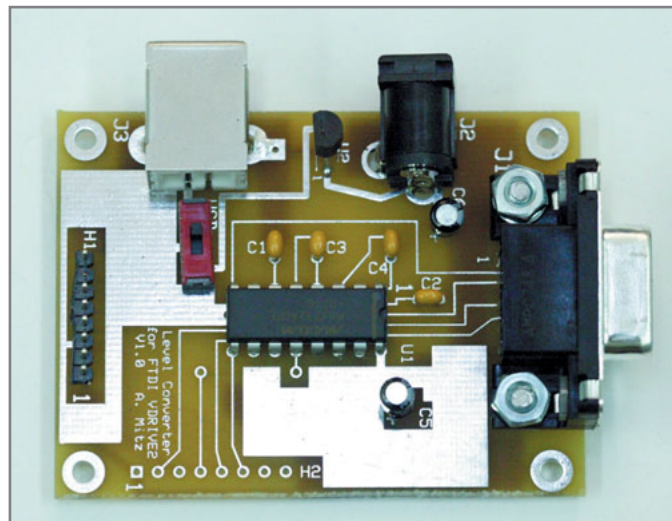


Photo 2—The VDRIVE2 has 5-V (TTL/CMOS) signals. It needs a level converter to interface directly with a serial port. Our level converter circuit board is shown here. It has a header for the VDRIVE2 on one end and a board-mounted serial port connector (DE-9) on the other end.

your computer as a test environment for learning about the VDRIVE2 and even for embedded code development.

Photo 2 shows the populated circuit board. The schematic is in **Figure 1**. **Table 1** has a complete description of the parts, including the parts for power and packaging. A nine-pin D-sub connector (J1, which is technically a DE-9, but often called a DB-9) mounts directly on one end of the PCB for connection to the computer serial port. The other end of the circuit board has a single-row 2-mm pitch header (H1). The header matches the jumper cable that comes with the VDRIVE2. For tight spaces, you can cut one connector off of the jumper cable and solder the cut wires directly to the circuit board. An alternative header socket (H2) is on the side of the circuit board; it has the same connections as H1, but

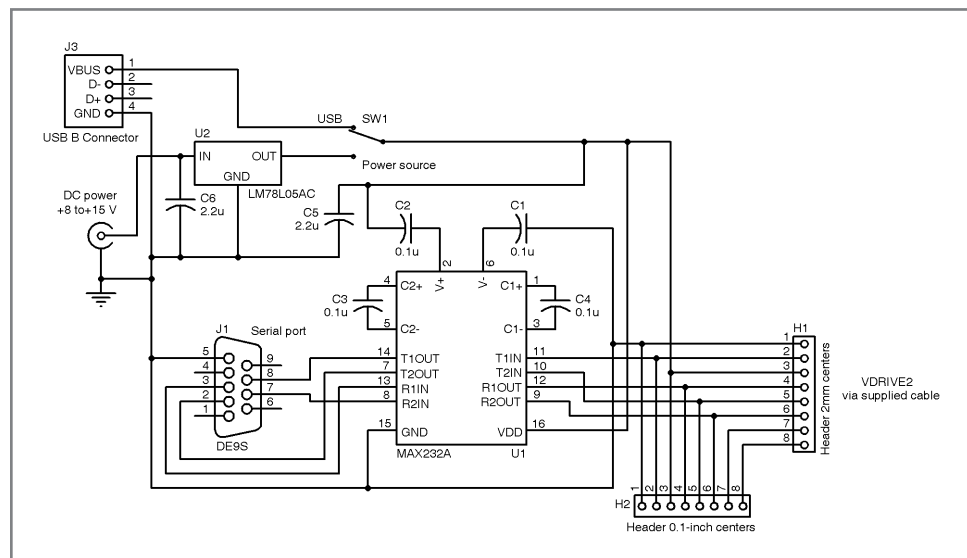


Figure 1—The level converter schematic shows two power options: wall transformer or USB port power. The circuit can be assembled for one or both power options.

with 0.1" spaced pins. H2 can be used for soldering the cut VDRIVE2 wires or to gain access to the VDRIVE2 signals for testing. Like the two connector alternatives, two alternatives are provided for powering the board, as well. The layout has holes for a circular connector (J2) commonly used for wall power transformers (bricks). Any DC brick rated for 9 to 15 V at 100 mA will work. A low-power, three-terminal regulator (LM78L05, U2) drops this voltage down to 5 V for the MAX232A and the VDRIVE2. Alternatively, regulated 5 V can be pilfered from any available USB port via J3. J3 is a type-B USB connector,

Quantity	Ref	Description	Source	Part Number	Notes
1		Vinculum VDRIVE2 USB flash memory drive interface	Mouser Electronics	895-VDRIVE2	
2	C1, C2, C3, C4	0.1-μF/50-V Ceramic capacitor	Digi-Key	399-4264-ND	
3	C5, C6	2.2-μF/50-V Miniature electrolytic capacitor	Digi-Key	P825-ND	
1	H1	Single-row 2-mm header, 50 pins	Digi-Key	SAM1176-50-ND	Break off eight pins for board
3	H2	Single-row 0.1" header			Do not install a connector
2	J1	Nine-position female D connector, PC mount, right angle	Digi-Key	A32075-ND	
1	J2	DC power connector, PC mount, 2.1 × 5.5 mm	Digi-Key	CP-202A-ND	Mount connector only if using a wall transformer
1	J3	USB B connector, PC mount	Digi-Key	AE9925-ND	Do not mount connector if only using a wall transformer
1	SW1	Ultra-miniature slide switch	Digi-Key	360-2133	Use jumper instead of switch if only one of J2 or J3 is used
1	U1	MAX232A RS232 Driver/receiver, 16-pin DIP	Digi-Key	MAX232ACPE+-ND	
1	U2	LM78L05AC Low-power 5-V linear regulator, TO-92	Digi-Key	MC78L05ACPFS	
2		Mounting screws for J1, 4-40 × 3/8	Digi-Key	H781-ND	Box of 100, only two needed
2		Nut 4-40	Digi-Key	H216-ND	Box of 100, only two needed
1		D-Sub hardware set	Digi-Key	609-1420-ND	
1		Printed circuit board	ExpressPCB		Order using ExpressPCB software and .PCB file
1		Wall transformer 9 VDC at 125 mA	Digi-Key	MT7141-ND	
1		Serpac series A plastic case, black	Digi-Key	SRA21B-ND	
1		End panel with DB9 cutout for Serpac A-21	Digi-Key	SR2005-DB9B-ND	

Table 1—This is a complete listing of parts and parts sources. A number of parts (J2, C6, U2, SW1, wall transformer) can be omitted when using the computer's USB source for power.

making it easy to bring power through a standard USB peripheral device cable. When using the USB port for power, J2, U2, and C6 are not needed. Switch SW1 is shown for those who might want both options on the same board, but more commonly a board will be assembled for only one power option with a wire jumper in place of S1.

You might want to mount the VDRIVE2 separately from the level converter board, but we chose to package them together in a plastic box (Serpac Series A) (see [Photo 3](#) and [Photo 4](#)). A wide rectangular slot is cut into one end plate of the box to accommodate the VDRIVE2. The other end plate is replaced with one precut for the DE-9 (Serpac A-21) (see Table 1). Standard DE-9 hardware secures the connector to the end plate and provides a threaded fastener for the connecting cable. The arrangement is a tight fit and some of the internal plastic ribs and stand-offs must be trimmed with a hobby knife, but the final product is appealing. The end plates and remaining stand-offs provide plenty of support for the circuit board and VDRIVE2. No additional mounting screws are needed. Cuts in the box's sloping sidewalls provide access to the power connectors.

Look at the VDRIVE2 datasheet before completing your assembly. The three-pin jumper on the back of the device (UART/SPI) should be set for pull-up. Once assembled, connect one of the power sources and connect your computer's serial port. You can use a modem program in either DOS or Windows (e.g., HyperTerminal) running at 9,600 bps to chat with the VDRIVE2. You can

even plug in a flash memory drive and get a directory listing with simple text commands. Refer to the Firmware manual for examples.

SOFTWARE CONSIDERATIONS

Our reason for using special hardware was to preserve DOS memory by avoiding resident drivers. Thus, from the outset, the plan was to produce a set of DOS commands for each aspect of talking to the flash memory drive (read, write, directory listing, and more). These primitive commands could be used on their own or serve as the backbone for an alternative user interface, perhaps a Norton Commander-style interface.

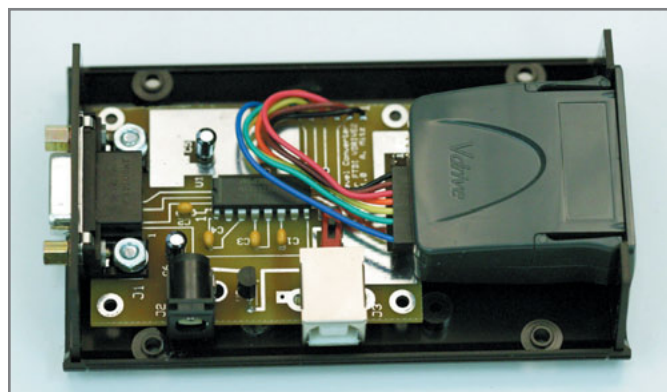


Photo 3—This shows how we packaged the VDRIVE2 together with the level converter circuit board. The fit is tight, so the VDRIVE2 cable is wired directly into the H2 pads rather than through a connector.

Saelig

UNIQUE PRODUCTS & SUPPORT

www.saelig.com



Testgear

Misc

CGR-101

www.saelig.com

RF Testing / EMI Tests

RFW320

PSoc Starter

UPSCAP / DLP-THT

Quantum

Color LCD Scope



2-ch + trigger standalone USB bench scope. **\$325 / \$599**

Handheld Scope



20MHz / 60MHz rugged handheld USB 2-ch scope. **\$593 / \$699**

Low-Cost Scope



2-ch 40/100/200MS/s 8-bit scope range with 5/10/25MHz. **\$297 +**

USB Bus Analyzers



Packet-Master™ - USB 1.1/2.0 analyzers and generators. **\$699 +**

Waveform Generator



USB2.0 speed 16-bit digital pattern or arbitrary waveform generator.

Wireless Data Loggers



Log and display temp, hum, volt, event-time or pulse-counting data.

USB to I2C



"Drop-in" solution connects PC to I2C/SMBUS + 32 I/O lines. **\$89**

Keyboard Simulator



USB board adds 55 I/O and 5 x 10-bit A/D inputs, 1 x 10-bit analog O/P.

.NET Board



Small (2.2" x 2.2") lowest cost .NET Micro Framework dev system.

2-ch 1GSa/s Scope



2-ch 1GSa/s (25GSa/s equiv.) 50/100 MHz scope. **\$595 / \$795**

Pen Scope



10/25MHz USB powered scope-in-a-probe! Up to 100MS/s. **\$193 / \$308**

1/2GHz RF Generators



High accuracy/stability, wide range, low phase noise/leakage, serial control.

16-Ch Logic Analyzer



Intuitive full-featured 16-ch 4MB 200MHz sampling memory. **\$299**

I2C Xpress



Versatile USB 2.0 I2C protocol exerciser and analyzer.

Multiparameter Loggers



Mini-logger with built-in temp/hum/pressure/3-axis accel sensors.

FTDI USB ICs



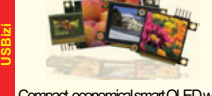
Popular UART and FIFO chips. Upgrade Legacy designs to USB.

Instant Ethernet



No OS needed. TCP/IP offload. ICs improve system performance.

Easy OLED Display



Compact, economical smart OLED with graphics drive from USB or RS232.

Scope + Analyzer



25MHz 2-ch / 16 logic scope and logic analyzer. **\$699**

6 in 1 Scope



200kHz 2-ch 10-bit scope, 2-ch spectrum analyzer, 16-ch 8MHz logic analyzer, 5-ch sig gen, 8-ch pattern gen. **\$199**

Mixed-Signal Scope



100MHz Scope, + Spectrum/Logic Analyzer and Signal Generator. **\$1259+**

SPI Bus Analyzer



Protocol exerciser/analyzer for standard SPI and non-standard 4-wire and 3-wire serial protocol interfaces up to 50 Mbps.

Automotive Testing



Kits turn your PC into vehicle-electrics diagnostic tool.

USB Logger



Standalone USB temp / hum / volt / current loop data logger. **\$49+**

CAN-USB



Intelligent CAN connection from PC's USB port. **\$299**

Ethernet-IO



UDP/IP-controlled 24 digital I/O board 3 x 8-bit TTL ports.

RF Modules

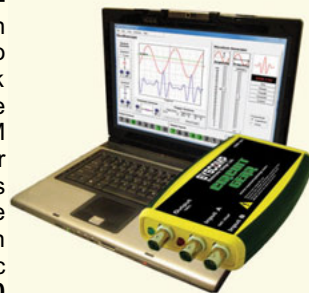


Simultaneously transmit composite video and stereo audio signals.

Amazing 7 in 1 Scope! \$180

CircuitGear CGR-101™ is a unique new, low-cost PC-based instrument which provides the features of seven devices in one USB-powered compact box: 2-ch 10-bit 20MSa/sec 2MHz oscilloscope, 2-ch spectrum-analyzer, 3 MHz 8-bit arbitrary-waveform/standard-function generator with 8 digital I/O lines. It also functions as a Network Analyzer, a Noise Generator and a PWM Output source – all for less than \$180! What's more – its' open-source software runs with Windows, Linux and Mac OS's!

Only \$180



EMC Spectrum Analyzer



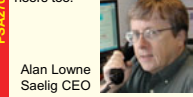
RF & EMF Spectrum Analyzer 1Hz to 7GHz for WiFi, mikes, etc.

EMC Spectrum Analyzer



Handheld Palm PC-based 2.7GHz Spectrum Analyzer.

"I really like this scope adapter - it's really meant for teaching electronic experiments but it's ideal for engineers too."



Alan Lowe Saelig CEO

CAN Gateway



Janz - Full-featured standalone fanless industrial Linux PC.

RF Generator



High-res, extremely low-noise, portable 3GHz RF generator.

RF Testing/EMI Tents



Portable RF test enclosures & shielding tents with external frame.

Electronic DC Load



Const. current, resistance, conductance, voltage & power modes

60/100/120MHz AWG



60/100/120MHz USB 14-bit ARB with USB RS-232, LAN/GPIB.

TorqSense



Configurable, patented USB-output non-contact SAW digital rotary torque transducers with integral electronics.

Serial-Ethernet Cable



Network serial product easily without a PC using this 28" cable. **\$89**

Lorlin Switches



Fantastic array of stock and custom switching devices.

PSoc Starter



Get going quickly with PSoc visual design environment.

FPGA Systems



Ready-to-go out-of-the-box FPGA/DSP designs for beginners and experts!

Wireless Solutions



Analog input, bluetooth wireless modules 433/868/915MHz.

Temp/RH Sensors



Novel ambient sensors & modules accurately measure temp/RH.

RS232 to 422/485



9p-9p or 25p-25p self-pwrd, isolated RS232-RS422/485

1/2/4/8/16 x RS232



Add 1-16 COMports via your PC's USB Port easily.

Quantum



Quickly add capacitive touch on / off & X / Y - sensing ICs.

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

Above are some of our best selling, unique, time saving products see our website for 100s more: WiFi/910MHz antennas, wireless boards, LCD display kits, Ethernet/IO, USB/RS232/485, USB OTG, instant Ethernet serial, CAN/LINbus, USB cables/extenders, line testers, logic analyzers, color sensors, motion controllers, eng. software, wireless boards, SMD adapters, I2C adapters, GPS loggers, automotive testing, security dongles, video motion detectors, crystals/oscillators, custom switches, barcode scanners, DSP filters, PLCs, Remote MP3 players, etc. FREE Starbucks card with your \$50 order! Check www.saelig.com often for special offers, bargains, business hints, blog, etc.

Industrial Non-Contact Distance Sensors



- Indoor or Outdoor
- Level, Dimension, Proximity
- 2 in. to 50 ft. multi-output
- PC Setup, networkable

Senix
Distance Measurement

800-677-3649
www.senix.com
cci@senix.com



Photo 4—This is the complete assembly, with the VDRIVE2 on one end and the level converter power connectors accessible from the side. The serial port connection (not visible) is opposite the VDRIVE2.

Celebrating Our 10th Year Anniversary

Get All Your Test Equipment In A Quick and Speedy Shipment!

20% Sale On Entire Selection Of Test Equipment!!

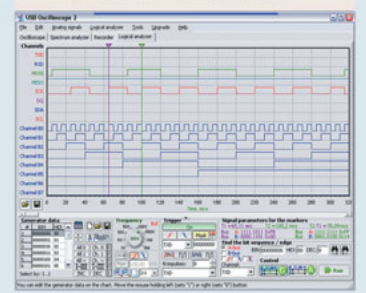
Oscilloscopes
Probes
Digital Multimeters
Function Generators
Voltage Testers
Capitance Meters
Analog Multimeters
Test Leads
& More...

DesignNotes.com
What Your Electronic Hobby Stores Used To Be

1-800-957-6867 www.DesignNotes.com

USB Oscilloscope for \$169.50

Logic and Spectrum Analyzers, Generator.
www.HobbyLab.us



A number of issues arose before and during development that were related to the problem of operating without a resident driver. A resident driver can remember the data rate and other serial port settings entered earlier by the user. Programs not loaded into memory need another way to avoid Korsakoff's syndrome. The two standard options are environmental variables and configuration files. We chose to use environmental variables. You can place SETENV commands in the AUTOEXEC.BAT file for boot-up configuration, but we recommend using our batch file that takes the parameters on the command line and checks for valid environmental variable names.

Environmental variables provide memory for the different programs using the VDRIVE2, but things get messy when trying to change the data rate of the VDRIVE2. If the VDRIVE2 has been set to one baud rate (e.g., 9600) and then you change the environmental variable to a new baud rate, the old baud rate information is lost. Multiple environmental variable sets could be employed—one for program tracking and one for user requests, but this approach was eschewed out of personal preference. Rather, we chose an algorithm that hunts for the proper data rate if communication with the VDRIVE2 is lost. The algorithm is useful for other occasions, such as when the

VDRIVE2 is first powered up.

Perhaps the most vexing problem associated with implementing read-write programs for the VDRIVE2 is dealing with the directory structure of the flash memory drive. The VDRIVE2 does not keep track of the current default directory. Without a resident program, it is difficult, albeit not impossible, to keep track of the current directory. Our programs require you to take responsibility for knowing the current default directory after one or more change directory commands. To complicate matters, the DRIVE2 supports only 8.3 (eight-character alphanumeric name followed by up to three-character alphanumeric extension) DOS file names. Thus, Windows XP directory names will often look like PROJEC~2. In our application, these limitations do not pose any real inconvenience. We use simple directory structures to move files from our NIMH Cortex data acquisition system to a Windows computer for editing or data analysis. With this in mind, we did not implement complex directory parsing at the command line. Specifications like ..\..\PJM\DAY2 are not supported, although specifications like ..\DAY2 and \PJM\DAY2 are.

The VDRIVE2 firmware must be at version V3.64 or later. Updating the VDRIVE2 can be done in two ways. One, you can download VPROG reflasher COM utility and

the latest VDAC ROM file from the Vinculum download page on the FTDI web site. The Reflasher COM utility runs from DOS or a DOS window and programs the VDRIVE2 via the serial port. Alternatively, you can rename the FTD file for the latest release to FTRFB.FTD, put the file on the root directory of a flash memory drive, and just plug the flash memory drive into the VDRIVE2. The VDRIVE2 will find the file and update its own firmware.

SOFTWARE IMPLEMENTATION

The eight DOS commands for using the VDRIVE2 are in Table 2. Two commands, FCONFIG and FBAUD, provide ways to establish the serial port communications parameters with the VDRIVE2. FCONFIG is the only command implemented as a DOS batch file (.BAT). The other commands are executable files (.EXE). FCONFIG sets environmental variables in a convenient way, providing some helpful error checking. Here are two example FCONFIG commands:

```
fconfig com 2 baud 115200
fconfig address 0x2E8 irq 5 baud 9600
```

The first example selects COM port 2 at 115,200 bps. The port and interrupt addresses are the standard ones for COM port 2. The second example explicitly sets (nonstandard) addresses. FCONFIG does not try to initiate communications with the VDRIVE2, FBAUD does. FBAUD initiates a search for the VDRIVE2 device, tries to establish a new data rate, then


reestablishes communications at the new rate. Allowed data rates are 2,400, 9,600, 19,200, 38,400, 57,600, and 115,200. A power-up reset places the VDRIVE2 at 9,600 bps.

FCD is used to change the default directory on the flash memory drive. When first inserted, the flash memory drive is set to the root directory. FCD can return the flash memory drive to its root directory using just a backslash as the command line parameter, just like the DOS CD command. Other operations are similar to the DOS CD command, with two exceptions, as noted above. FCD does not try to manage complex tree commands, and FCD will not report the current default directory path. FDIR also does not report the current directory path. FDIR with no command line parameters lists the entire contents of the current directory without any further details. FDIR with a specific file name will display the file size of that file. Simple DOS wild cards will work as expected (e.g., FDIR *.DAT), but more complex wild card constructs are not implemented. This limited directory functionality could be greatly expanded. We encourage you to think about writing a more convenient user interface using our source code as a starting point.

The file operation commands are FPUT, FGET, FDEL, and FREN. We thought about implementing an FCOPY command rather than FPUT and FGET, but having separate commands lets us use the file path parsing capabilities of DOS. Access to files on the DOS system can use any legal


DOS Command	Purpose
FCONFIG.BAT	Set COM port address, data rate, and IRQ in the environment.
FBAUD.EXE	Establish communications with VDRIVE2 at selected data rate.
FCD.EXE	Change default directory.
FDIR.EXE	List current default directory.
FPUT.EXE	Copy a file from the current DOS directory to the flash memory drive.
FGET.EXE	Copy a file from the flash memory drive to the current DOS directory.
FDEL.EXE	Delete a file on the flash memory drive.
FREN.EXE	Rename a file on the flash memory drive.

Table 2—This table lists the eight DOS commands for accessing files on the flash memory drive. FCONFIG is a batch file; the others are all exactly the same .EXE files with different names!



Pololu

Robotics and Electronics




NEW!

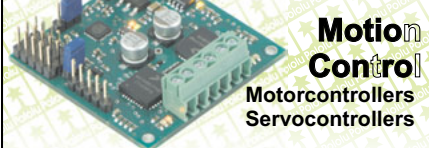
RobotKits
Linefollowers
Robotarms
Hexapods
Chassis

3piRobot
\$99.95

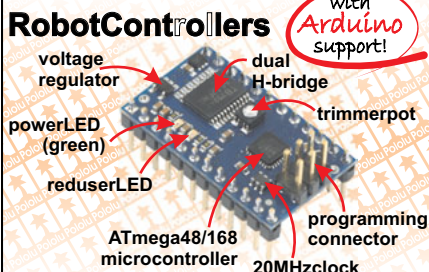
High performance, C programmable, ATmega168 based robot (with Arduino support)!



Mechanical Components
Motors, servos
Wheels, ballcasters

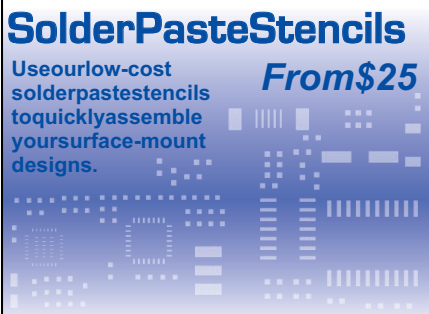


Motion Control
Motorcontrollers
Servocontrollers




RobotControllers *with Arduino support!*

voltage regulator
powerLED (green)
reduserLED
ATmega48/168 microcontroller
20MHz clock
programming connector
trimmerpot
dual H-bridge



SolderPasteStencils
Use our low-cost solder paste stencils to quickly assemble your surface-mount designs.

From \$25



CustomLaserCutting
From \$25

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

1-877-7-POLOLU
www.pololu.com
6000S Eastern Ave. 12D, Las Vegas, NV 89119

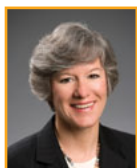
Standards Make Sense

Standards improve quality and enable designers to share components across different projects. Today, ARM® Cortex™-M profile processors, combined with the Cortex Microcontroller Software Interface Standard (CMSIS) and optimized middleware from the industry's largest ecosystem, are setting the hardware and software standards for microcontrollers.

These standards enable leading vendors such as Luminary Micro, NXP, and STMicroelectronics to supply advanced microcontrollers, while maximizing code reuse across multiple platforms.

Cortex-M3 Microcontrollers Make Sense

"We based our award-winning Stellaris® microcontrollers on Cortex-M3 to provide users with 32-bit performance while eliminating future architectural upgrades or software tool changes."



Jean Anne Booth
Chief Marketing Officer,
Luminary Micro



LUMINARY MICRO™

For more information visit
www.onARM.com

ARM

The Architecture for the
Digital World®

© ARM Ltd. AD158 | 01.09

66

A combined hardware/software solution provides an affordable way to read from and write to flash memory drives while in DOS.

99

file path construction, while access to files on the flash memory drive are limited to the current default directory of the flash memory drive. Once again, there is an opportunity for you to program a more general-purpose interface that supports more complete file path constructions for the flash memory drive and renaming files as they are copied. FREN lets you manually change a file name on the flash memory drive and FDEL can delete a file. FPUT, FGET, and FREN will overwrite only a destination file if a /y option is added as the first command parameter. (Note that the option /h will list instructions for any of the commands.)

The most challenging part of the source code development was managing the FIFO of the UART during serial port transfers. The limited computational power of the VDRIVE2 and the wide variation in personal computer hardware require careful coding of the serial port handshaking. The FIFO of the computer UART must be enabled and disabled at critical moments to maximize throughput without overrunning the VDRIVE2 buffer at high data rates. The VDRIVE2 provides an acknowledge handshake, so errors are always trapped. Even with the careful handshake, a file copy (FPUT or FGET) will fail on occasion. These failures are rare unless the computer hardware just cannot handle one of the high data rates. For large files (over a few megabytes), it probably makes more sense to reboot the system to Windows and use the full speed of a USB port. It takes about 50 minutes to copy a 30-MB file at 115,200 bps, which is close to the theoretical minimum time (about 46 minutes).

COMPILER & SOURCE CODE

The source code is posted on the *Circuit Cellar* FTP site. Although the source code is rather generic C

and should work with any C compiler, the code comes with a project file for compilation using Borland C++ version 1.1. The Borland compiler is available as a free download for non-commercial use (<http://dn.codegear.com/article/21751>). On the web site, you will find a lot of information for installing and using the compiler. For this project, installation on a Windows XP computer is straightforward. We use all installation defaults except the source disk (C drive instead of a floppy disk). To keep things simple, the source code is unzipped to a new directory C:\TC\FSOURCE, a subdirectory of the Turbo C++ compiler root. After everything is set up, double click the TC.EXE file in the C:\TC\BIN subdirectory to get character-based Turbo C++ GUI. If you want, use the properties option of the DOS window to switch to full-screen mode and flash back to the good old days of DOS.

Compile is as easy as install. Once Turbo C++ is running, type ALT-P to drop down the Project menu and select Open Project. Navigate to the FSOURCE directory and select MASTER.PRJ. You should end up back in the main Turbo C++ screen. Type ALT-C for the Compile menu and select Build All. That will create MASTER.EXE, the only file you need. Why is MASTER.EXE the only compiled file? Because all of the other .EXE files are just clones of MASTER.EXE with different names. When the program is started, it looks at the first parameter of the command line, which is the file name of the program; the program's file name determines what action to take. You can manually make copies of MASTER.EXE, renaming each copy to match the commands, or you can use the batch file MAKE.BAT to generate the entire set. Consolidating the object code to

Listing 1—The file name for each command is hard-coded in MAIN.C. Change the string in the strcmp function and recompile MAIN to rename a command.

```

if(!strcmp(szCommand, "FDIR.EXE", 8))          g_eCmd = COMMAND_FDIR;
else if(!strcmp(szCommand, "FPUT.EXE",        8))    g_eCmd = COMMAND_FPUT;
else if(!strcmp(szCommand, "FGET.EXE",        8))    g_eCmd = COMMAND_FGET;
else if(!strcmp(szCommand, "FCD.EXE",         7))    g_eCmd = COMMAND_FCD;
else if(!strcmp(szCommand, "FREN.EXE",        8))    g_eCmd = COMMAND_FREN;
else if(!strcmp(szCommand, "FDEL.EXE",        8))    g_eCmd = COMMAND_FDEL;
else if(!strcmp(szCommand, "FTER.EXE",        8))    g_eCmd = COMMAND_FTER;
else if(!strcmp(szCommand, "FBAUD.EXE",       9))    g_eCmd = COMMAND_FBAUD;
else      g_eCmd= COMMAND_UNKNOWN;

```

a single program simplifies development in many ways, but the downside is that you cannot rename the file operations. Thus, if you do not like the program name FPUT.EXE, you might be tempted to rename it to, for example, TOFLASH.EXE. However, the renamed program will just give you an error message. To change the name of a command, you will also have to change the name in the source code of MAIN.C, shown in Listing 1. Change the file name in quotes and then update the number of characters to match the new name. Do not exceed the 8.3 (12 character) DOS limit. After updating MAIN.C, repeat the Build All step above and then copy MASTER.EXE to the new command file name.

A BETTER INTERFACE

A combined hardware/software solution provides an affordable way to read from and write to flash memory drives while in DOS. In this implementation, inexpensive hardware obviates the need for resident drivers. While version 1 of the software can be a little awkward when the flash memory drive has complex directory structures, the source code and information presented here provide a pathway to better user interfaces. It is easy to envision a Norton Commander-like interface, or some other classic semi-graphical character-based user interface for selecting and operating on files and directories. We hope to hear from readers who take on this part of the challenge. Because USB ports have such blazing speeds compared to their serial predecessors, large files are best managed by setting up dual-boot operating

systems that enable you to defragment DOS for Windows when more serious file transfers are necessary. For smaller files, the VDRIVE2 and the new DOS commands are an ideal solution.

Although the software was written

for DOS, much of the code is reusable for microcontrollers with C compilers, or as an example for other embedded applications. If you build the level converter, you have a great set of tools for experimenting with the VDRIVE2. ■

Authors' note: This research was supported by the Intramural Program of the National Institute of Mental Health, National Institutes of Health.

Andrew Mitz (arm@nih.gov) is a Ph.D. research scientist at the National Institutes of Health where he studies the electrical activity of the brain. He received his Bachelor's and Master's degrees in electrical engineering from Washington University and the University of Maryland, before entering a medical research program at Emory University. Andrew's laboratory work involves instrumentation of physiological signals (e.g., muscle activity, heart rate, eye tracking, and microvolt recordings of brain cells). He has worked on the development of many real-time embedded systems, including robotics. In his free time, Andrew collects and repairs antique radios and supports emergency communications through amateur radio. He has also developed devices for people with disabilities. Some of Andrew's designs are now in commercially shipping products, and many end up as publications in a wide variety of professional and hobby magazines. His favorite moments in embedded design are at the beginning and the end of each project. "The middle," Andrew says, "is often quite maddening!"

Jon Daley is an embedded software engineer who currently can be found alternating between the two extremes of assembly language and ajax/php for his startup company Lime Daley. Wherever he is, you can find him at <http://limedaley.com> or circuitcellar@jon.limedaley.com.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/226.

SOURCES

FTDI VNC1L Host controller chip and VDRIVE2 reader module
Future Technology Devices International | www.ftdichip.com

NIMH Cortex
NIMH Laboratory of Neuropsychology | www.cortex.salk.edu

Transformerless Power Supply

Instead of buying expensive transformers or converters to run your simple circuits, try using a transformerless power supply. Tom used one to construct a switch that controls lights in a pantry. When the door is opened, a processor senses a reed relay and powers up the outlet. Three minutes later, if the door is not closed, the outlet shuts off.

Do you need a low-cost way to run a simple 5-V circuit from 120-VAC power without bulky and expensive transformers or converters? If so, a transformerless power supply may be just what you require.

Before I begin, review the circuit in [Figure 1](#). Do you see something of concern? Do you understand why your mother would warn you against building this circuit?

If you said “no,” stop right here and read no further. This is by no means a beginner’s circuit. It should be considered an *experimental* design. I make no claim that this design is safe. You are responsible for knowing and implementing all of the necessary safety precautions when working with this or any circuit connected

directly to mains voltages. Now, make your mother happy by being safe and responsible. Have fun and let’s continue.

THE PROBLEM

This design started from my dissatisfaction with both the Smart Home Systems X10 and Insteon light switch controllers. I never could get a reliable signal out to all of the modules. Plus, I really needed only a simple open/close sensor and maximum on-timer functionality. I thought there should be an easier solution. Having done some recent work with an Atmel AVR processor, it seemed a simple answer would be to use an AVR processor monitoring a switch and controlling a TRIAC.

Here’s a quick preview of the end of the story just to give you an idea of the potential applications. I built an AVR switch with my transformerless power supply. My AVR switch controls lights I placed inside a pantry (see [Photo 1](#)). The AVR switch resides inside an outlet box in

the inside wall just over the door. The outlets on the right half of the box go to small under-cabinet lights mounted along the inner pantry wall. The AVR switch monitors a reed relay embedded in the door frame directly below the outlet box. When the door is opened, the AVR processor senses the reed relay and powers up the outlet. Three minutes later, if the door is not closed, the outlet is turned off.

Refer to the simple timed light switch in [Photo 2](#). On the left side of the wall plate, this AVR switch configuration has

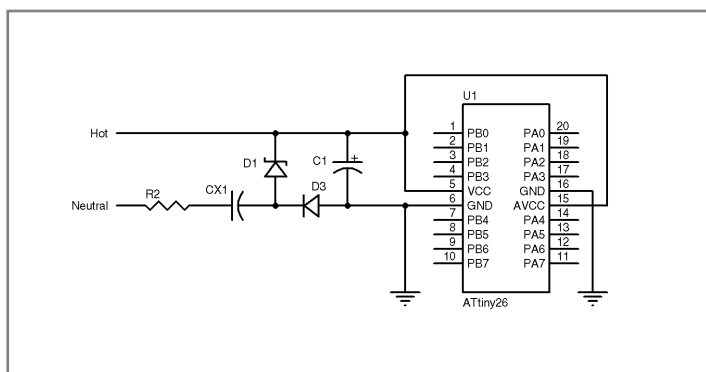


Figure 1—This is the basic dangerous circuit.

Photo 1—This AVR switch is mounted next to the receptacle that it controls. Opening the door powers up the receptacle and turns on the lights inside the kitchen pantry.



a push button and a small LED nightlight. Pushing the button turns on the light. Pushing it again turns off the light. If I leave it on for longer than 45 minutes, the light automatically turns off. As a result, my lights no longer stay on all day and my electric bill is under control.

As you can see in [Photo 3](#), the AVR switch is compact. The AVR processor is at the top. The transformerless power supply components are at the bottom.

THE SEARCH FOR A SOLUTION

At the beginning of this project, the idea was to use a low-power AVR processor to control a TRIAC. A quick Internet search turned up numerous examples of how to use a microcontroller with a TRIAC to control a 120-VAC load. One example is presented in Microchip Technology application note TB094, "Dimming AC Incandescent Lamps Using a PIC10F20." I thought this would be easy, so I started listing components: a processor, a TRIAC, a transformer, a rectifier, a capacitor, a zero-crossing detector, a TRIAC optoisolator, and the list went on. It quickly became apparent that this would never fit in the 1.5" × 2.5" × 1" area inside a standard

wall outlet box. I needed something smaller and simpler.

The largest components obviously compromised the power supply. A typical wall-wart power supply, with its transformer and rectifier, would have taken up all the space by itself. Thus, the problem became how to make a tiny power supply. I then remembered I had a handful of unused cell phone mini-chargers lying around. They were small, so I thought perhaps there was an answer inside. A victim was selected so I could find out. The circuit board from the mini-charger is shown in [Photo 4](#).

As I expected, there was no large transformer or rectifier involved; instead, it was a miniature switching power supply. Unfortunately, even with the smallish inductor, the overall size was still too large for the space I had to work with. Thus, after estimating the size of a complete solution, the mini-switcher was removed from the list.

If you are still interested in a mini-switcher for your own project, check out the TinySwitch-II family of parts from Power Integrations. (A TNY266PN is shown on the left in [Photo 4](#).) A small 4- to 15-W power supply is possible.

Another interesting part I ran across during my Internet searches was a Supertex SR086/87 adjustable off-line inductorless switching regulator. This is a true inductorless power supply that works by switching a transistor on or off when the rectified AC is below or above the desired output voltage. This part can source only 100 mA, which was sufficient for my purposes, but it had one major problem. The circuit called for a rectifier that would cause the DC outputs to float relative to the AC mains. Therefore, I would not be able to drive a TRIAC directly from the processor outputs.



Photo 2—In this variation, the AVR switch turns the lights on when the push button is pressed and automatically turns the lights off if the maximum on time is exceeded.

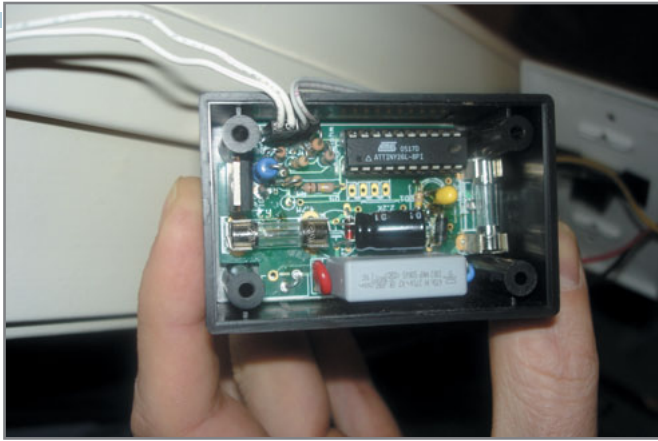


Photo 3—The low parts count enables the transformerless power supply, processor, and TRIAC to all fit in a 1.75" x 2.5" enclosure.

Using an optoisolator to drive the TRIAC would have been possible, but it would have increased the parts count and thus the overall physical volume. Still striving for the minimum size and parts count, my search continued.

Finally, I wondered how the original X10 modules fit a complete power supply into such a small space. This seemed like a potential path to a solution, so I decided to dissect one (see [Photo 5](#))!

Inside the X10 module was a notable absence of any type of transformer, power regulator, or rectifier; it included mainly resistors and capacitors. It appeared that the inductor on the right (see [Photo 5](#)) was used only for reducing the output noise caused by the X10 dimmer functionality. Few components were left to make up the actual power supply. The X10 module was the essence of a minimal design. So, the question became, How did X10 build this transformerless power supply?

It turned out that the phrase "transformerless power supply" was a good Internet search term. I found various designs that all focused on a simple capacitor used to siphon power from the AC mains. One find was a tutorial on transformerless power supply design published as a Microchip technical brief, "TB008: Transformerless Power Supply." I also found a Microchip application note, "AN954: Transformerless Power Supplies: Resistive and Capacitive," that actually went into the calculations for the various component sizes. This finally appeared to be a solution. I now had the beginnings of my experimental transformerless power supply.

MAINS-POWERED SUPPLY

My first objective was to use the calculations in application note AN954 to determine the required component sizes. I based the design on the high-voltage capacitor because its physical size, cost, and availability would control the overall design. For example, an X2 class capacitor is relatively large for its small farad value. Higher value X2 capacitors increase rapidly in both size and cost, which greatly limits the available component

choices. Application note AN954 also makes some safety suggestions, which I incorporated into the basic design. Finally, note that while not currently Underwriters Laboratories-approved, I knew it would be possible to produce an enhanced design to meet UL approval requirements. Read the "Other Considerations" section in the application note or visit the UL web page (www.ul.com) for more information.

After crunching some numbers, I defined the basic circuit in [Figure 2](#). By design, this circuit can provide only about 10 mA of continuous current. Drawing any more current will cause severe output voltage sag. Unfortunately, I found that attempting to create a higher current design increased only the size of the X2 capacitor even more rapidly.

I read about alternative designs with higher available currents in the application note. One such alternative involved adding a rectifier to the front end of this design. However, as before, I knew that using a rectifier would make it impossible to drive a TRIAC gate from a processor output directly.

“To minimize exposure to high voltages during construction, I decided to build a transformerless power supply prototype as a unit separate from the logic circuit prototype.”

Another alternative was a resistive design, rather than a capacitive one. The problem with the resistive design was that it required a 10-W resistor, which would result in both size and power dissipation issues. Ultimately, I determined that the 10-mA capacitive design was a reasonable compromise between available current and component size.

To minimize exposure to high voltages during construction, I decided to build the transformerless power supply prototype as a unit separate from the logic circuit prototype. The idea was to allow for the troubleshooting of the logic section from a normal 5-VDC supply and risk only high-voltage exposure when troubleshooting the power supply itself. However, at some point, I knew both sections would have to be integrated, but being built as separate prototype units helped minimize risk.

Once I had an assembled prototype of a transformerless power supply, the first issue was trying to determine how well the power supply was working. The problem was that the DC ground for the circuit design floated just 5 V below the hot line of the 120-VAC mains. Thus, if I had tried to measure the 5-DC output with my mains-powered oscilloscope, and if I was

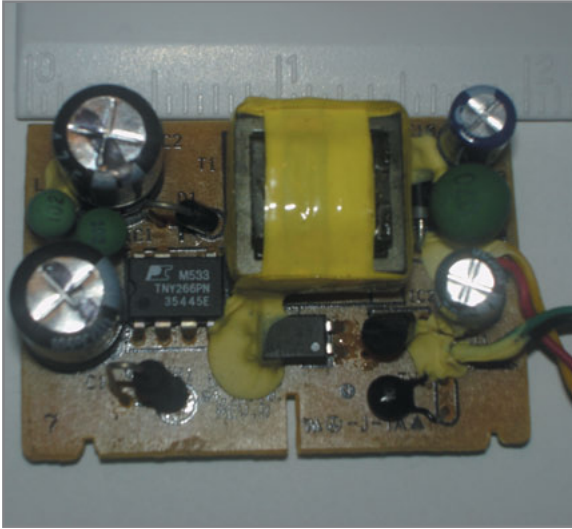


Photo 4—This is a TinySwitch-II miniswitcher.

extremely lucky, I might have seen a 115-VAC signal. However, it was much more likely that I would have simply fried everything. The dilemma was that the signal ground on the oscilloscope was at Earth ground. If I had tried to connect both grounds together, the 115-VAC potential difference between the two grounds would have resulted in substantial current.

The solution would have been an isolation transformer. This would have effectively kept the two grounds separated and enabled me to connect the oscilloscope signal ground to the circuit DC ground and observe the 5-VDC output. However, even with the isolation transformer in place, there would still be 120

VAC in the circuit. So, it still would be a dangerous circuit.

Unfortunately, I didn't have an isolation transformer handy, so I had to improvise by wiring two step-down transformers secondary-to-secondary (see [Figure 3](#)). With a pair of 120-/24-VAC transformers wired back-to-back, the 120-VAC input was converted to 24 VAC and then back to 120 VAC on the output side. More importantly, no DC current could flow between the input and output. With my

improvised isolation transformer, I could then verify the proper operation of my transformerless power supply prototype.

If you build this isolation transformer, think the component selection through and don't overload the transformers. Remember 100 mA at 120 VAC becomes 0.5 A in the 24-VAC winding, so plan accordingly. This can really become an issue if you plan on testing your circuit while it's actually controlling a load! A 60-W bulb could pull over 2.5 A through the 24-VAC windings. Fuse appropriately and check the VA rating of the transformers. Be careful. Check your numbers and double-check your setup before you proceed. Remember that you still have 120-VAC potentials in the circuit even with the isolation transformer!

AVR POWER SWITCH

With my experimental transformerless power supply designed, I needed to add an AVR processor and a TRIAC to finish my AVR power switch. Because this was intended to be an experimental platform, I added a few expansion options to the circuit—that is, at least as much as I could shoe-horn into such a small space.

I chose an Atmel ATtiny26L. The “L” variant had an operating range down to 2.7 V with an idle current of 0.18 mA. Again, low power consumption was critical because the transformerless power supply could provide only a small amount of continuous current. Another benefit was that with the wide operating range, it was possible to draw a little more current from the power supply and still survive the resulting voltage sag.

I then selected an isolated gate TRIAC. For smaller loads, the TRIAC could operate without a heatsink; but for larger loads, it would need a heatsink. I placed my final design in a plastic box with a metal lid. The metal lid made a good heatsink. But because of the transformerless power supply, the TRIAC had to be an isolated tab version. Otherwise, the TRIAC tab and thus the heatsink/metal lid would have also been energized.

With the processor and TRIAC in place, I included two digital switch inputs in the circuit, one zero-crossing interrupt, one analog input with an adjustable potentiometer, one digital output with a display LED, and the basics of a two-wire interface. Because the entire circuit was at mains potential, any wiring leaving the circuit was also at mains potential. So, to help improve the safety of the switch inputs, I placed current-limiting resistors on both sides of the switch inputs. The chosen resistor values were the highest values possible that still allowed the processor to detect the switch closure. Putting everything together resulted in the circuit shown in [Figure 4](#). Remember: Even with these precautions, you still need to install the box and circuit so that no one can come into direct contact with potentially energized components.

One of this transformerless power supply's advantages was that it had a built-in zero-crossing signal. However, with the capacitive version of the design, the zero-crossing signal had a substantial phase shift. I was not planning to use any dimming functionality, so the zero-crossing signal

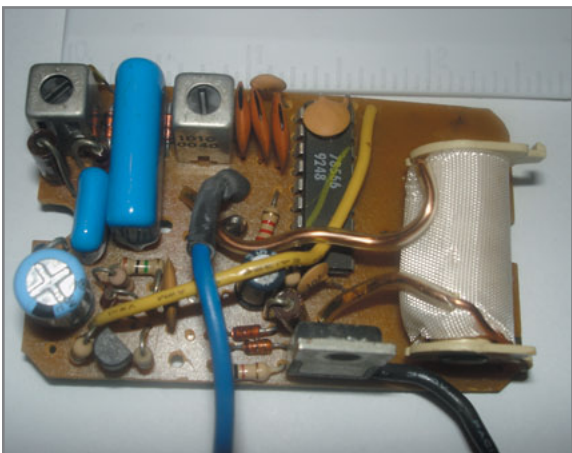


Photo 5—This is an opened Smart Home Systems X10 light switch.

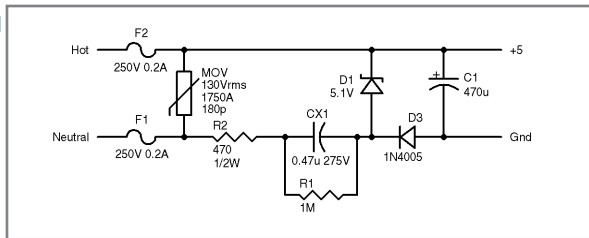


Figure 2—This is a basic capacitive transformerless power supply.

wasn't necessary. However, because this was an experimental platform and the signal was available, I included it anyway. One result of that decision was that I could also use the zero-crossing signal to shorten the time the TRIAC gate needed to be powered. As a result, the overall circuit current demand was reduced. To implement the TRIAC control, the software was designed to turn on the TRIAC immediately after the zero-crossing and to leave it turned on only until sufficient current was being conducted through the TRIAC to latch the gate. At that point, the TRIAC gate could be released and the TRIAC would stay energized.

A second issue associated with the zero-crossing signal was a tendency to ring when there were switching transients present on the incoming 120-VAC line. I tried to compensate for this. But if the circuit is used in a noisy environment, there may be some false triggering of the zero-crossing interrupt.

SOFTWARE

The sample software for the AVR power switch platform was written in WinAVR C. It was relatively straightforward to write. Code size and performance were not really issues with this application.

The entire software package comprised several interrupt-driven functions. The first routine, *Main*, was nothing more than an empty loop. The *IOInit* routine did just that; it took care of initializing the processor. *OnTime* was a simple routine to look at the ADC value and select the appropriate "maximum on time." A minimum ADC value equated to a 5-minute delay while

increasing values gave a longer delay up to a maximum of 59 minutes.

The interrupt service routine (*INT0_vect*) was the heart of the software. It was driven by the zero-crossing signal from the transformerless power supply.

The routine was respon-

sible for determining the remaining "on time" and debouncing the two switch inputs. It also used the current TRIAC state, the input switch states, and the remaining maximum on time to determine the next TRIAC state. If the routine determined that the TRIAC should be on, it started *Timer1* with the necessary delay time to compensate for the zero-crossing detector's phase shift. Then the *Timer1* compare interrupt service routine (*TIMER1_CMPB_vect*) turned on the TRIAC and restarted *Timer1* for the TRIAC latching delay. When the *Timer1* compare routine triggered again, the TRIAC was released. Remember, at this point, the TRIAC would stay latched by itself until the next zero crossing. This also meant that in the future, if I wanted to modify the software to support dimming, the basics were there. It would just involve lengthening the phase-shift delay as required to control the TRIAC turn-on point.

Input ports PA0 and PA1 were defined as the inputs for the switches. PA0 was intended to act as a door open/door close switch. This means shorting PA0 to DC ground moves the TRIAC state to off and floating PA0 moves the TRIAC state to on.

Switch input PA1 was intended to act as an On/Off push button. Each press and release of a push button

would toggle the TRIAC state. Yes, the software was designed so that both PA0 and PA1 could be used at the same time.

ADC2 was used to sample the potentiometer to determine the maximum on time. Output port PA4 was designed to control the indicator LED, while output port PA7 was used to drive the TRIAC gate.

If you decide to try your hand at building this circuit after reading this article, there are a few other considerations to keep in mind. Skip in-circuit programming capability; otherwise, I guarantee that at some point you *will* accidentally hook-up your in-circuit programmer while the circuit is connected to the mains. I'm not sure if you and your PC will survive that connection.

Always use your isolation transformer if there is even the slightest possibility that you or your equipment could come into contact with the circuit. *It's still a dangerous circuit.*

Remember to measure! Your voltmeter and oscilloscope are your friends. Before you make a connection, measure the potential between the connection points. Is it really 5 VDC, or did the 120 VAC sneak in? It's much better to measure often rather than smoke parts.

Construct your circuit so that no external component can be touched. For example, use in-wall magnetic reed-relay switches, such as those found in alarm systems, for external switches. Remember that despite the fact that the processor is running at 5 VDC, it is actually at a 120-VAC potential.

For any external push buttons, make absolutely sure that there are no exposed grounded or other metal surfaces. Those metal surfaces will

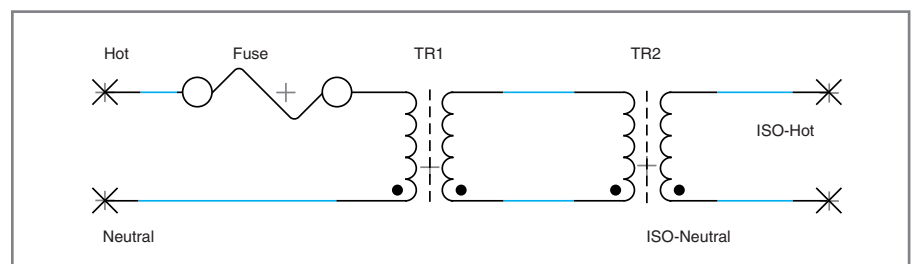


Figure 3—This is a simple isolation transformer.

What's the difference?



Price!

Electronic components work no matter what price you pay. Jameco carries everything you expect at prices below what others charge. But the price savings don't stop there. Jameco offers additional savings with its array of house brand and factory-overrun products.

The Jameco difference begins with the industry's highest quality catalog and is backed by the industry's longest warranty plus much more.



- Over 100,000 skus
- 99% of catalog products are in stock right now
- Low price guarantee

JAMECO[®]
ELECTRONICS
1-800-831-4242

◀◀ Order your FREE catalog today at www.Jameco.com/Price

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

Dear Design Community:

Circuit Cellar would like to thank the following companies for their sponsorship of our 2008 Sample Pool product evaluation program. A wide cross section of our readership received tens of thousands of dollars worth of hardware and software development tools because of the generosity of these sponsors and their recognition of the value of Circuit Cellar's audience.

- Crossware Products, Inc.
- Keil
- Vesta Technology Inc.
- FlexiPanel Ltd.
- Saelig Co., Inc.
- Reach Technology, Inc.
- Luminary Micro, Inc.
- Custom Computer Services, Inc.
- SEGGER Microcontroller System LLC
- Multilabs
- Schmartboard
- microEngineering Labs, Inc.
- ADM Designs, LLC
- Total Phase, Inc.

The 2008 Sample Pool program was initiated through News Notes, Circuit Cellar's e-newsletter. To subscribe to this free newsletter, visit www.circuitcellar.com/newsletter/. Although some of the newsletter material is archived on our site, certain programs are only available through the email portion of this monthly publication.

Please note: Circuit Cellar is now preparing for its latest readership survey and is currently signing up sponsors for another Sample Pool program in conjunction with the survey's promotion. Watch for additional information about how you can participate in this year's survey through Circuit Cellar's May edition of News Notes and www.circuitcellar.com. I look forward to having your input and being able to help many of you gain access to product samples from Circuit Cellar advertisers.

Sincerely,



Sean Donnelly, Publisher
Circuit Cellar
sean@circuitcellar.com

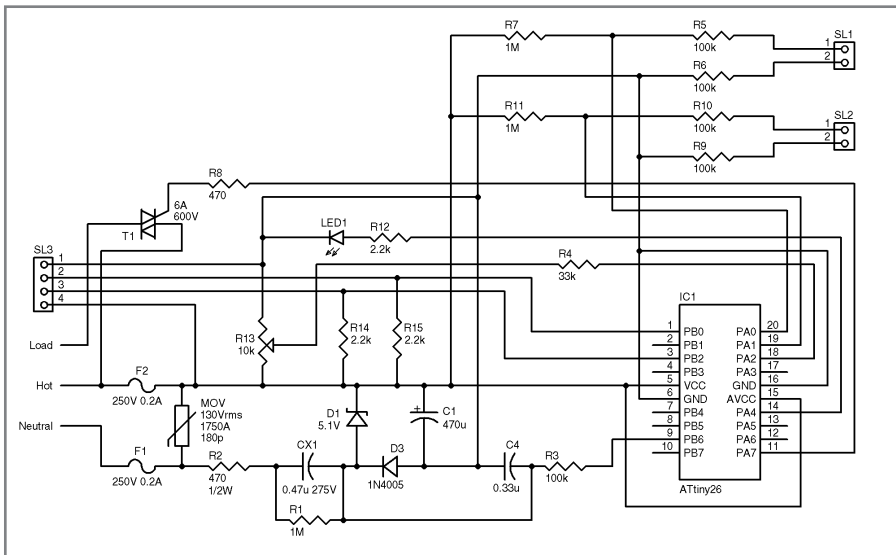


Figure 4—The completed circuit contains the transformerless power supply, a TRIAC, an optional zero-crossing signal, and various other I/O options. The end result is a simple experimental platform, which can be easily configured to support many different uses.


most likely be at circuit ground, not Earth ground, and thus at 120-VAC potential! Also be aware of your push buttons' failure mode. Do the plastic tops pop off to expose metal underneath?

Keep the unfused areas as small as possible. That way, if you short something out, there's a better chance that what you shorted will be behind a fuse. Also, keep the 120-VAC side of the circuit together and away from the 5-VDC side as much as possible. This helps reduce the voltage differential between components in case something shorts out.

FUTURE PROJECTS

I plan to include a daylight sensor and a clock in the next version of my AVR switch. But what ultimately makes it into the next version will depend on the project's space and power consumption requirements.

If you'd like to experiment with your own AVR switch, I have circuit boards and part kits. You'll have to assemble everything, and more importantly, determine for yourself if this circuit is safe and appropriate.

I hope you find this information as useful and as interesting as I have. Maybe your next project will also incorporate a transformerless power supply! 

Author's note: Your safety is your own responsibility. You must use equipment and safety gear properly, and determine whether you have adequate skill and experience. Power tools, electricity, and the other resources used for these projects are dangerous, unless used correctly and with adequate precautions, including protective gear. Some illustrative photos do not depict safety precautions or equipment, in order to show the project steps more clearly. Use the instructions and suggestions listed here at your own risk. It is your responsibility to ensure that your activities comply with applicable laws. You can download the sample code at www.JenRathbun.com/Electronics/AVRSwitch.html.

Tom Struzik (tpstruzik@earthlink.net) has been building and taking things apart from an early age. He built his first Heathkit project at 12 and sold his first

computer program at 16. Tom has a BSEE from Purdue University, and currently works for a Fortune-100 chemical company in its engineering systems organization as an IT systems architect. He continues to build software and hardware projects at home to "keep his hands dirty." One of Tom's current projects, the "Cat Faucet," was recently covered by Engadget.com.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/226.

RESOURCES

R. Condit, "AN954: Transformerless Power Supplies: Resistive and Capacitive," Microchip Technology, Inc., DS00954A, 2004.

S. D'Souza, "TB008: Transformerless Power Supply," Microchip Technology, Inc., DS91008C, 2008.

Microchip Technology, Inc., "TB094: Dimming AC Incandescent Lamps Using a PIC10F20," DS91094A, 2005.

More information on kits and discussion forums on this project: Sixerdoodle Electronics, "AVR Switch," www.JenRathbun.com/Electronics/AVRSwitch.html.

Underwriters Laboratories, Inc., www.ul.com.

SOURCES

ATtiny26L Microcontroller

Atmel Corp. | www.atmel.com

TNY266PN TinySwitch II

Power Integrations, Inc. | www.powerint.com

SR086 Switching regulator

Supertex, Inc. | www.supertex.com



A World Without NTSC

Bridge the Gap Between NTSC and VGA

NTSC will soon be a thing of the past. So, what will you do in a world without the NTSC? Jeff answers that question and more. Read on to learn how he is using a chip to bridge the gap between the NTSC and VGA formats.

The United States Federal Communications Commission (FCC) has mandated that most broadcasters cease transmitting NTSC in favor of digital television. What will happen in a world without NTSC?

I was raised on NTSC. My uncle Ray was the first in my family to have a color TV. I remember saying, to my uncle's dismay, "I prefer black and white to color; look how awful the picture is." The grainy, fuzzy, rainbow-colored objects were tough to watch. And I'll admit now that this may have been due to early set design and fringe reception. Back then, we were considered fortunate if

we could receive all three major networks. Today's TVs (or should I say those of the recent past) do a great job at receiving broadcast signals. Strong stations give crystal-clear pictures. I don't know the exact numbers, but many viewers have now given up their antennas for cable or dish connections. Their broadcasts are already digital. Their receiver boxes translate the ones and zeros into NTSC output so we can connect our legacy TVs. For those of you still using an antenna for reception, the new digital broadcast transmissions can not be received directly by legacy TVs. They require a converter box to stupefy the new broadcast format down into an NTSC output that can be used by the outdated equipment.

I'm not going to debate the pros and cons of the new digital broadcast format. Instead, I want to point out that this means an end to using inexpensive NTSC TVs and monitors as display devices. All of this may have started back with Don Lancaster's design of the TV typewriter that appeared on the cover of *Radio-Electronics* magazine in September 1973.^[1]

NTSC is a composite video standard used by the first personal computers (TRS-80 and Apple) and video game systems (Coleco and Atari). As higher resolutions were required, the composite video signal was separated into multiple components allowing finer control of the video format. While (S)VGA uses discreet signals, each color is still basically analog. One of the newer standards, the Digital

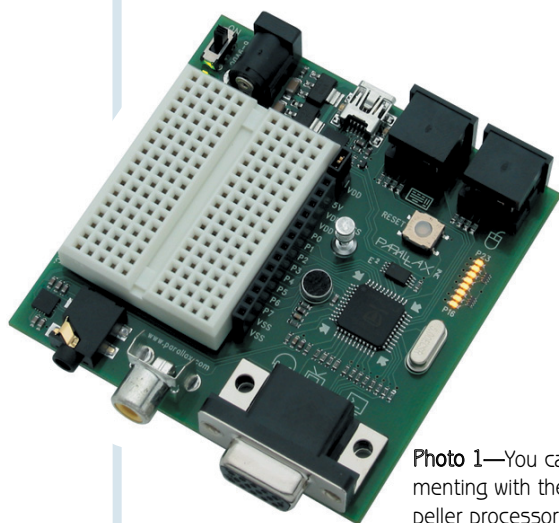


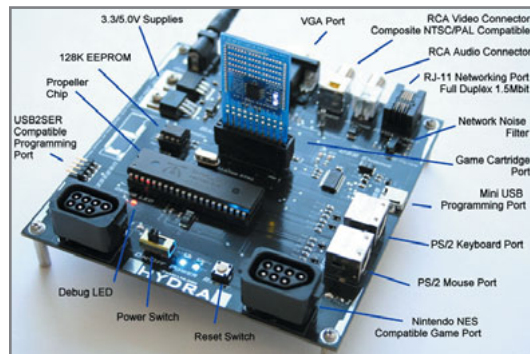
Photo 1—You can start experimenting with the Parallax Propeller processor for \$80 using the Propeller demo board.

Video Interface (DVI) combines DVI-D (digital mode) and DVI-A (VGA in analog mode). Of the advertised interfaces on today's TVs—such as, HDMI, component, VGA, S-Video, RF, and composite—which option do you think will be the first to go on future models?

PROPELLING

I spent most of my early hard-earned pocket money playing Space Invaders and Asteroids at the local hangout. I thought I had since shed my addiction for gaming. Little did I realize the demon had only moved into the shadows. I continually collect products and technologies that I think have potential

Photo 2—The HYDRA game console is based on the Propeller chip. The experimental console comes complete with a P52 mouse, a P52 keyboard, a game controller, a power supply, and cables. It also includes the book *Game Programming for the Propeller-Powered HYDRA* and a CD for \$200.



for future spotlight time in one of my monthly raves. For instance, after having a Parallax Propeller chip sitting on my shelf for a few years—along with Andre LaMothe's *Game Programming*

for the Propeller-Powered HYDRA—I recently started playing with it. I read "GPFTPPH," but it wasn't until I saw the FCC's writing on the wall that I understood how the Propeller could

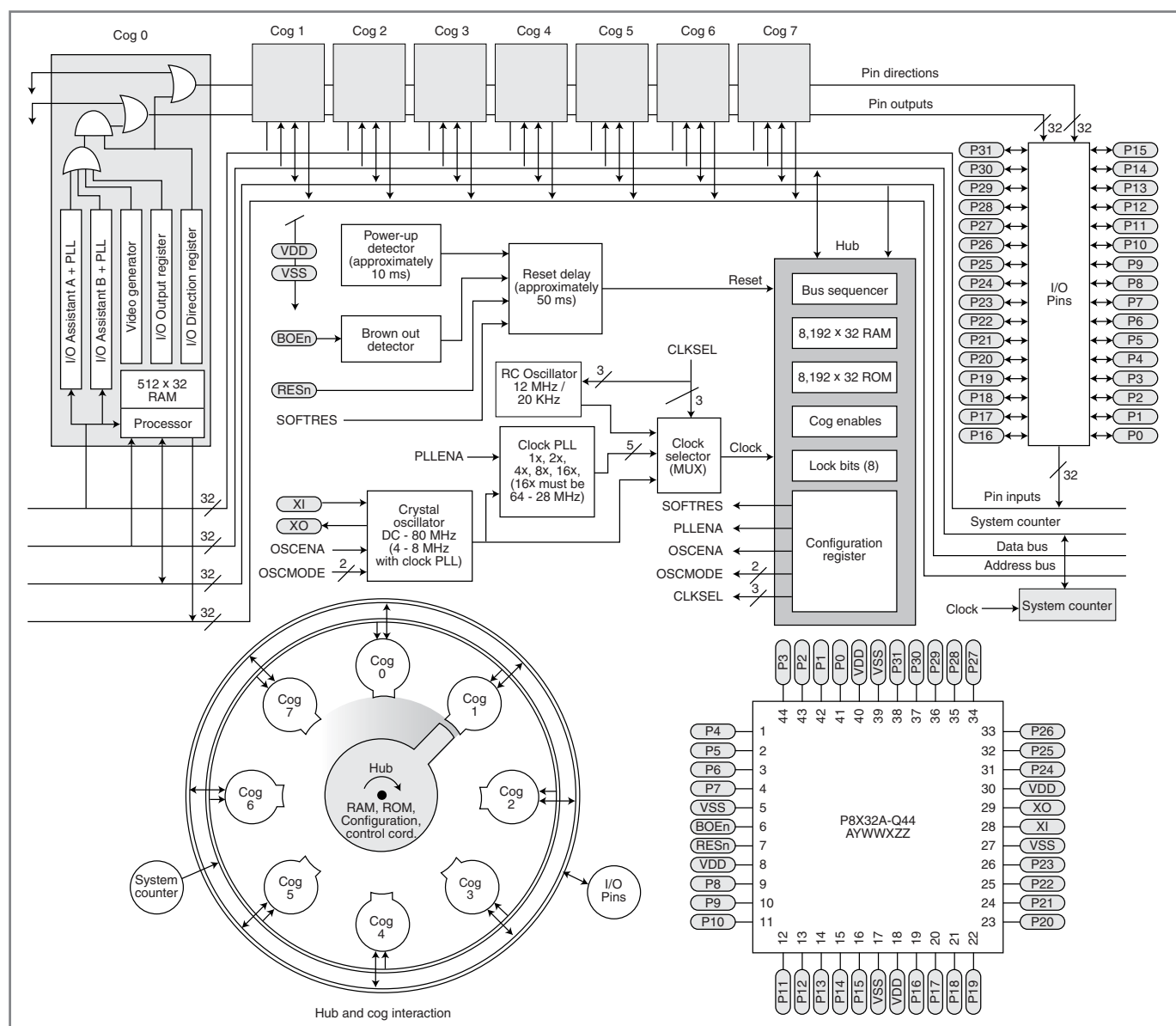


Figure 1—The Propeller block diagram shows the interaction between the hub and eight cogs. Each cog has access to all I/O pins and the hub's RAM and ROM, as well as its own RAM.

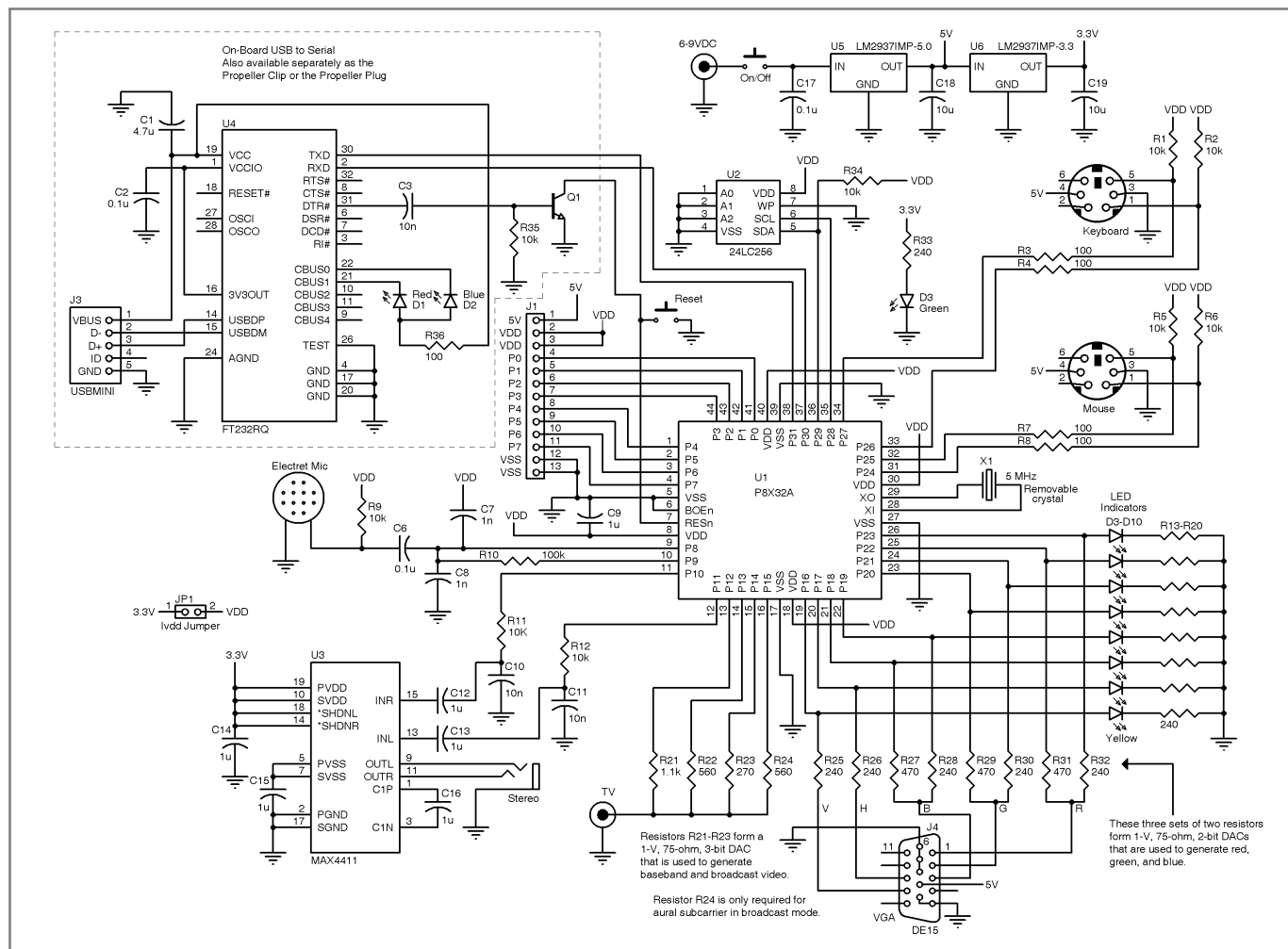


Figure 2—The Propeller demo board schematic shows how simple resistors added to the Propeller's I/O pins act as a DAC (with a monitor's 75-Ω input impedance) creating an inexpensive interface for either NTSC or VGA video.

easily bridge the gap between NTSC and VGA. So, I took my Propeller demo board and went to work (see [Photo 1](#)). The HYDRA is shown in [Photo 2](#).

The Propeller chip consists of eight independent processor units called cogs. Each cog has its own 512 double words (32 bits) of RAM. The last 16 bytes of this RAM are special function registers that enable the cog to access all I/O pins, its own counters, and a video generator. The RAM is used to hold code and local VARs. Each cog executes its own code independently, yet all cogs run from the master clock. The Propeller can take oscillator or XTAL input or use an internal RC oscillator to drive the internal clock directly or through a $2 \times 4 \times 8 \times 16 \times$ PLL for a maximum clock speed of 80 MHz!

Note that in addition to eight cogs, there is an additional device called the hub (see [Figure 1](#)). Besides handling the

basic reset, brown out, and master clocking, the hub has its own memory, both RAM and ROM, with 8,000 double words each. Hub ROM contains character definitions, math functions, a bootloader, and a Spin interpreter. During reset, the bootloader loads COG0 with some code that checks for communication (enabling you to take control), checks for an external EEPROM, and loads it into the hub's RAM or shuts down all operations. If an application has been transferred into the hub's RAM, then the SPIN interpreter is loaded into COG0 and begins to execute the application in the hub's RAM.

Like all microcontrollers, the Propeller has a number of assembly instructions that make up its vocabulary. You may want to write your application code (or parts of it) in assembly language. However, there are those who detest having to work with assembly code, so the Parallax folks created a

higher-level language called Spin. It removes much of this burden by providing a bunch of useful functions.

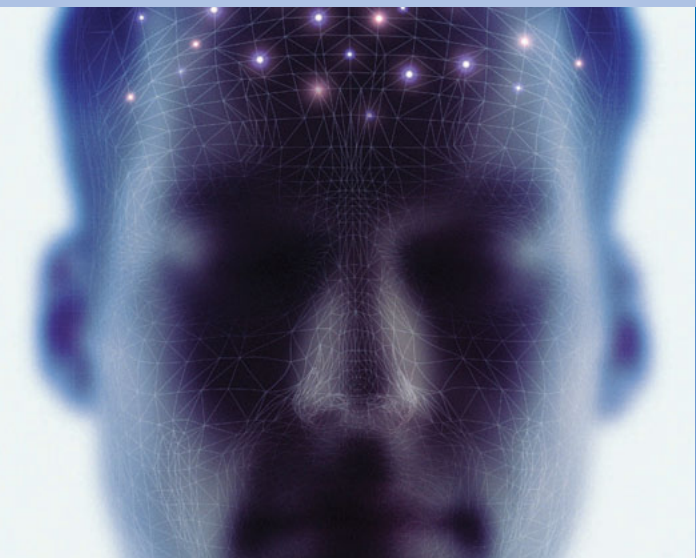
While each cog executes on its own, your application directs this operation and will determine exactly how a cog will be used. For instance, if your application requires asynchronous serial communication, you might write a cog application that samples the RX input looking for a start bit, and upon reception uses the system clock to continue sampling the input at the proper data rate. Collected bytes might be put into hub RAM (available to any cog). The hub continuously does a "round robin" on all of the cogs. It controls when a cog has access to the system RAM and keeps cogs from simultaneous access. A cog may have to wait its turn, which is a maximum of once every 16 clock cycles, depending on whose turn it is. If a cog needs to update multiple RAM locations prior to allowing

sensors expo & conference

Conference: June 8-June 10, 2009
Exhibits: June 9-June 10, 2009
Donald E Stephens Convention Center
Rosemont, Illinois
www.sensorexpo.com

Advances in Measurement, Monitoring, Detection & Control

New Approaches • New Technologies • New Applications • New Ideas



Don't Miss the Sensors Opening Keynote



Cassini: Five Years at Saturn **Dr. Kevin Grazier**

*Investigation Scientist & Science Planning Engineer,
Cassini/Huygens Mission to Saturn & Titan, NASA's
Jet Propulsion Laboratory (JPL)*



A World of Interconnected Sensors

Beth Wozniak

*President of Sensing & Control, Honeywell
Automation & Control Solutions*

This Year's Conference Program Covers 18 Tracks

- Sensor Interfaces & Sensor Integration
- Sensor Systems Design
- RF Sensing
- Wireless Sensor Networks
- Energy Harvesting
- Energy Conservation
- Low-Power Sensing
- Harsh Environments
- Position Sensing
- Fiber Optics
- Machine Health & Predictive Maintenance
- Smart Materials
- Novel Approaches to Measurement & Detection
- Environmental Monitoring
- Business Trends & Issues
- Wireless Standards
- Location-Aware Sensing
- Novel Approaches to Biodetection

Register Today for Your Conference Pass at the Early Bird Rates! Or, Sign Up Now
for a FREE Expo Hall Pass! Visit www.sensorexpo.com or call 877-232-0132
or 972-620-3036 (Outside U.S.). Don't Forget to Use Your Source Code: 303M

Bits	Video configuration register (VCFG)			
31	n/a			
30:29	VMode (Enable)			
	0	0	Disable VSU	
	0	1	VGA Mode	
	1	0	NTSC Mode (Broadband on upper pins, baseband on lower pins)	
	1	1	NTSC Mode (Baseband on upper pins, broadband on lower pins)	
28	CMode (colors/shades)			
	0	Two-color mode		
	1	Four-color mode		
27	Chroma 1 (Broadcast)			
	0	Disable chroma (color) on broadband		
	1	Enable chroma (color) driver		
26	Chroma 0 (Baseband)			
	0	Disable chroma (color) on baseband		
	1	Enable chroma (color) driver		
25:23	Aural subcarrier (source)			
	0	0	0	Use COG 0's PLLA
	0	0	1	Use COG 1's PLLA
	0	1	0	Use COG 2's PLLA
	0	1	1	Use COG 3's PLLA
	1	0	0	Use COG 4's PLLA
	1	0	1	Use COG 5's PLLA
	1	1	0	Use COG 6's PLLA
	1	1	1	Use COG 7's PLLA
22:12	N/A			
11:9	VGroup (Port drive)			
	0	0	0	Group 0 (P7:0)
	0	0	1	Group 1 (P15:8)
	0	1	0	Group 2 (P23:16)
	0	1	1	Group 3 (P31:24)
	1	x	x	Reserved
8	n/a			
7:0	VPins (Pin drive)			
	0	1111		Driving lower four pins only (NTSC)
	1111	0000		Driving upper four pins only (NTSC)
	1111	1111		Driving all eight pins (VGA)

Table 1—This 32-bit register defines how the VSU hardware is used. The upper bits define the NTSC/VGA modes, resolution, chroma, and audio carrier source, while the lower bits define which processor pins are used for output.

others to access the data, it can indicate this with a lock-flag. Other cogs should respect this flag and cease access until it is cleared.

There are no interrupts on the Propeller. Think of a cog as an interrupt routine that continuously executes its code, independent of other cogs. Every cog can read and write to every I/O at any time! You can use this to your advantage, but without proper attention, it can cause you headaches. Any pin configured as an output by one cog will force the configuration of the pin to an output even if another cog is trying to use the pin as an input. Any cog outputting a high on a pin will force the pin high even if another cog is outputting a low to the same pin.

The hub's RAM (\$0000-\$7FFF) will hold your application after it is transferred at boot time. The hub's ROM code consists of 256 printable characters and graphics (\$8000-\$BFFF), Log and Anti-log tables that help convert between base-2 and

floating point (\$C000-\$DFFF), a sine table for 0° to 90° with 0.0439° resolution (\$E000-\$EFFF), and the bootloader/Spin interpreter (\$F000-\$FFFF).

VIDEO HARDWARE

Each cog has its own video hardware consisting of two configuration registers and the ability to stream data using Propeller output pins via the video streaming unit (VSU). (Note that while the primary use here is video, don't overlook the audio possibilities.) The digital outputs are meant to interface to an external DAC producing composite NTSC video output. Because a composite monitor presents a 75-Ω load, discrete resistors can be used to implement a DAC (see Figure 2). The VCFG 32-bit register is used to configure the VSU in a number of modes (i.e., Composite Baseband, Composite Broadband) (55.25 MHz = channel 2) or VGA (see Table 1). The VSCL 32-bit register contains two values: the number of CTLA PLL clocks per pixel (PixelClocks), and the number of clocks per frame (FrameClocks) (see Table 2). A CTRA PLL clock is based on your XTAL value, the PLL multiplier, and the CTRA PLL divider. When using NTSC, the active pixel area of a scan line is 52.6 μs. If you want to divide this into 256 pixels, that's approximately 205 ns/pixel (52.6 μs/256 pixels). If the CTRA PLL clock is running at 40 MHz, that's 25 ns (1/40,000,000). The closest you could come to 205 ns would be to use a count of eight CTRL PPL clocks. That would be 200 ns (PixelClocks = \$08). If you were using 2-bit (four-color) mode, you would be storing 16 pixels worth of 2-bit information in each 32-bit double word. This means that the FrameClocks value would need to be 16 pixels × PixelClocks, in this case 8 (FrameClocks=\$080).

With these registers set up, the cog has all of the timing information it needs to automatically output a stream of data via selected output pins. But what about the data that needs to be moved? The data will come from a RAM buffer. The Propeller has 32 KB of RAM for system use. This includes variable storage, program storage, and stack space, so you have only a fraction for video data. Just how much is necessary and how does it all fit together? From the VCFG and VSCL registers, you have defined a single scan line as having 256 pixels. (Actually, 256 colored pixels is beyond the bandwidth of NTSC. But let's not worry about that right now.) Each pixel will require 2 bits of data to determine which color (or shade of gray) will be displayed at that pixel location. This will require 512 bits of data per line (i.e., 256 pixels/line × 2 bits/pixel). A field has 262.5 scans lines that make up one screen scan. The first and last few are usually out of the field

Bits	Video scale register (VSCL)
31:20	N/A
19:12	PixelClocks (Number of CTRA PLL clocks/pixel) \$00-FF 8-bit value
11:0	FrameClocks (16 or 32 × PixelClocks) \$000-FFF 12-bit value

Table 2—This 32-bit register contains an 8-bit count of clocks per pixel and a 12-bit count of clocks per frame (1- or 2-bit resolution-dependent).

Accelerate your Design Time with EmbeddedDeveloper.com



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

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

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

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

HEARST *business media*
ELECTRONICS GROUP

EMBEDDEDDEVELOPER.COM
FIND. COMPARE. BUY.

Convergence
PROMOTIONS

of view, leaving about 244 maximum viewable lines. Gamers will limit themselves to approximately 200 lines to be sure their environment is not chopped off at the top or bottom. So, at 200 scan lines, you need 12.8 KB of data space (512 bits/scan line \times 200 lines/8 bits). In many circumstances, you would like to use double buffering, which means two equal size video buffers. That would mean 25.6 KB of RAM. That sure doesn't leave much room for the application.

TILING

Earlier I mentioned that there are 256 character graphics stored in ROM. Each character is made up of 16 bits (horizontally) \times 32 bits (vertically). Two characters are combined to make use of the 32-bit double word data format. When the even or odd bytes of horizontal data are displayed on 32 separate scan lines, a picture of that character appears on the screen. If you want to print a character to the screen, you need a single byte to define the chosen character. The application doesn't need to figure out what data is required to form a character on the screen. All that is waiting for you to access it via a ROM address. This might require 1 byte pointer instead of 64 bytes of RAM (i.e., $16 \times 32 = 512$ bits).

Similarly, you can create special characters called tiles. A tile is used to create a background. You can think of a tile as a piece of a puzzle. The puzzle (picture) is a grid of horizontal and vertical positions on the screen where these pieces may be placed. The number of horizontal and vertical positions depends on the screen and tile resolutions. For instance, if each tile is 8 pixels \times 8 pixels and the screen is 256 pixels \times 200 pixels, then you can fit 32 tiles horizontally (256 pixels per scan line/eight pixels per tile) and 25 tiles vertically (200 lines/eight lines per tile). The tile bitmap of an 8 pixel \times 8 pixel tile would require 16 bits (eight pixels \times two color bits) per row times eight rows or 128 bits (four double words). For a gamer, the screen might be a bird's eye view of a maze. The entire picture could be drawn using only two tiles: a wall tile and a floor tile. Various mazes could be displayed by rearranging the two tiles in different patterns. Again, this reduces the amount of work associated with computing and storing a screen of information.

You can use tiles to dynamically change the way a screen is displayed; however, another object has been developed to operate in a more useful way. It is a sprite. While a tile and a sprite may have the same dimensions (and pattern), the latter has the ability to be placed anywhere on the screen and not just at the grid locations of tiles. One color of a sprite's pixel pattern is used as a transparent indicator that can let any tile color show through. The sprite can be magnified to become a multiple of its original size. And, most importantly, it has a depth associated with it that enables it to pass in front of or behind other sprites, creating a 3-D effect. While all of this is based on the tile/sprite generator written to run within a cog, this and many other Spin drivers written by the Parallax folks and other contributors like Andre LaMothe are available at www.parallax.com.

A BIT OF COLOR

If you have seen black and white TV, you may have been able to visualize color because your brain can take your

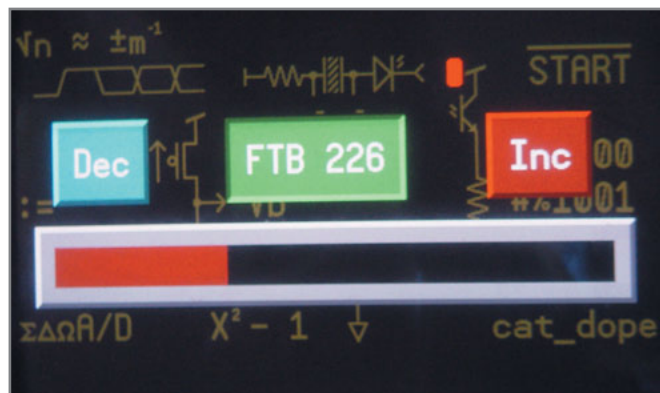


Photo 3—This photo of my VGA monitor's screen shows the simple display of three buttons with a horizontal gauge on a background of random characters and graphics. The red spot is the mouse cursor used to select a button. The buttons change a variable whose value determines the gauge's length. This data could come from an internal cog running a sampling application or from an external processor using the Propeller strictly as a display device.

real experiences and alter those sights broadcast in black and white. It was an extraordinary engineering feat to add color information to the standard black and white transmission signal without negatively affecting all the existing black and white receivers. A color sync signal hidden in the horizontal blanking portion of each scan line is disregarded by black and white sets, as is the modulated (and phase-shifted) color burst cycles during the active portion of each scan line. The average value left (of the color-modulated signal) remains as luminance levels for the black and white monitor producing levels of gray (between a black 0.25 V and white 1 V). A color monitor uses the phase difference between the color sync and the modulated color signal in the active portion of the scan line to determine color and the amplitude of the modulated signal to determine color saturation (or intensity). You saw earlier that the two video registers can configure the hardware to produce a composite video output signal containing all of the necessary syncs and modulations to do both black and white and color signal streams.

A cog driver can actually produce a signal with the full gamut of color. However, we've determined that it requires a lot of RAM to hold high-resolution color information. Because of the required RAM limits, RAM is conserved by limiting the resolution to 1 or 2 bits per pixel. One bit gives you black and white (or two colors). Two bits gives a few more colors to choose from, but how does this value relate to a specific color?

The grid that makes up the screen tile locations horizontal (rows) by vertical (columns) has a tile pointer map (TPM) associated with it. There is a 16-bit pointer for each tile location. Each pointer has double duty. The top 6 bits are an index into the tile color set table (TCT). The bottom 10 bits are an index into the tile bitmap memory (TBM). The tile color set table is a list of 64 4-byte entries. Each TCT entry holds the 8-bit color information for each of the four potential colors. Thus, this tile could choose to use any of the 64 color sets. The 10-bit

tile bitmap memory index is used as the upper 10 bits of a 16-bit address that holds all of the tile's bitmapped data.

GRAPHICS

While I consider characters, tiles, and sprites to be graphical in nature, the graphics engine driver uses point plotting to draw lines and polygons. You may be familiar with Cartesian coordinates, where x as a horizontal offset and y is a vertical offset from 0 in the center. Offsets increase when moving up and to the right, while they decrease when moving down and to the left of center. Any point P consists of an x and a y offset from 0. It is usually written as $P(x,y)$. Note that the video screen is usually mapped with an inverted y -axis so that $P(0,0)$ is the upper-left corner of the screen and $P(\text{screen_width}, \text{screen_height})$ is the lower-right corner of the screen. While the Cartesian coordinate system eases rendering, it is labor-intensive when dealing with rotations. Therefore, you may find the polar coordinate system more efficient when you must deal with rotational movements even though you will need to convert back and forth.

VGA

Up to this point, I've been discussing the power of the Propeller to work with NTSC video output. VGA video is actually less demanding than NTSC because the sync signal is digital in nature and separated from the color information. The color information is broken down into the three primary colors, and each has its own signal. The pixel clock is internally generated by the VGA monitor and will support 640×480 pixels. The VGA output consists of separated horizontal and vertical syncs plus separate R, G, and B analog outputs. Resister DACs can be used for each color similar to the DAC used for NTSC. Whereas the NTSC output requires 3 to 4 bits, the VGA output requires 8 bits. As for the VSU, it doesn't care which monitor is connected on the outside, as long as the timing configured into the video configuration registers is correct for the monitor type. **Photo 3** is an old VGA monitor (DB15 connection) I had connected to a Linux system here in the shop. It shows what can be done with just the Propeller demo board (or Hydra game console). If you do the math on 640×480 , you'll find that there isn't anywhere near enough RAM for this resolution, even at only 1 bit/pixel. However, by using 32×16 tiling, the RAM requirements are minimal.

I used the embedded character set to design a three-button screen with a linear bar graph. A mouse input enables button pushing, which in turn increases or decreases a variable that controls the length of the bar graph. The center button selects alternate color sets for the bar inside the graphs frame. Think of the variable associated with the bar's length as data coming through the USB port or other alternative connections, such as SPI, I²C, TTL serial, or a parallel port controlled by a spare cog.


While most of the examples in LaMothe's book use a composite NTSC output, you can find enough stuff inside about VGA to get started experimenting with some useful outputs. You can certainly start writing your own drivers for another microcontroller, but Propeller has some good things

going for it. With the internal PLL and an external 5-MHz crystal, the Propeller can clock at up to 80 MHz. With eight cogs, it's like having eight programs executing in parallel. Each cog has its own cog and also has its own VSU that makes outputting streaming video or audio a snap. While the resolution might be limited by the RAM available, the Propeller makes transitioning from NTSC to VGA a simple matter of software.

FARE THEE WELL

Our broadcasting buddy NTSC brought us closer to our world. We've seen world disasters, war, and poverty, as well as disaster relief, the Olympic games, and men landing on the moon. Thanks to NTSC, we've experienced positive and negative events together as one world. Digital broadcasting won't improve the standard of living for those in need, but it can carry on the tradition of NTSC by helping us understand more clearly (in HD) that we are no better than the least of our brothers.

And so we say goodbye to NTSC. It's been good knowing ya!

If you check your endangered species list, you might find that the CRT is hovering around the top. Not many bulky lead-shielded glass tube TVs (or computer monitors) are being manufactured. This is a case of less is more. LCDs have less weight and require fewer watts. Although we might see a continuing variety of interfacing connectors, for now all conform (for the most part) to the all-encompassing VGA (UXGA covers 1080p) standard. Who would've thought we'd have access to streaming TV programming via a cell phone? Cartoonist Chester Gould gave Dick Tracy the first two-way wrist radio in 1946, thanks to Al Gross's work on the walkie-talkie.^[2] 

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

REFERENCES

- [1] D. Lancaster, "TV Typewriter," *Radio Electronics*, Gernsback Publications, New York, NY, 1973.
- [2] Winnipeg Free Press, "Born Too Soon," 2001, www.comsoc.org/socstr/org/operation/awards/assocpress.html.

RESOURCE

A. LaMothe, *Game Programming for the Propeller-Powered HYDRA*, www.parallax.com, www.xgamestation.com.

SOURCES

HYDRA Development kit
Nurve Networks | www.xgamestation.com

Propeller
Parallax | www.parallax.com



FAT File System Review (Part 2)

C Code for the File System

Now that you're familiar with the FAT file system, George approaches the subject from a different angle. By presenting you with a general-purpose FAT file system, he challenges you to remove the features you don't need and customize your application.

In "DIY Signal Generation," Neal Martini writes: "I have been using C to program my past few projects. It was a little painful coming up to speed, but now I am a real convert." (*Circuit Cellar* 219, 2008) He went on to explain how C helped.

Isn't it great that everyone is reading my articles about the C programming language and converting? Actually, I don't believe my work has had much effect just yet. If you look at the articles in *Circuit Cellar* closely, you will see a movement toward structured design. This can be demonstrated in two ways: one, with the use of structured languages (e.g., C) and, two, in the flowcharts. Years ago, assembly language was king, and convoluted flowcharts that looked like plates of spaghetti were the norm. Today, designs are much more refined (structured). So quit hiding your head in the sand. Bite the bullet, take that plunge, move toward the light, and step over to using C in your embedded systems. (I like to see how many clichés I can get past my editors.)

Last time, I started a discussion about using a removable memory card (either CompactFlash or Serial Digital) in your next system and a file system on those devices. These cards offer a large capacity at a low price. At some point in your system analysis, you might decide that a file system is the best way to manage all the data on the card. If that's the case, then here is another solution to add to your bag of tools.

I discussed the low-level routines used to access and control these memory devices. And I also presented one solution to the FAT file system design I found in the *Circuit Cellar* archives. (Refer to the following two articles: "PIC a CompactFlash

Card," [M. Samuels, *Circuit Cellar Online* 127, 2001] and "Portable FAT Library for MCU Applications," [I. Sham, W. Hue, and P. Rizun, *Circuit Cellar* 176, 2005]). You'll find the original C code and the analysis output that has been generated using Source Publisher (www.scitools.com). The original code has documentation dated June 2004. Also I would like to point you to an Intel reference design "Intel IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor: Using CompactFlash." Don't let the title scare you. It's a good article about interfacing to a CompactFlash card and the code to support the interface. It's written from a hardware point of view more than a software (file system) point of view.

I also discussed the support that the C language has for file operations. I did not go into much depth about C file I/O support. That could be another multipart article series. If you've never written such routines, I recommend that you download a free (as in free beer) copy of Turbo C++ (or any other C environment). You need to work with these procedures. Keep in mind that Borland did a good job of protecting the designer from himself (or herself). For example, if you open a file and then exit the program, Turbo C++ takes care of the details and closes the file so that your disk does not become corrupted. But, if the power fails on your PC while a file is open, look out. You need to design for these issues. Actually, this is a good test to see if your program can recover the file(s) that were open when the lights went out. I recommend CTL-ALT-DEL as a gentler method to crash the Turbo C++ program than switching off the power.

Jeff Bachiochi recently presented a two-part article series about connecting an SD card to a PIC

processor ("Access SD Memory Cards," Parts 1/2, *Circuit Cellar* 221/222, 2009). Most of the articles I've come across don't give you much information, detail, or control over the FAT file system. This is not a criticism, it is just an observation. As practical embedded system designers (geeks), we're not going to implement only what's needed to get the job done. We're likely to probe, kick, stretch, test, and add features that improve and enhance the design, even if only implemented for use during in-house testing. We also like to get "under the hood" and understand the inner-workings.

With that in mind, I'd like to present a more general-purpose FAT file system. You can remove the features you won't use and tune the operation to suit your application and specific hardware. I purchased a low-power Altera NIOS evaluation system for another project. On that board is an interface to an SD card along with the embedded file system library (EFSL). This library is available for download from SourceForge, as are other interesting hardware and software projects. If you haven't looked at the site yet, be careful. It's full of great projects and you could spend mega-hours wandering through them.

GET STARTED

The EFSL is written in C and is only a library. It doesn't do anything out of the box but will tie in nicely with your embedded C code. It has hardware targets for Linux, the Atmel ATmega128, and the Texas Instruments DSP families. In a future article, I'll get this up and running on another embedded system and make that available to you. The download package includes a manual that explains the design and how to apply the library to your specific application. This library handles FAT12, FAT16, and FAT32, with long file names and time and date support.

The EFSL has the structure shown in Figure 1. The designers describe this as a linear object model. And you perhaps thought objects were used only in object-oriented languages such as C++ or Visual Basic. That is not so. You can use objects in regular old C. This is a good way to help you transition into those object-oriented designs and languages. Continue reading for a description of the EFSL.

FILE & FILESYSTEM OBJECTS

The File and Filesystem objects in the EFSL design deal with handling file system-specific details. The module `file.c` contains functions dealing with files such as `fopen()`, `fread()`, and `fwrite()`. The module `fs.c` contains general file system functions supported by the functions of `dir.c` and `fat.c`. Note that `file.c` uses these functions heavily.

The Partition object is responsible for translating partition relative addressing into disc-based LBA addressing. The functions in the Partition object are partition-specific. These include searching FAT-type partitions and read/write functions to partitions. They are found in the module `partition.c`.

The Disc object holds the partition table and has a direct link to a cache manager. The `disc.c` file contains the functions regarding the entire disc, such as loading the Master

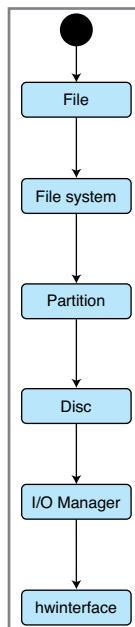


Figure 1—This is the EFSL structure.

Boot Record (MBR) and performing read/write tests.

The IOMan[ager] object receives all requests for sectors. This enables it to make smart decisions regarding caching. You will find these routines in `ioman.c`. One key define in this file is `IOMAN_NUMBUFFER`. It determines how many sectors IOMan can cache. I find that the IOManager holds several clever solutions that apply to most embedded designs. It supports delayed write. So as you fill up buffers with data, you can either force a write or let the IOManager decide when a write is necessary. IOManager also can allocate memory itself or use memory that you have previously allocated. This is key to being flexible enough to make it applicable to a broad variety of systems.

The `hwInterface` (hardware interface) object has three responsibilities: initialize the hardware, read the sectors, and write the sectors. In the first part of this article series, I presented code outlines for interfacing to a CompactFlash card and included those routines.

As you study this design, I think you will find it straightforward yet sophisticated enough to handle your embedded system requirements. I was looking for FAT32 support because I predicted that smaller capacity memory cards will soon disappear from production or their prices will rise above the larger capacity designs.

The text in the EFSL manual reads: "This library is released under the Lesser General Public License (LGPL). This means that you may use the library and its source code for any purpose you want. You may link with it and use it commercially. But ANY change to the code must be released under the same license." (<http://efsl.be>)

If I look up more details on the LGPL, it seems that if you keep the EFSL as a separate library and do not merge it with your code, then your code does not become part of the LGPL license and you can keep it confidential. However, if you mix the EFSL with your source code, the LGPL governs the entire package. I am not a lawyer. You should get proper legal advice if you proceed down this path. Still, I'm convinced the LGPL can be made to work in our competitive embedded world.

I recently used the Intel application note to create a FAT file system on a project. It worked well. There were a few areas that were a bit unpolished and my design was perhaps not the greatest. Specifically, I needed to open each file from a directory, one at a time, and read the data in that file. I stumbled my way through this. The code is presently used by the factory for system configuration purposes, and my solution was acceptable. In all fairness, this area of operation was not part of the Intel application note so this is not a criticism of that approach in any way.

Put in a more general way, the functions missing are functions like listing a directory—using standard C functions like `fopen()`, `fread()`, and `fwrite()`—and managing buffers. Also, you may be looking for commands like `mkdir()`, `rmdir()`, and `lsdir()` to make, delete, and list a directory. And consider how do you format memory cards? Well, it looks like the EFSL either provides these functions

directly or gives you the hooks need to implement them rather easily.

USING EFSL

How do you use the EFSL? I would create a directory to hold the library, make that directory part of your project, and then perform a compile and link. Remember: If you mix the EPFSL code with your code, then the entire work becomes covered by the LGPL.

To figure out what is in the EFSL, I put the library through a documentation program. I use Source Publisher from Scientific Toolworks. A compressed file (sp_EFSL.zip) is posted on the *Circuit Cellar* FTP site. The file unzips into a directory named "sp_EFSL." The directory includes the output from the Source Publisher program. The index.html file is the starting file. I ran Source Publisher on the EFSL library and enabled every feature. Perhaps I should be brought up on charges for excessive documentation, but I'm sure I can talk my way out of that one.

From the index, you can view the Navigation Menu, read the Understand Reports for the project (another utility offered from STI), view the Globals Used Report, view the Globals Report, view the code, and view the metrics. All of these reports contain hyperlinked references. So, if you are not sure of the definition of a constant, variable, or routine, one click will get you to that definition. If you haven't used one of these programs yet, this should impress you. If you are used to this richness of information, then just read on. Code review and SourcePublisher deserve a complete article. Perhaps you will write that one.

Starting with the config.h file, you can see the various definitions used in the project. The first is HW_ENDPOINT_NIOS. This sets the code up for compiling all that is needed to run on the NIOS evaluation board. You could add HW_ENDPOINT_CCI for this code to run on the *Circuit Cellar* evaluation board if there was one. (That's also another article.) Next, you'll see BYTE_ALIGNMENT commented out. This is for the Big-endian and Little-endian issues. Next are cache definitions. I'm glad to see this early on in the definitions. I bet memory allocation is on everyone's list of things to look out for. A bit further on

you'll find DATE_TIME_SUPPORT defined. Do you need to keep the file date and time up-to-date in our system? Well, here's the flag to do just that. I bet you'll have to hook into routines that provide a time structure from your hardware to this code. Notice that all these definitions come with well-written comments. A really nice piece of work. As you look through this code, why don't you offer to add more comments to clean up any unclear portions? It's an open-source community project.

In debug.h, you'll find definitions for adding debug functions to the code. Again, you can tailor this to your system requirements. In the other ".h" files, you'll find definitions that will look familiar if you've been reading my articles on C or you're already using C language. It's sort of like an old pair of shoes. It's comfortable and gives you the confidence to incorporate

this work into your project.

I don't mean to bore you with all these details, because you just want a prepackaged answer to your FAT file system issue. But you'll probably find that you need to tailor some portion of any design to fit your needs. And isn't it great to have such a complete solution?

LOOKING AHEAD

As I mentioned earlier, I plan to use this library in a project I'm working on. At that time, I'll report on just how fast some of these routines operate and the size of the generated code.

If there are any areas of the C language (short of homework assignments) you would like me to cover in more detail, just send me e-mail and I'll try to address your comments in an upcoming article. Remember: *Circuit Cellar* writers like me are doing this for you. ☐

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

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/226.

RESOURCES

G. Martin, "FAT File System (Part 1): Open Files and Perform Operations," *Circuit Cellar* 224, 2009.

———, "Structured Design (Part 1): An Introduction to Structured Techniques" *Circuit Cellar* 128, 2001.

Intel Corp., Application Note: Intel IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor: Using CompactFlash," 2004.

SourceForge, "Embedded Filesystems Library," <http://efsl.be/>.

Wikipedia, "Structured Systems Analysis and Design Method," http://en.wikipedia.org/wiki/Structured_Systems_Analysis_and_Design_Method.

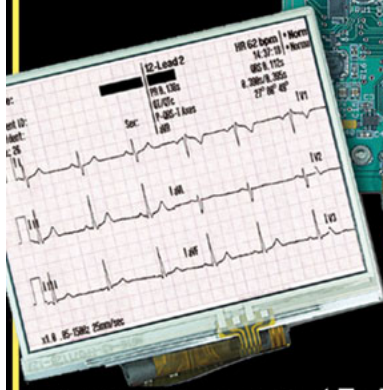
SOURCE

Source Publisher

Scientific Toolworks, Inc. | www.scitools.com/products/sourcepublisher

ezLCD- The "Smart Display" makes integrating a GUI ez!

- *Versatile Programming LCD Module
- *USB, SPI, RS232/TTL Interfaces
- *Bright 350 Nit LED Display (320 x 240)
- *Integrated Touch Screen
- *LUA Scripting Language Capable-
For stand alone embedded apps
- *Memory 3.8 MB + SD to 2G
- *ezLCD's are also available in 2.7",
5.6", 6.4", 8.0", 10.4"



**3.5" ezLCD
p/n: ezLCD+103**

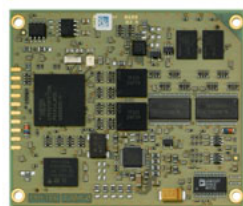
**Call for Custom
Display
Configurations**

www.earthlcd.com

PHYTEC

... Inside great products. Behind great ideas.

phyCORE® System on Module technology provides a production-ready hardware and software solution that enables product engineers to bring new ideas to market in the most timely and cost-efficient manner.



phyCORE-LPC3520 provides state-of-the-art power management, a Floating Point Unit, and rich peripherals such as USB OTG, Ethernet, and an integrated LCD controller. These features render the LPC3520 suitable for embedded deployment requiring high performance and integration crossed with low power consumption.

phyCORE-i.MX35 offers Graphics Acceleration, state-of-the-art power management, security architecture, a Floating Point Unit, and a rich peripheral set including CAN, DDR2 memory support and Ethernet that make this device an ideal candidate for a wide variety of embedded applications.



Design Services

PHYTEC offers hardware and software design services which can further reduce time-to-market and design risk. These engineering services can range from technical support, to design of target hardware and BSP adaptation, to complete turn-key solutions.

www.phytec.com | 800.278.9913 | www.phycore.com

CIRCUIT CELLAR

Designer's Notification Network

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

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



Find out more at www.circuitcellar.com/network.

EMBEDDED PROGRAMS

A Recipe for a Killer Embedded Application

Time Domain Reflectometry Explained

More on Programmable Robotics

Solar Data Logger Design

Event: Circuit Cellar "Digital Plus" launch

Date: March 23, 2009 – May 31, 2009

You hold in your hands a ticket to access Circuit Cellar magazine's new "Digital Plus" edition.

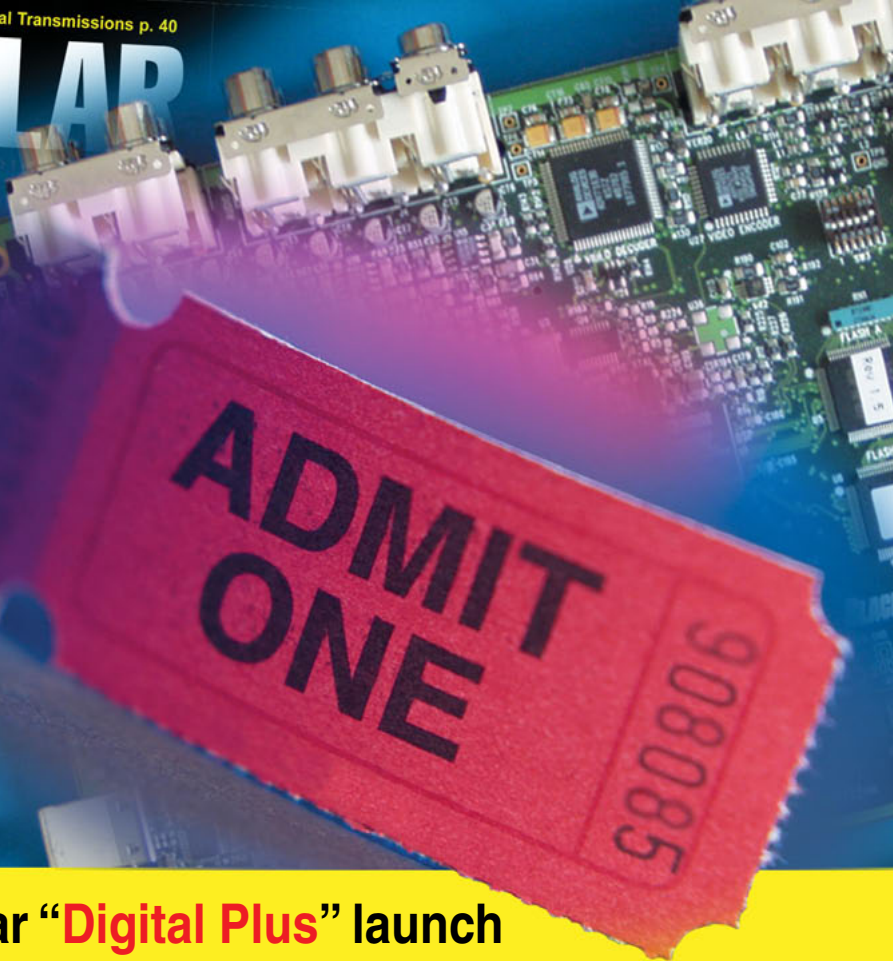
"Digital Plus" is your chance to:

- Read Circuit Cellar BONUS ARTICLES not available in print
- Enjoy audio/video-enhanced project articles
- Archive Circuit Cellar's print magazine as a PDF for your design library

Bring your friends to this limited-time public preview of Circuit Cellar's new venue. "Digital Plus" means more content, super-fast delivery of your favorite publication, and many additional features. As a current reader, you're entitled to special introductory pricing on "Digital Plus" subscription services.

Act now to receive the April 2009 preview bonus material, which launches with "Time-Triggered Systems: Co-Operative Schedulers 101," by Circuit Cellar feature author Michael Smith.

To access the preview edition, visit www.circuitcellar.com and click on the April issue icon on our home page.



CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

BONUS ARTICLE

Time-Triggered Systems (Part 1) Co-Operative Schedulers 101

In this series of articles, Mike takes a closer look at time-triggered systems and presents an interesting approach to using them.

Recently, an overseas internship student, Phillippe G., joined my group for an eight week research project. It was planned that he and a local music research professor would collaborate on using a new time frequency analysis algorithm called the ultra fast s transform for voice training. However, they needed to demonstrate an implementation of this DSP algorithm working in real time on a low cost processor before even thinking of proposing any commercial ideas to the algorithm's original inventors—two colleagues of mine at the University of Calgary, Canada. Over a period of about a month, many unusual, but reasonably musical, noises floated around my laboratory. Phillippe was moving fast down the road of getting the algorithm going on an Analog Devices Blackfin processor. This processor looked like a suitable target for this project because of its DSP capabilities and ability to

support both audio and video peripherals. However, about six weeks into the eight week project, the algorithm was working well, but the resolution of the TV display was not satisfactory. I offered my BF548 evaluation board, which has expanded on the basic BF533 evaluation board, to Phillippe. I admit not being able to find specific connection type of flash, that type, but there was a hard disk, a color LCD, a UART, Ethernet, and a USB port.

The BF548 processor has the same basic core as the BF533, so moving Phillippe's existing audio and video system to the new board was straightforward. Then I was able to merge this code with the existing LCD video interrupt driven screen demonstration code. This gave him a grayscale display of the audio data. In the time he had remaining, Phillippe was able to get a fully functional color display running. It would initially work, and then, over a period of 4 minutes, color synchronization would be lost, and the display would then rotate between red and blue color casts. We concluded that there was an occasional race condition between the audio interrupts updating a number of buffers, all the various memory DMA transfers occurring on those buffers, and the video driver

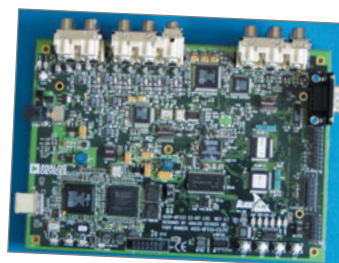
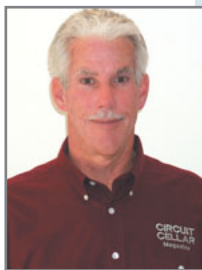


Photo 1—The Analog Devices BF533 evaluation board is capable of supporting basic audio and video demonstrations.



Whistle While You Work

A Look at a Modern DSP

Remember your father's DSP? Well, the Texas Instruments Piccolo isn't it. Tom presents this "DSP in disguise" and describes how you can use it to handle signal-processing applications such as switcher power supplies and other handy applications.

Labels and acronyms have always been part of the silicon game. Shorthand can be helpful, but sometimes it can be misleading, especially as the underlying technology changes over time.

RISC and CISC are examples of labels whose meanings have devolved to the point of meaninglessness. Yes, when originally coined many years ago, the terms clearly defined distinct architectures. But over time, as each camp adopted the best features of the other, the differences have become blurred to the point that what a chip is called has little to do with its "instruction set complexity."

"Microcontroller" (i.e., MCU) and "microprocessor" (i.e., MPU) are other schizophrenic labels, though still meaningful at the extremes. For example, an 8051 is clearly an MCU, while the 'x86 under the hood of your PC is clearly an MPU. But in the middle are a vast array of parts with aspects of both MCUs (e.g., on-chip flash memory and I/O) and MPUs (e.g., external bus).

This month, let's contemplate another acronym de jour, "DSP." The term itself has semantic ambiguity. After all, don't MCUs and MPUs process digital signals? Historically, DSPs were differentiated by their multiple busses and high-speed math capabilities, as typified by the classic MAC (multiply and accumulate) operation at the heart of signal-processing (e.g., filter) inner loops. But these days, virtually every 32-bit MCU or MPU has a measure of those capabilities as well.

Indeed, if anything, the trend has seen classic DSPs on the defensive. For instance, long-time *Circuit Cellar* contributor Professor Michael Smith wrote an article titled "To DSP or Not To DSP: Will a RISC Chip Do It Better?" in *Circuit Cellar* 28. That was way back in 1992!

But DSPs aren't dead. They've just been hiding, retro-marketed with new labels such as Digital Signal Controller. Along the way DSP suppliers have diligently worked to overcome historic DSP objections: high-price, high power consumption, hard-to-program, needs extra chips and glue logic, and more.

Let's put all the labels and preconceptions aside and take an impartial look at a modern DSP in disguise, the Piccolo from Texas Instruments. Yes, it's a natural for classic DSP apps such as fancy (e.g., sensorless motor controls and smart (e.g., power factor correction) switcher power supplies. But I think you'll be surprised to see the potential Piccolo offers general-purpose applications as well. In many, many respects it

definitely isn't your father's DSP.

PIPE DREAM

Piccolo comprises a family of parts with roots in the venerable Texas Instruments 'C2000 DSP line-up (see Figure 1). Nevertheless, from 50,000', you'd be hard-pressed to tell the difference between Piccolo and a traditional 32-bit MCU (see Figure 2). Notably, it's got on-chip memory, peripherals, and glue logic and is fully capable of stand-alone, single-chip operation.

But dive down to treetop level and the DSP difference becomes more apparent. The processing core itself, with an eight-stage pipeline, is more complicated than the three-to-five-stage unit you'll find on a typical 32-bit MCU. It's arguably a bit of architectural

overkill because the slowest Piccolo runs at blue-collar 40 MHz, but makes more sense when you realize Piccolo should, and does, maintain a measure of compatibility (i.e., assembly source) with higher-end 'C2000 parts that run at hundreds of megahertz.

Longer pipelines can be more hazardous—for example, when one instruction tries to read an operand not yet written by the preceding one, and Piccolo is no exception. However, the TI design features a measure of hardware interlocking that will keep you out of trouble. Ideally, you, or more likely the C compiler, will schedule instructions to avoid hazards; but if not, the pipeline will automatically stall. Notably, this does not incur the code bloat of

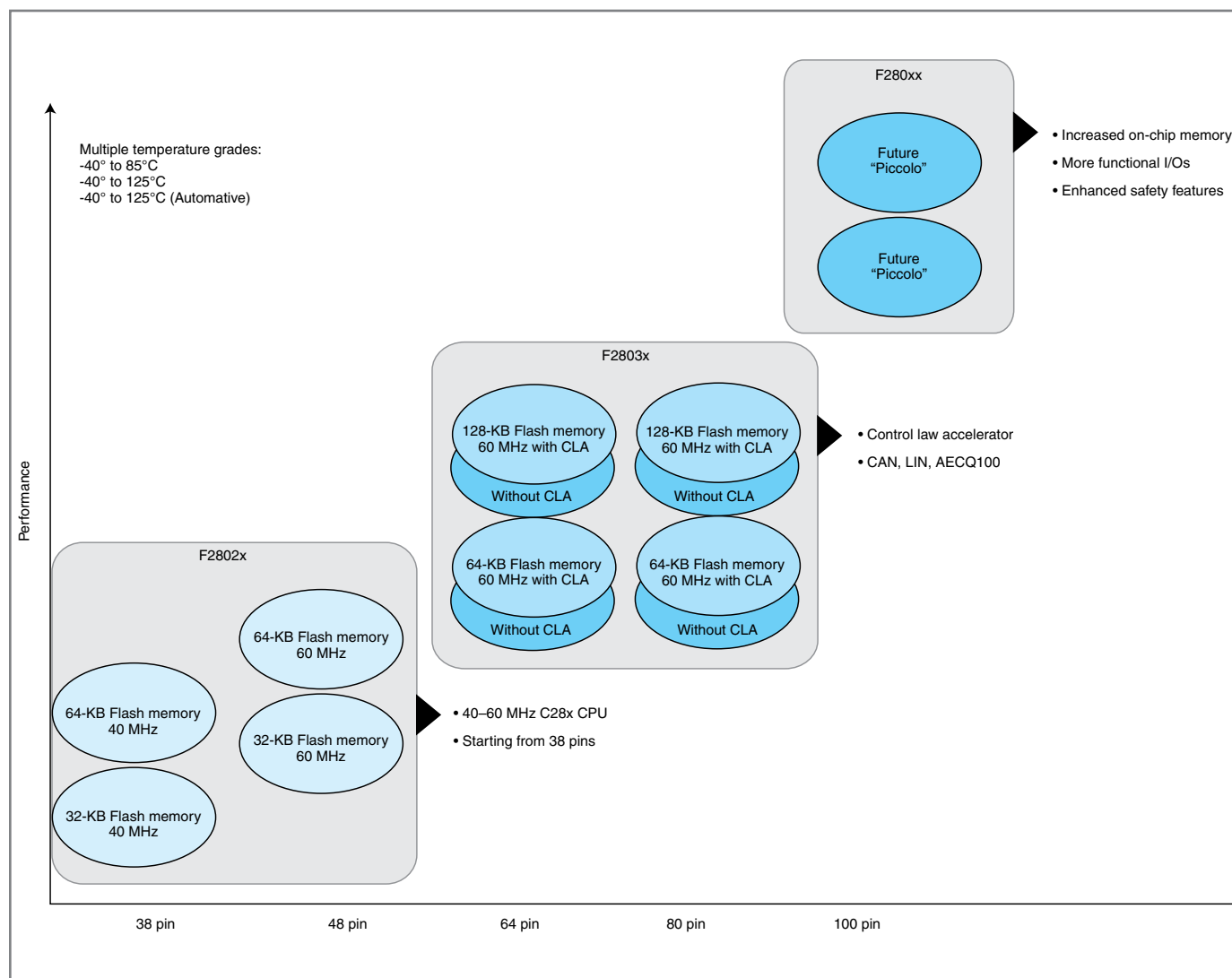


Figure 1—The TMS320F280xx Piccolo family is the latest addition to TI's venerable line of C2000 DSPs (er, make that MCUs).

delay slots (i.e., NOP insertions) that purely software-scheduled pipelines incur.

Piccolo features an interesting mix of architectural styles, as seen in the layout of the main registers (see Figure 3). The XT, P, and ACC registers do the DSP heavy lifting for the single-cycle 32×32 multiplier and barrel shifter. On the other hand, the 8×32 -bit XAR general-purpose registers lend a RISC-like feel. To streamline direct addressing, the 16-bit data pointer (DP) register fronts a 6-bit offset contained in the instruction to reach any location in the lower 4M words of data space. Of course, much of the big-iron addressing capability is underused in single-chip devices.

While Piccolo is a Harvard design capable of simultaneous instruction fetch, memory read, and memory write, access is made simpler by the fact memories are mapped into both

66

Instruction set-wise, Piccolo has something for everyone with RISC, CISC, and DSP all rolled into one.

99

program and data spaces. That's helpful because it makes it easy to access data stored in flash memory or run programs stored in RAM.

Instruction set-wise, Piccolo has something for everyone with RISC, CISC, and DSP all rolled into one. As noted above, the XAR registers support a measure of load/store processing, yet also can serve as pointers that allow instructions to operate directly on memory. The DSP DNA is apparent in all manner of number-crunching embellishments, such as saturation, rounding, and so on. But overall, Piccolo is quite CISCy, both as a matter of principle and to

maintain software compatibility with earlier 'C2000 parts.

EASY DOES IT

Historically, one of the knocks on DSPs was that they paid little attention to the details beyond their number-crunching mission, burdening a design (and the designer) with the need for extra peripheral chips, clock generation, multiple power supplies, and sundry glue logic. It's here that Piccolo stands out from its predecessors with features like an on-chip oscillator with PLL and missing clock detection, a single 3.3-V power supply with an on-chip regulator, power-on/brownout/watchdog RESET, and a vectored interrupt controller, all in tidy surface-mount packages. Unlike traditional DSPs, Piccolo is downsized with MCU-like 38-, 48-, 64-, and 80-pin package options. Peripheral-wise, there's a full complement comprising the usual suspects: GPIO pins (with an input glitch filtering feature), serial ports (UART, SPI, I²C—have it your way), and three general-purpose 32-bit timers. Piccolo's signal-centric aspirations are served by multichannel, high-speed (up to 4.6 Msps) 12-bit ADC with flexible triggering and auto-sequencing options.

These features are all well and good, but except for the formidable number crunching capability, they're otherwise little different than those found on a typical 32-bit flash memory MCU (which is, after all, the point). All else being equal, for those who aren't already in a committed relationship with Piccolo's C2000 predecessors, it would seem there's little compelling reason to switch.

But maybe all else isn't equal, considering the advanced I/O capabilities embodied in Piccolo's on-chip enhanced control peripherals, which include enhanced (ePWM) and high-resolution (HRPWM) PWMs, input capture (eCAP) and quadrature encoder

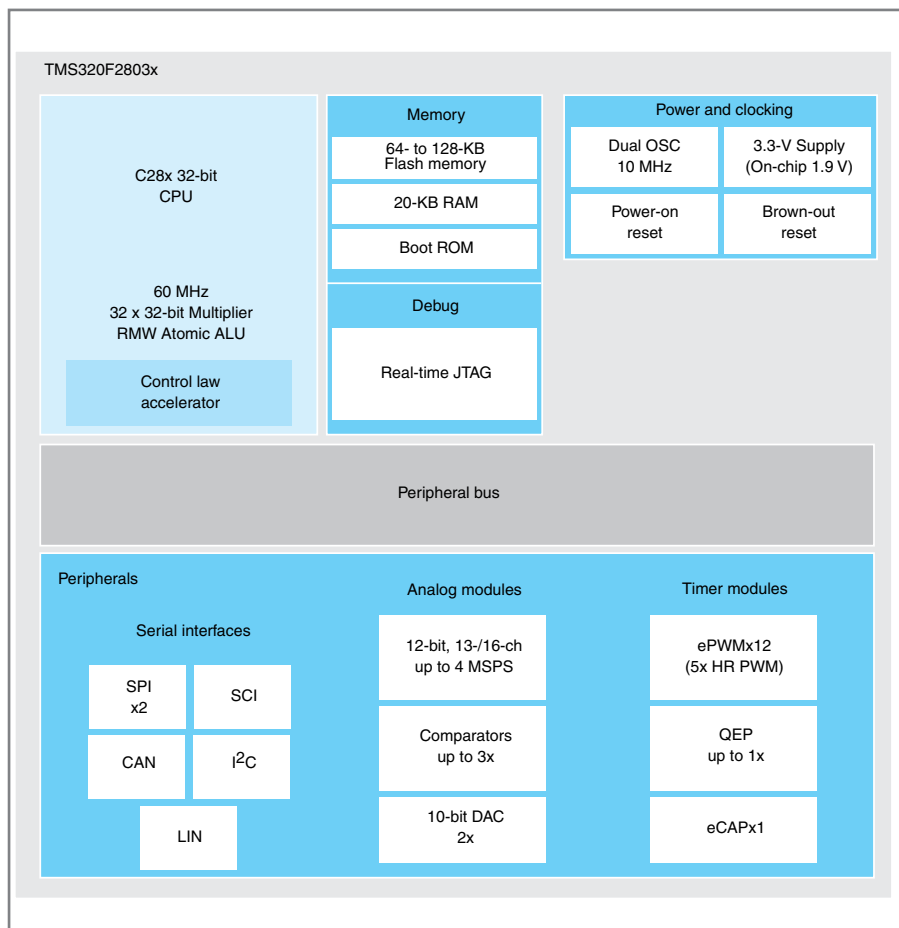


Figure 2—With on-chip flash memory and RAM, glue logic, and a full complement of I/O, Piccolo is a contender for low-cost single-chip applications.

(eQEP). Naturally, the enhanced I/O modules are a big plus for traditional DSP apps (e.g., motor control) but may find favor in other high-frequency, timing-centric applications. For example, using a fancy "Micro-Edge Positioning" technique, the HRPWM offers edge timing resolution on the order of 150 ps. Yes, that's "picoseconds" with a "p." Try that with your run-of-the-mill MCU.

Higher-end Piccolos will also include what TI calls a control law accelerator (CLA). I don't have specs

yet, but sifting through the press tea leaves reveals the CLA is an independent 32-bit floating-point coprocessor that autonomously runs control loops (e.g., PID). Able to communicate directly with peripherals (e.g., ADC, ePWM) and deal with interrupt requests, the CLA is said to significantly reduce overhead for the main processor (see Table 1).

SYMPHONY IN C

Of course, if you're already into DSPs, TI has you covered when it

comes to apps like motor control, smart power supplies, and such. They've got an effective development and prototyping regime comprising DIMM-like processor modules that plug into various application-specific motherboards and plenty of software, app notes, and more to go with. There's even something called DSP/BIOS, kind of a mini-me modular RTOS, comprising a library with hundreds of basic data acquisition, storage, and control functions.

But you don't need to be a rocket

No other PCB-design tool gives you more value per dollar

Boards designed under EAGLE are developed in one-man businesses or in large industrial companies. Most of the top companies are our customers. The crucial reason for selecting EAGLE is not usually the low price, but rather the high-end functionality along with the ease of use. And EAGLE users appreciate the outstanding level of support, which at CadSoft is always free of charge, and is available without restriction to every customer.

These are the real cost killers!

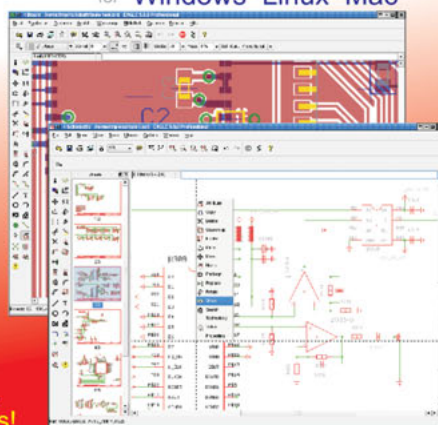
Version 5 is even easier to use, especially for beginners, due to an enhanced user interface.



EAGLE
Version 5

Schematic Capture • Board Layout
Autorouter

for Windows® Linux® Mac®



Version 5 Highlights

- ▶ Stand-alone schematic editor available.
- ▶ Automatic signal/contact cross references using frame coordinates.
- ▶ Right-mouse click for more consistent Windows UI.
- ▶ User-definable attributes for parts.
- ▶ Schematic sheet management.
- ▶ Hiding approved DRC and ERC errors.
- ▶ PRINT preview and text-searchable PDF output.
- ▶ Improved search engine for help.
- ▶ And much, much more.

Pick the level that is right for you — pay only the difference for upgrades!

	Light	Standard	Professional
Max. number of schematic sheets	1	99	999
Max. board size	4x3.2 inch	6.4x4 inch	64x64 inch
Max. # of signal layers	2	4	16
Layout or Schematic Editor		\$249	\$498
Layout and Schematic Editor		\$498	\$996
Layout Editor and Autorouter		\$498	\$996
Layout Editor and Schematic Editor and Autorouter	\$49	\$747	\$1494

Standard and Light Editions have full functionality except for the limitations mentioned in the table.

You can use EAGLE Light for evaluation and non-commercial applications without charge. Download it from our web site.

www.cadsoftusa.com
800-858-8355

CadSoft Computer, Inc., 19620 Pines Blvd., Suite 217, Pembroke Pines, FL 33029
Hotline (954) 237 0932, Fax (954) 237 0968, E-Mail: info@cadsoftusa.com
Windows / Linux / Mac are registered trademarks of Microsoft Corp. / Linux Torvalds / Apple Computer, Inc.

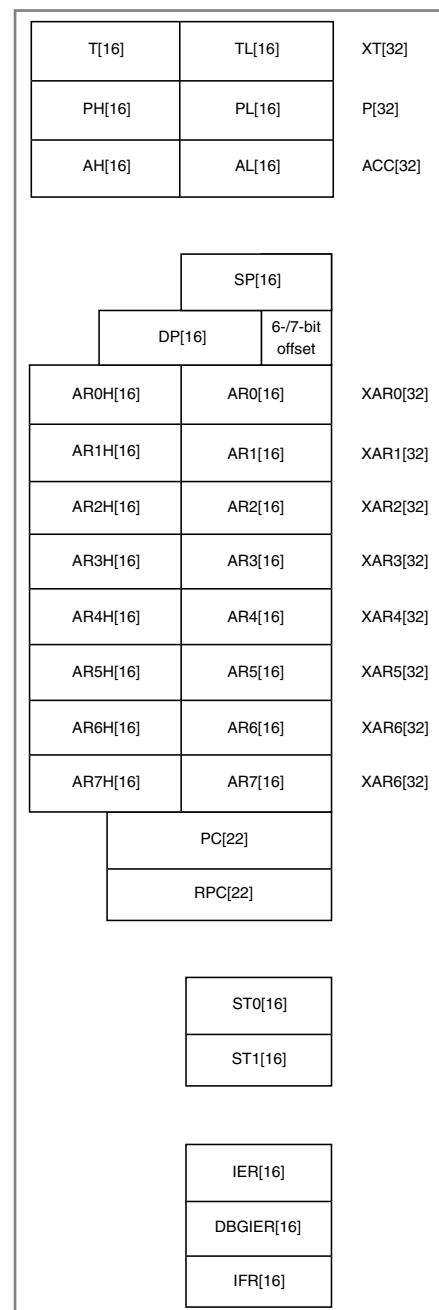


Figure 3—RISC, CISC, or DSP? The Piccolo architecture combines aspects of all three.

scientist, or have a budget like NASA, to kick the tires. Let's check out the Piccolo MCU controlSTICK, which definitely qualifies as an impulse buy at just \$39.

If you haven't caught on to the fact Piccolo can impersonate an MCU, [Photo 1](#) should make it clear. There, you can see that the controlSTICK comprises little more than the Piccolo itself along with a Future Technology Devices International chip to handle the USB interface. If it walks like an MCU and talks like an MCU?

The same goes for the Texas Instruments Code Composer Studio, which at least at first glance, looks like a typical MCU toolchain with a dizzying array of menus and windows to play with (see [Photo 2](#)). Indeed, if anything, Code Composer Studio takes it a step further with advanced analysis capabilities that leverage Piccolo's on-chip debug hardware (see [Photo 3](#)). Scratch a bit further under the surface and you'll find unique vestiges of Piccolo's DSP heritage. For example, you can certainly monitor data with the conventional Watch Window. But you can also capture data as a graph, and Code Composer Studio can even run it through an FFT for you (see [Photo 4](#)). Pretty cool.

I'll be frank and admit with the complexity of modern toolchains, kicking the tires is just that. It would take quite a while to get up to speed and actually test drive all of the advanced features. The best I can do is tell you I ran through some of the demo projects and everything worked as advertised.

As someone who has always been interested in computer architecture, within the thousands of pages of chip, tool, and application documentation, my attention was captured by the "Optimizing Your Code" chapter in the C/C++ compiler manual.^[1] I learned long ago that any discussion of an architecture's merits is moot unless compiler quality is factored in.

It was fun and interesting to see all the hoops the compiler jumps through to tweak your code. I mean

it's almost as though the compiler folks would just as soon get rid of the application programmer and do it all themselves.

There's a laundry list of dozens of

potential optimizations. Many of these are pretty obvious and old-school, such as dead-code removal (i.e., remove code that is not reachable) and common sub-expression

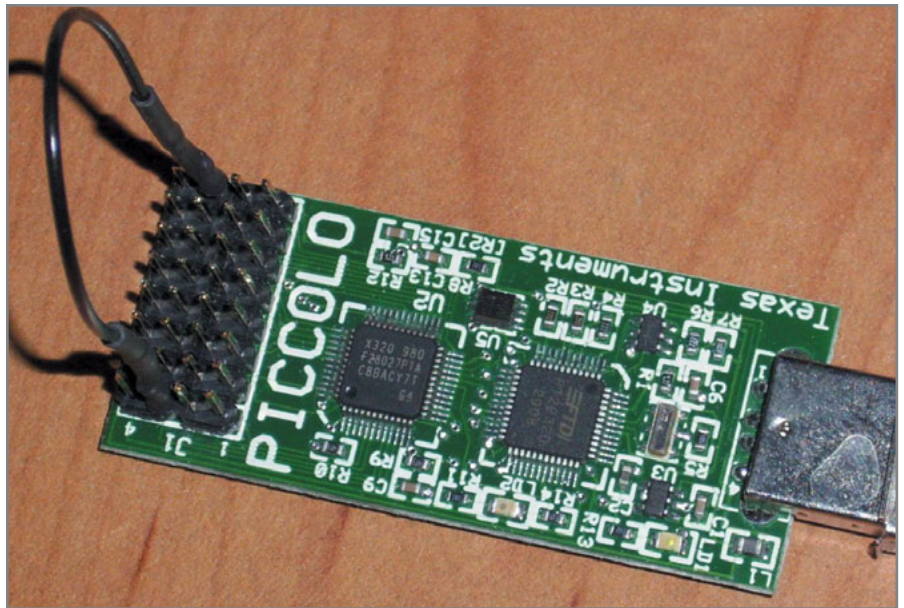


Photo 1—If it walks like an MCU and talks like an MCU? The MCU controlSTICK highlights the fact Piccolo (shown here along with a Future Technology Devices International chip that handles the USB interface) is well-suited for low-cost single-chip applications.

GENERAL CIRCUITS CO., LTD

CHINA PCB SUPPLIER

QUALITY PCB & SERVICE PROTOTYPE TO PRODUCTION

instant online quote

shopping cart ordering system

China competitive prices

free electrically test

web <http://www.pcbcart.com>

E-mail sales@pcbcart.com

Tel +86-571-87013819

Fax +86-571-87036705

Add No.76 GuCui Road, Hangzhou, China

WWW.PCBCART.COM

Operation	C28 (60 MHz)	C28/CLA (60 MHz)
Feedforward control cycles	482	482/0
Feedback control cycles	1,081	0/550
Total control Law cycles	1,563	482/550
Megahertz used (20-kHz loop)	32 MHz	10/11 MHz

Table 1—The control law accelerator (CLA) in higher-end Piccolos can handle closed-loop control by itself, freeing the main processor for other tasks. In this example provided by TI, the combination of the C28x processor core and the CLA is nearly three times faster than the processor core alone.

elimination (eliminates redundant calculations). And as you might imagine, there are a number of low-level instruction scheduling optimizations to keep that long pipeline from stalling while preserving the intent (e.g., ordering) of the programmer.

Actually, the DSP gurus have been at the front of the pack leading the charge to fancier compilers, and it shows in CCStudio with some truly Poindexter high-level optimizations. You're probably familiar with ones like loop unrolling (eliminates branches) and function inlining (ditto) but how about Alias Disambiguation?

C programmers love pointers, and C compiler writers hate them. Here's the "alias" dilemma. Compiler writers want the flexibility to move instructions around willy-nilly. And they could if it weren't for those darn data dependencies. For example, if the program is:

```
a = a1 * a2
...
b = b1 + b2
...
z = a + b
```

The first two statements can be moved around and even their order can be reversed. The only restriction is the last statement has to remain last because z's value depends on the prior setting of a and b.

Easy enough. But what if instead of referencing the variable by name, the programmer referenced it using a dynamically calculated pointer to (i.e., address of) the variable.

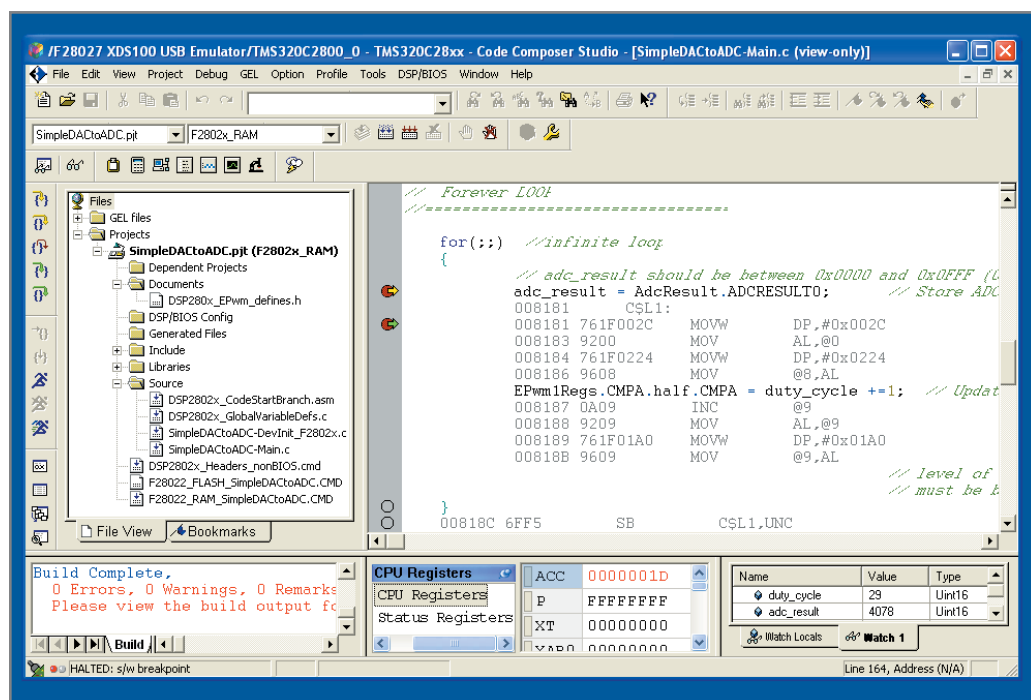
Photo 2—Programmers will feel right at home with Code Composer Studio, TI's full-featured C/C++ IDE for Piccolo.

Now the compiler has to try to establish the possible run-time values a pointer can, or can't, take on in order to guarantee dependencies aren't compromised. Two different pointers may point to the same variable (the "alias") and the compiler has to figure out whether that is, or isn't, possible (the "disambiguation").

I'm no expert, so it all seems like magic to me. I do remember discussing the subject of alias disambiguation with a compiler expert once. He told me there are ways to deal with the challenge theoretically, a minor caveat being possibly infinite compile time. An effective compromise is for you, the programmer, to give the compiler some hints. For instance, the TI compiler has "aliased_variable" options that enable you to tell the compiler that you're sure a particular pointer is safe from aliasing.

Just keep in mind that optimizers, especially ones that move instructions around, can make debugging more mind-numbingly complicated than it already is. Consider the common technique of viewing your compiled program as C source mixed with assembly language. The problem is a piece of the assembly language associated with a particular line of C code may be moved to a different location. The compiler has an interlist option that keeps the listing sane by restricting the optimizations.

Similarly, watch out if you mix inline ASM with your C code, especially if it messes with C variables, functions, and so on. For example, if your ASM code calls a C function, you may find the compiler didn't know that function was needed and optimized it away. The compiler offers a "FUNCTION_EXT_CALLED" pragma you can use to explicitly mark functions that should be preserved. There's also a "CALL ASSUMPTIONS" option that tells the compiler whether your ASM code does, or



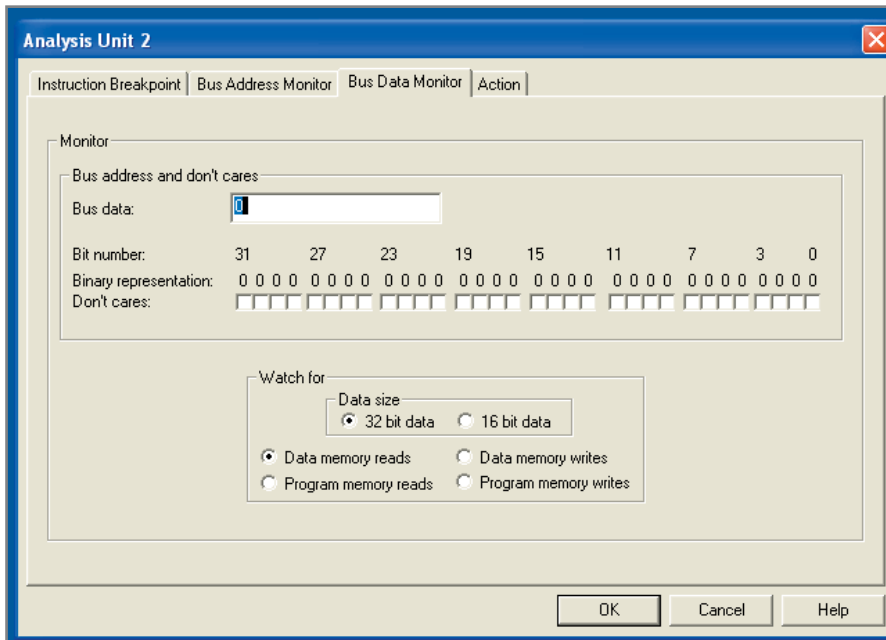


Photo 3—With a single-chip MCU, all the action is hidden inside behind the pins. Piccolo includes an on-chip bus analyzer that goes far beyond a simple “breakpoint” so you can see what’s going on.

doesn’t, call C functions or modify C variables.

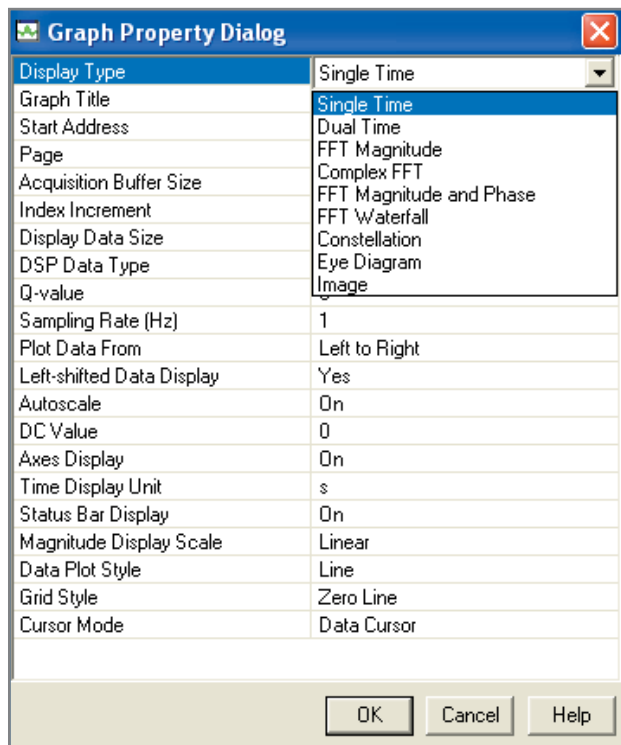
In short, debugging compiler optimized code is a pain, although you’re welcome to try. If you get really deep into it, CCStudio includes a pipeline simulator that may come in handy (see [Photo 5](#)).

THREE Ps

So how does Piccolo measure up to other 32-bit flash memory MCUs when it comes to the “three Ps”—performance, power, and price?

Of course, performance depends on the application being performed.

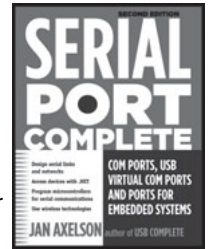
Photo 4—TI may not call Piccolo a “DSP,” but if you should happen to stumble across a signal, Code Composer Studio has got you covered with unique signal-centric debug features.



Clearly, Piccolo will excel in traditional DSP number-crunching applications, all the more true by taking advantage of the unique control law accelerator. But as well, there’s every reason to believe Piccolo can hold its own in general-purpose applications too. Indeed, TI claims Piccolo delivers

THE SERIAL PORT LIVES!

Everything you need to know about COM ports, USB virtual COM ports, & asynchronous serial ports for embedded systems.



Hardware & software for RS-232 & RS-485. Wireless options and more.

Serial Port Complete Second Edition

Jan Axelsson

ISBN 978-1-931448-06-2 \$39.95
Lakeview Research LLC www.Lvr.com

From the author of *USB Complete*

PIC-SERVO

MOTION CONTROL

MOTION CONTROLLERS FOR
BRUSH, BRUSHLESS AND
STEPPER MOTORS.

- controller chips
- controller boards

www.picservo.com
JEFFREY KERR, LLC

MACH64

PROGRAMMABLE LOGIC STARTER KIT

Based on the Lattice
ispMach 4064.

Includes 250
page lab manual.

Learn CPLDs
the fun way with
the MACH64! This complete kit
comes with everything you need to
take you from mystery to mastery
with CPLDs and programmable logic.
Learn to turn *software* into
hardware!

www.XGAMESTATION.COM

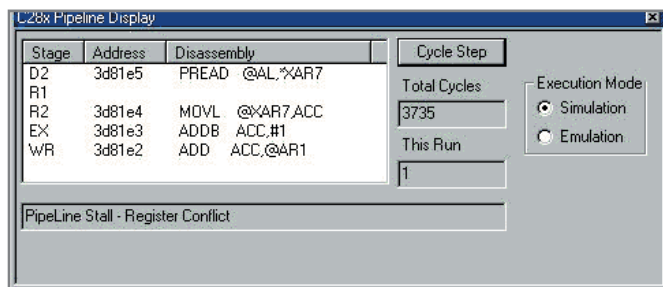


Photo 5—Code Composer Studio includes a Piccolo simulator that lets you see what's going on under the hood.

25% better Dhrystone performance than a Cortex-M3-based MCU.

Power consumption is equally system- and application-dependent. The specs

show active power consumption of about 2 mA/MHz, which is a little higher than generic 32-bit flash memory MCUs, but not bad considering

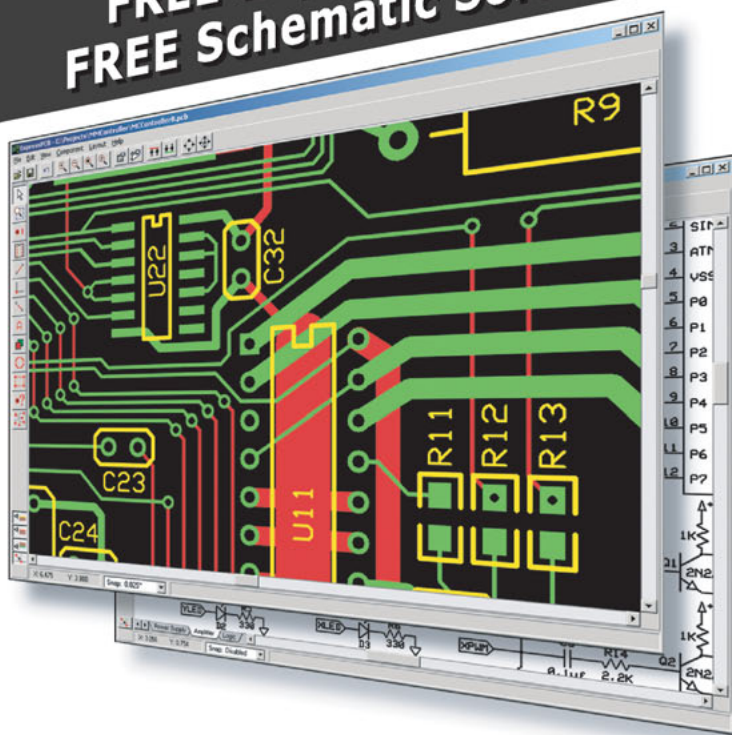
the horsepower on tap. Besides the clock rate, a key factor is which, if any, of the Piccolo peripherals aren't used and can be powered down. For example, turning off the CAN module saves 11 mA and in fact simply disabling the CPU clock output pin (XCLKOUT) saves a whopping 15 mA. Standby current consumption in the lowest power mode (i.e., everything off) is on the order of 100 μ A, which is somewhat higher than typical MCUs, but certainly not a showstopper. My conclusion is that the slightly higher Piccolo power consumption would only be an issue in the most battery-life-sensitive applications.

Finally, there is price. I don't have an official price quote, but the press release says Piccolo starts at less than \$2 in volume. In terms of historic pricing for DSPs, that's quite a bargain considering high-end parts (such as TI's own C6x line) can have triple-digit price tags. At the same time, we've all seen the headlines for \$1 chips from other 32-bit flash memory MCU suppliers.

There's no free lunch, and you get what you pay for. If you need only a plain-vanilla MCU that's probably what you should use. But if you can take advantage of even just one of its advanced features—notably the number-crunching capability or the advanced peripherals (e.g., HRPWM, CLA)—Piccolo may hit just the right note in your application. ■

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.

\$51^{For 3} PCBs
FREE Layout Software!
FREE Schematic Software!



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

expresspcb.com

REFERENCE

- [1] Texas Instruments, "TMS320C28x Optimizing C/C++ Compiler," 2007, <http://focus.ti.com/lit/ug/spru514c/spru514c.pdf>.

SOURCE

controlSTICK Evaluation kit and Piccolo microcontroller
 Texas Instruments, Inc. | www.ti.com



Saelig *UNIQUE PRODUCTS FREE SUPPORT!*

USB-Serial  A complete USB-serial converter in a DB9 shell. \$26	Fast 4-ch Scopes  4-ch 2 GSa/s DSO 5.7" TFT color LCD b/w to 200MHz.	I2C Xpress  Versatile USB 2.0 I2C protocol exerciser and analyzer.
USB Bus Analyzers  Packet-Master™ - USB 1.1/2.0 analyzers and generators. \$699+	Color LCD Scope  2-ch + trigger standalone USB bench scope. \$325 / \$599	Wireless Solutions  Analog input, bluetooth wireless modules 433/868/915MHz.
Keyboard Simulator  USB board adds 55 I/O & 5 x 10-bit A/D inputs, 1 x 10-bit analog O/P.	USB to I2C  "Drop-in" solution connects PC to I2C/SMBUS + 32 I/O lines. \$99	7 in 1 USB Scope  2-ch 10-bit 2MHz scope/spectrum-analyzer, 3MHz 8-bit wfm gen. \$180
.NET Board  Small (2.2" x 2.2") lowest cost .NET Micro Framework dev system.	CANxtra  .NET Micro box with CANbus, Ethernet, RS232, A/D, analog out	EMC Spectrum Analyzer  EMC RF & EMF Spectrum Analyzer 1Hz to 7GHz from \$782 / \$1875

Saelig *unique electronics* 1-888-772-3544 www.saelig.com


Ethernet to I2C

Around the office...
or around the World

MCC
Micro Computer Control

www.mcc-us.com



CIRCUIT CELLAR®

back issues available as



Searchable Archives on CD-ROM

CD-ROM #13 2008 Issues 210-221
 CD-ROM #12 2007 Issues 198-209
 CD-ROM #11 2006 Issues 186-197
 CD-ROM #10 2005 Issues 174-185

Order Online:
www.circuitcellar.com
 or call 860.875.2199

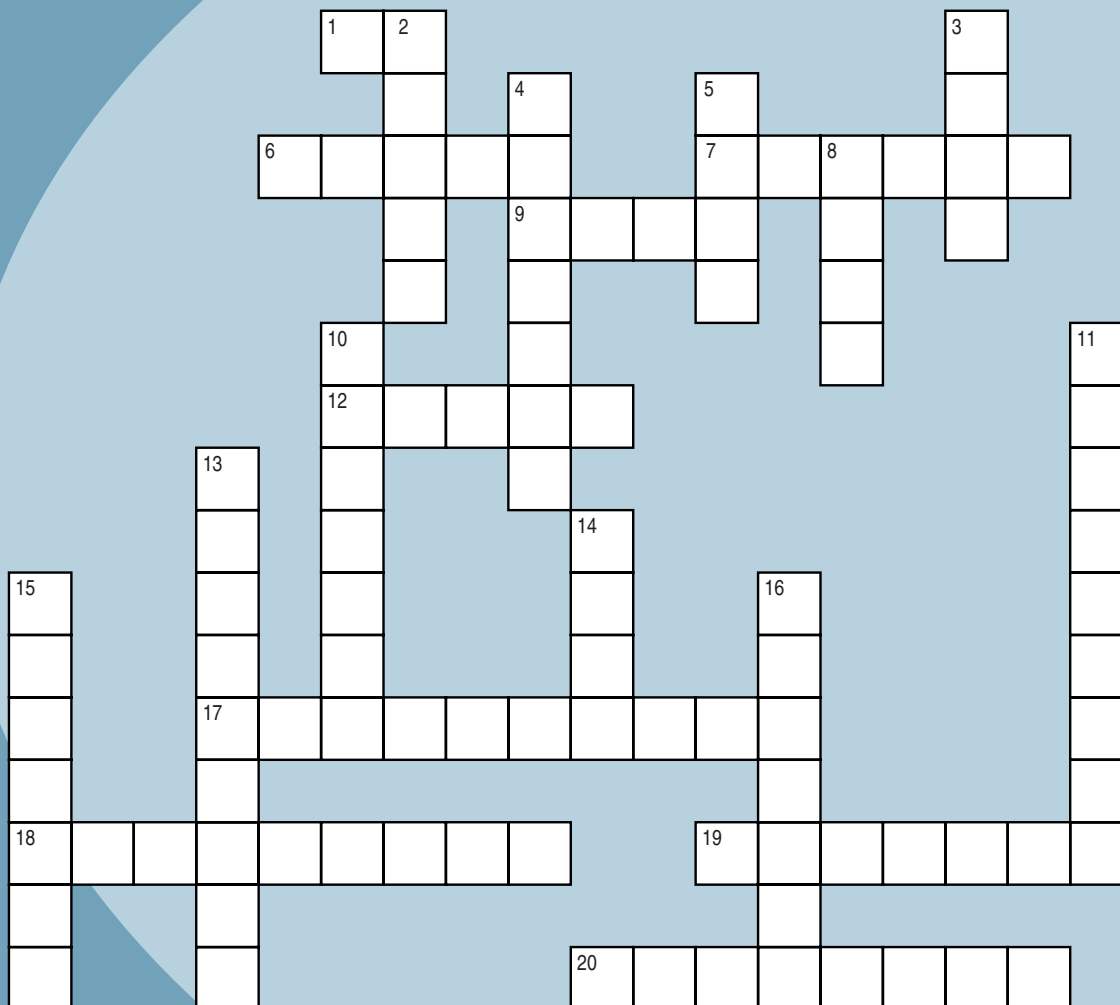
PROFESSORS

The Circuit Cellar college program puts quality engineering information in the hands of your students every month. Sign up now to get Circuit Cellar distributed to your class this semester.

To update your professor account or to find out more about our college program, visit www.circuitcellar.com/products/collegeprogram/

CROSSWORD



Down

2. An app's server
3. Fluid/air resistance
4. Mini notebook
5. Plus for loopback tests
8. Reduce OSC amplitude
10. Positively charged ions
11. In 1882, Edison's network provided 110 V to customers in this borough
13. Examine an older system, replace problem parts
14. 10^9
15. Unidirectional communication
16. Diode, 1904

Across

1. Weber
6. Explosion in a star
7. 57.2958 degrees
9. 10^{12}
12. Negatively charged ions
17. The "R" in $I = V/R$
18. cd/m^2
19. Si
20. 6.0221415×10^{23}


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

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. E-mail adcopy@circuitcellar.com with your file and digital submission or send it to IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066. For more information call Shannon Barraclough at (860) 875-2199.

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



Full Speed

It writes your USB Code!

NEW! HIDmaker FS for Full Speed FLASH PIC18F4550

Creates complete PC and Peripheral programs that talk to each other over USB. Ready to compile and run!

- Large data Reports
- 64,000 bytes/sec per Interface
- Easily creates devices with multiple Interfaces, even multiple Identities!
- Automatically does MULTITASKING
- Makes standard or special USB HID devices

NEW! "Developers Guide for USB HID Peripherals" shows you how to make devices for special requirements.

Both PC and Peripheral programs understand your data items (even odd sized ones), and give you convenient variables to handle them.

PIC18F Compilers: PICBASIC Pro, MPASM, C18, Hi-Tech C.

PIC16C Compilers: PICBASIC Pro, MPASM, Hi-Tech C, CCS C.

PC Compilers: Delphi, C++ Builder, Visual Basic 6.

HIDmaker FS Combo: Only \$599.95

DOWNLOAD the HIDmaker FS Test Drive today!

Trace
SYSTEMS, Inc.

www.TraceSystemsInc.com
301-262-0300

USB

Add USB to your next project—it's easier than you might think!

- USB-FIFO up to 8 mbps
- USB-UART up to 3 mbps
- USB/Microcontroller boards pre-programmed with firmware
- 2.4GHz ZigBee™ & 802.15.4
- RFID Reader/Writer

Absolutely NO driver software development required!

www.dlpdesign.com




ALL ELECTRONICS CORPORATION

Electronic and Electro-mechanical Devices, Parts and Supplies.
Wall Transformers, Alarms, Fuses, Relays, Opto Electronics, Knobs, Video Accessories, Sirens, Solder Accessories, Motors, Heat Sinks, Terminal Strips, L.E.D.S., Displays, Fans, Solar Cells, Buzzers, Batteries, Magnets, Cameras, Panel Meters, Switches, Speakers, Peltier Devices, and much more....

www.allelectronics.com
Free 96 page catalog
1-800-826-5432

I2C SPI 1 Wire



3 Separate Buses
5V & 3V
Simple ASCII Interface
Cross Platform - All OS

www.i2cchip.com

Flashlite 186



\$69
QTY 1

- 186 processor @ 33 MHz
- DOS w/ Flash File system
- 44 Digital I/O lines w/ CPLD
- Console / Debug Serial Port
- 7-34V DC or 5V DC power
- 2 Serial Ports
- Accepts 8MB DiskOnChip
- 2 16-bit Timers
- 512K DRAM & 512K Flash
- Watchdog Timer
- Expansion options with Peripheral Boards

\$99

Development System

Development kit includes:

- Flashlite 186 controller
- Borland C/C++ ver 4.52
- FREE Email Tech Support
- Serial Driver library
- AC Adapter and cable

Call 530-297-6073 Email sales@jkmicro.com
On the web at www.jkmicro.com

JK microsystems




Free Training

for Rabbit Developers

Learn More at
Rabbit-U.com

08078

Add a color touch interface to your embedded product!



- High level RS232 interface
- Low cost
- Easy to program
- In stock!

Add color graphics to any 8/16 bit embedded system.
Easy, fast and flexible. Up and running in hours!

REACH
TECHNOLOGY INC.

www.reachtech.com • 510-770-1417

842 Boggs Avenue • Fremont, CA 94539

World Leading Driver-Free Win/Mac/Linux USB Chips



A/D - I/O - SPI - I2C
BASIC programming
USB Status & Control

USB USB-DAQ

- USB-DAQ / FileSys sensing & logging
- USB-to-UART/I2C/SPI and I/O expanders
- No microcontroller programming required
- Add USB to your products in a day!

www.hexwax.com - Mouser - Farnell - Digikey

Amazing PIC programmer

Most devices supported
ICSP, SQTP, & copy limits

\$32 at Digikey & Mouser

www.flexipanel.com

Actual size - patents pending

LV- MaxSonar

Ultrasonic Ranging is EZ

LV-MaxSonar Products

- High quality • Low cost
- Low power, 3V-5.5V, (< 4mA avg.)
- Easy interfacing • Serial, pulse width, & analog voltage outputs
- Reliable and stable range measurement • No dead zone



LV-MaxSonar-EZ

- Choice of beam patterns
- Tiny size (<1 cubic inch)
- Light weight (<5 grams)

LV-MaxSonar-WR1 (IP67)

- Industrial packaging
- Weather resistant
- Standard 3/4" fitting
- Quality narrow beam

www.maxbotix.com

PCB Fab & Assembly Since 1997




Experts in Quick Total Turnkey Assembly at Low Prices!

- PCB Design, Fab & Assembly
- Cable Assembly
- Box Integration
- Functional Testing
- BGA
- Product Development
- ISO Certified



CapTron Corporation

301-869-6100
sales@captroncorp.com



CIRCUIT CELLAR

FOR COMPUTER APPLICATIONS

Make sure you're signed up to receive Circuit Cellar's monthly electronic newsletter. *News Notes* will keep you up to date on Circuit Cellar happenings. Stay in the loop!

Register now. It's fast. It's free.
www.circuitcellar.com/newsletter/

The \$69 PLC

Work as Stand-Alone Ladder Logic PLC.
Or as Smart Remote I/Os of PC/ PLCs.
RS485 allows 256 units to be networked.



E10-Relay+ RS485
Opto232
RS232 RS485

E10-npn+

Incredibly Easy to Program!
Our software is used by many colleges for teaching PLCs!

Get Free Ladder Logic Simulator:
www.tri-plc.com/cci.htm


Tel: 1-877-874-7527 - PLC specialist since 1993

TRI
Triangle Research

Adapt9S12XDP512 Modular Prototyping System

- * Robotics and Mechatronics
- * Electronic Fuel Injection
- * Freescale 9S12XDP512
- * RTOS-capable

Starting at \$125!



Evaluate * Educate * Embed

Program in
Assembler, BASIC, C, and Forth

www.TechnologicalArts.com

CCS PIC® MCU C Compiler Version 4.100

NEW

- Optimized String Handling
- IDE Compilers include **NEW** Menu Manager Coming Summer 2009

Support for **NEW** Enhanced Mid-Range PIC16 Devices

Prices start at
\$150

262.522.6500 x35 • sales@ccsinfo.com
www.ccsinfo.com/PIC16CC

PIC, PICU and dsPIC are registered trademarks of Microchip Technology Inc.



Weather Instruments for PCs

AAG
electrónica

www.aagelectronica.com

Solve complex signal acquisition problems...

- positioning & control
- environmental
- acceleration
- transients
- pressure
- vibration
- sonar
- GPS
- Linux Driver
- Guaranteed in stock
- Many newly added features
- 16-bit analog inputs and outputs
- Million sample FIFO eliminates interrupts
- Wide analog input and output ranges
- -40°C to +85°C Standard
- Order 24/7, fast and easy.

www.stx104.com
Apex Embedded Systems
help@stx104.com • 608-256-0767 x24

PHYTEC

phyCORE[®] SBCs Accelerate 32-bit Designs

phyCORE-LPC3250

ARM: LMX31 (ARM11), LMX27 (ARM9), LPC3250 (ARM9), LPC3180 (ARM9), LPC2294 (ARM7)

XScale: PXA320, PXA270, PXA255

PowerPC: MPC5554, MPC5200B, MPC565, MPC555

ColdFire: MCF5485 **Blackfin:** ADSP-BF537

- COTS Single Board Computers: shorten time-to-market, reduce development costs, forgo substantial design issues and risks
- Windows[®] Embedded CE and Linux Board Support Packages (processor-dependent)
- \$170/unit benchmark price at 1K for ARM9-based SBC
- Rapid Development Kits start at \$399
- include SBC, Carrier Board, software, docu as well as kit-specific cables, adapters and LCD
- SBC module easily ports from Carrier Board to user target hardware
- Carrier Board serves as target reference design

www.phytec.com ■ 800.278.9913 ■ www.phycore.com

www.can232.com

Only \$108 €89

CAN232 Features:
Free sample programs
8-15VDC supply via CAN
Timestamp in mS
Small size 2.7" by 1.2"
100% Bandwidth up to 125Kbit
Both 11 & 29 bit ID support
32 Message Receive FIFO
Works up to 1Mbit CAN
Simple ASCII protocol
Supports RTR Frames
Max 230Kbaud RS232
Firmware upgradable
No drivers needed
OS independent
CE Approved

CANUSB Features:
Free ActiveX component
PC, MAC & Linux support
Both 11 & 29 bit ID support
Simple CAN logger included
Free Threaded Windows DLL
Firmware upgradable via USB
Sample programs in C, C++, VB, Delphi, C#, PureBasic etc.
No need for external power
Works up to 1Mbit CAN
Supports RTR Frames
USB 2.0 Full Speed
Free USB drivers
CE Approved

Only \$154 €129

www.canusb.com

ezLCD - The "Smart Display" makes integrating a GUI ez!

8.0" ezLCD p/n: ezLCD-105

- *Versatile Programming LCD module
- *USB, SPI, RS232 Interfaces
- *Bright 250 Nit LED Display (800 x 600)
- *Integrated Touch Screen
- *LUA Scripting Language capable- For stand alone embedded apps
- *Memory 3.8 MB + SD to 2G
- *ezLCD's are also available in 2.7", 3.5", 5.6", 6.4", 10.4"

www.earthlcd.com
Call for Custom Display Configurations

I²C/SMBus

- Bus Monitors
- Protocol Analyzers
- Host Adapters
- Multiplexers
- Battery Applications
- Software Tools

MCC
Micro Computer Control

I²C is a trademark of Philips Corporation

www.mcc-us.com

SensorCore(SC)[™]

Low Cost Data Acquisition System with 48 24-bit ADCs

OEM \$89

50+ Low Cost Controllers with ADC, DAC, UARTs, 300 I/Os, solenoid, relays, CompactFlash, LCD, Ethernet, USB, motion control. Custom board design. Save time and money.

- Directly work with Thermocouples, Strain gauges . . .
- 100 M BaseT, FAT file system and CompactFlash.
- 2.0" x 4.5", C/C++ programmable, 80MHz x86, 48 24-bit ADCs, DAC, RS232, I/Os. Standalone SBC.

TERN INC. 1724 Picasso Ave., Suite A, Davis, CA 95616 USA
Tel: 530-758-0180 • Fax: 530-758-0181 www.tern.com • sales@tern.com

Going Wireless is Easy!

Zigbee[™]

Wireless Mesh Network
NEW USB Zigbee[™] Stick offers an easy way to Zigbee[™] enable PC's

- ETRX2 USB Based on Ember Single Chip Zigbee[™] 802.15.4 Solution
- No need for RF Design Experience
- 2.4 ISM Band

LEMOS INTERNATIONAL www.lemosint.com

Call Toll Free 1-866-345-3667 or email at sales@lemosint.com

CUSTOM MEMBRANE KEYBOARDS / SWITCHES



- 1 TO 2 WEEKS TURNAROUND
- VERY COMPETITIVE PRICING
- Ex.: (5) 4-switch keyboards for \$395.00
- PCB backed switches
- Custom metal backplates/assemblies
- Electronic assemblies/graphic overlays
- Electronic file transfer capabilities

Picofab Inc.

4780B Blvd. Henri-Bourassa
Charlesbourg, Quebec, Canada G1H 3A7
Tel: (418) 622-5298 • Fax: (418) 622-9996
Email: sales@picofab.net

RIGOL
Beyond Measure

NKC ELECTRONICS



From
\$595

Rigol Technologies DS1000E series

Up to 1GSa/s and 1Meg memory
50MHz and 100MHz models - TFT LCD - USB
Advanced triggering: Edge, Pulse, Video, Slope, Alt

Zeroplus LAP-16032U Logic Analyzer

16-channel - 100MHz - USB 2.0
SPI - I2C - UART - 7-segment
2 free additional protocols

\$120

Open Source Hardware

Arduino - Freeduino - Seeduino boards
Arduino Shields
BlinkM



Arduino
Duemilanove
\$30

WWW.NKCELECTRONICS.COM

OFFICIAL RIGOL DISTRIBUTOR GLOBAL SHIPPING

USB Data Acquisition

ADU208 -USB Relay I/O Interface



Complete SDK Online
at: www.ontrak.net

FEATURES

- 8, 5-AMP relay outputs
- 8, ISOLATED digital inputs
- Port Powered, Aux 5VDC output \$189.00 QTY 1

Other Models:

ADU200- 4 Channel Version with RS232 \$139.00
ADR218- Solid-State Version 8 Channel \$225.00
ADU100- 3 CH, 16-Bit ISOLATED Analog Inputs, PGA,
4 digital I/O, RS232 and 5 AMP Relay Output \$199.00

ONTRAK CONTROL SYSTEMS INC.
PH: (705) 671-2652 Fax: (705) 671-6127

www.ontrak.net

CROSSWARE

ARM7 MODULAR TOOLSET

Add support for Atmel, NXP or STMicro ARM7 variants to Base Package to suit your requirements and budget

- ☐ C/C++
- ☐ Code Wizards
- ☐ Debugging
- ☐ Simulation

Advanced software tools since 1984

www.crossware.com
info@crossware.com

PRINTED CIRCUIT BOARDS

QUALITY PRODUCT
FAST DELIVERY • COMPETITIVE PRICING

<ul style="list-style-type: none"> • Aluminum-Backed PCB • Single & Double Sided • SMOBC, RoHS • LPI Mask • Through Hole or SMT • Nickel & Gold Plating • Routing or Scoring • Electrical Testing • Artwork or CAD Data • Fast Quotes • Flex Circuits 	<p>10 pcs. (3 Days) 1 or 2 Layers \$249</p> <p>10 pcs. (5 Days) 4 Layers \$695</p> <p>(up to 30 sq. in. each) Includes Tooling, Artwork, LPI Mask & Legend</p>
--	--

PROTOTYPE THROUGH PRODUCTION

PULSAR, INC.

9901 W. Pacific Ave. • Franklin Park, IL 60131
847-233-0012 • Fax: 847-233-0013
www.pulsar-inc.com • Email: sales@pulsar-inc.com

Connect to SMT Pads

IC SMT Pads Become Connection Interface with SMT Emulation

Industry's widest selection for SMT Pattern Interconnect Technology allows Test, Prototype, Package Conversion, Emulation

Quick Custom Designs for all package types

- BGA
- QFN (MLF)
- QFP
- SOIC
- PLCC

Ironwood ELECTRONICS 1-800-404-0204
www.ironwoodelectronics.com

Order online at:
www.melabs.com

microEngineering Labs, Inc.

Development Tools for PIC® Microcontrollers

Phone: (719) 520-5323
Fax: (719) 520-1867
Box 60039
Colorado Springs, CO 80960

USB Programmer for PIC® MCUs

\$89.95
(as shown)

RoHS Compliant

Programs PIC MCUs including low-voltage (3.3V) devices

Includes Software for Windows 98, Me, 2K, & XP

With Accessories for \$119.95:
Includes Programmer, Software, USB Cable, and Programming Adapter for 8 to 40-pin DIP

LAB-X Experimenter Boards

Pre-Assembled Boards Available for 8, 14, 18, 28, and 40-pin PIC® MCUs
2-line, 20-char LCD Module
9-pin Serial Port
Sample Programs
Full Schematic Diagram

Pricing from \$79.95 to \$349.95

BASIC Compilers for PICmicro®

Easy-To-Use BASIC Commands
Windows 9x/Me/2K/XP Interface

PICBASIC™ Compiler \$99.95
BASIC Stamp 1 Compatible
Supports most 14-bit Core PICs
Built-In Serial Comm Commands

PICBASIC PRO™ Compiler \$249.95
32-bit signed variables and math operations
Supports Microchip PIC10, PIC12, PIC14, PIC16, PIC17, and PIC18 microcontrollers
Direct Access to Internal Registers
Supports In-Line Assembly Language
Interrupts in PICBASIC and Assembly
Built-In USB, I2C, RS-232 and More
Source Level Debugging

Serial LCDs

2-line x 16 \$39.95
4-line x 20 \$49.95

Quantity Discounts Available!

PICPROTO™ Prototyping Boards

Double-Sided with Plate-Thru Holes
Circuitry for Power Supply and Clock
Large Prototype Area
Boards Available for Most PIC® MCUs
Documentation and Schematic

Pricing from \$8.95 to \$19.95

See our full range of products, including books, accessories, and components at:
www.melabs.com

INDEX OF ADVERTISERS

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

Page		Page		Page		Page	
77	AAG Electronica, LLC	57	Embedded Developer	29	LPKF Laser & Electronics	41	Pololu Corp.
25	AP Circuits	72	ExpressPCB	71	Lakeview Research	78	Pulsar, Inc.
42	ARM	27	ezPCB	77	Lawicel AB	19, 25	Rabbit, A Digi International Brand
75	All Electronics Corp.	76	FlexiPanel Ltd.	15, 77	Lemos International Co. Inc.	76	Rabbit, A Digi International Brand
77	Apex Embedded Systems	15	Front Panel Express LLC	28	Linx Technologies	76	Reach Technology, Inc.
7	Atmel	69	General Circuits	73, 77	MCC (Micro Computer Control)	39, 73	Saelig Co.
33	CWAV	35	Grid Connect, Inc.	35	MachinePIER	11	SEGGER Microcontroller Systems LLC
68	CadSoft Computer, Inc.	5	HI-TECH Software, LLC	76	Maxbotix	40	Senix
14	Calao Systems	40	HobbyLab, LLC	78	microEngineering Labs, Inc.	55	Sensors Expo & Conf.
76	CapTron Corp.	75	I2CChip	21	Mouser Electronics	C3	Tech Tools
23	Comfile Technology, Inc.	30, 31	ICbank Inc.	78	NKC Electronics	2, 3	Technologic Systems
78	Crossware Products, Inc.	1	Imagineering, Inc.	C2	NetBurner	76	Technological Arts
76	Custom Computer Services, Inc.	78	Ironwood Electronics	71	Nurve Networks LLC	77	Tern, Inc.
75	DLP Design	32, 34	JKmicrosystems, Inc.	78	Ontrak Control Systems	75	Trace Systems, Inc.
40	DesignNotes	75	JKmicrosystems, Inc.	12	PCB-Pool	76	Triangle Research Int'l, Inc.
14	EMAC, Inc.	49	Jameco	C4	Parallax, Inc.	34	Xytronix R&D, Inc.
63, 77	Earth Computer Technologies	71	Jeffrey Kerr, LLC	63, 77	Phytec America LLC		
32	Elprotronic	13	Keil Software	78	Picofab Inc.		

PREVIEW of June Issue 227

Theme: **Communications**

Keystroke Communication: Design a Customizable Virtual Keyboard

Autonomous Vehicle Design: Embedded Systems, Sensor Technology, & Motor Control

SSI Controller for Linear-Position Sensors

THE DARKER SIDE High-Speed Signal Transmission: Preemphasis, Equalization, & More

ABOVE THE GROUND PLANE Solar Data Logger (Part 2): Data Points

FROM THE BENCH Location Notification: A Look at Anisotropic Magnetoresistance Sensors

SILICON UPDATE Easy (E)mbed: An Alternative Approach to Embedded Programming

ATTENTION ADVERTISERS

July Issue 228 Deadlines

Space Close: May 12
Material Close: May 19

Theme
Internet & Connectivity

Call Shannon Barraclough
now to reserve your space!
860.875.2199

e-mail: shannon@circuitcellar.com

PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

It's All About the Content, Stupid!

A lot has been written about the demise of newspapers and commercial publishing companies. The popular opinion is that newspapers in particular are going down the drain because the same information is available faster and cheaper via the Internet. I suppose there is some merit to that conclusion, but it sure doesn't jibe with my explanation for not reading as many newspapers these days. Let me explain.

OK, I'm one of those anachronistic people who still do read "real" newspapers. When I'm away at the cottage, I read *USA Today* and a local paper every morning. On Sundays, I get the local paper and the Sunday edition from the nearby metropolis. I used to get a second large city paper on Sundays along with the big city daily newspaper, but I cancelled them after getting very tired that 90% of the editorial was always about big city blood, death, and mayhem. That same reason made me give up listening to the 11PM one-hour TV news programs—too much of, "If it bleeds, it leads."

Back in Connecticut, I have a similar newspaper "disconnect," but for different yet related reasons. I still follow the same morning regime with *USA Today* and a non-Hartford local paper. But because Hartford is a much smaller city, at least the local TV channels get over the blood and guts earlier and still have 15 minutes of a half hour news program worth watching. Unfortunately, Sunday is a total washout—no newspapers!

But wait. There are plenty of Sunday papers in the Northeast, so why not? The reason is content, not the information delivery speed or cost. For me, the rejection of support for various newspapers and magazines is because of bad content and better competition, not because of faster availability of the same information elsewhere. I used to get the Hartford paper seven days a week along with the Sunday *New York Times*. At first, I just got irritated that the news presentations and political coverage were increasingly one-sided, but when the opinion pieces and news stories became indistinguishable, I had to draw the line. I wasn't going to pay for lousy (in my opinion) content. I wasn't passing on buying the paper version so I could then go to their web sites and read the same content for free. I simply didn't believe in the credibility of their content anymore.

The Internet is the most open and uniquely diverse source of mass media ever developed. Part of its uniqueness is that it fosters a low-cost interchange of ideas and published materials. At one time, we sat around the campfire and swapped stories. Now, kids sign on to a variety of social networks to hear and exchange tall tales. The beauty of the Internet and the worry for traditional publishers is that the Internet is interactive and not meant solely to push content in one direction to a captive audience. Information on any subject, to any depth of analysis desired, and with virtually any degree of correctness, is available on demand.

In my opinion, a lot of the financial problems in publishing are coincident with bad economic conditions, but certainly many newspapers and magazines are feeling a lot more heat simply because readers no longer trust their content. The fact that readers might seek alternatives on the Internet is a consequence of smart people seeking a better source. It is too simplistic to attribute all print publishing failures merely to cheaper and faster access on the Internet. Similarly, moving from a print venue to an online one doesn't alleviate the issue of crappy content.

In my opinion, publishing survival—whether using print, the Internet, PDFs, wireless e-paper, or whatever—is all about readers valuing the credibility and content of the message and not solely about the delivery medium. Don't get me wrong, printing paper is a whole lot more expensive than simply posting an article online. But no one would read *Circuit Cellar* in any form (whether in print or online) if they didn't trust us in the first place. I know we owe the trust we have earned to the smart engineers, programmers, and scientists who write the exceptional editorial we publish. In turn, they know we go to extreme efforts to remain both a viable venue for enlightening their interests as well as enhancing their professional careers.

Right now, the entire publishing industry is in flux as it tries to deal with evolving delivery technologies. Like many magazines, *Circuit Cellar* has an online component. In our case, we view Digital Plus as a complement and enhancement to our print venue and not an escape from print. When the dust settles, the commercial publishing survivors will be the ones, regardless of venue, with superior content. We intend to be among them.

steve.ciarciac@circuitcellar.com

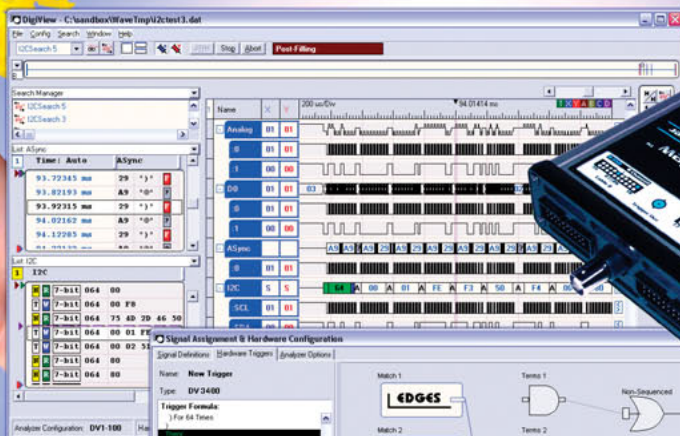
A handwritten signature of Steve Ciarcia.

PC Based Logic Analyzers

NEW Model DV3400

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

from \$499.00



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



See our Memory Emulators
from \$179.00



TechTools

www.tech-tools.com

(972) 272-9392 • sales@tech-tools.com

Copyright © 2007 TechTools • DigiView, FlexROM, EconoROM and QuickWriter are trademarks of TechTools • PICmicro is a registered trademark of Microchip Technology Inc.

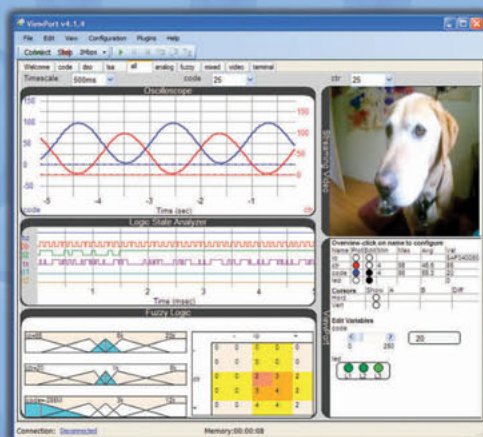
QuickWriter™

ViewPort

SOFTWARE



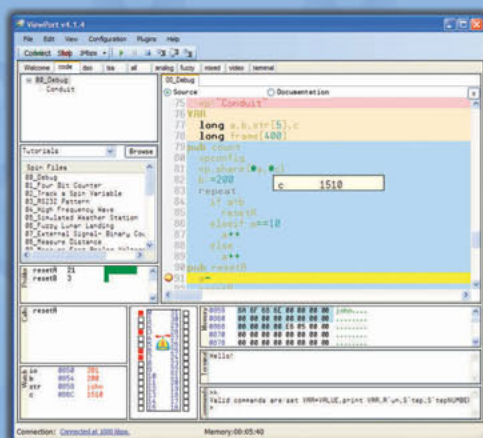
Available in two versions: Standard and Ultimate!



ViewPort is the premier debugging environment for the 8-cog Propeller multicore microcontroller. This versatile tool combines an integrated debugger with powerful graphics that show you what's going on within the Propeller chip. Monitor variables over time with the built in oscilloscope or change their value while your Propeller is running. Solve hardware problems with the logic analyzer at sampling rates up to 80 Msps. Add intelligence to your programs with the fuzzy logic module. Add vision processing to your application.

Features for ViewPort Standard (#32388; \$59.00):

- Spin-code debugger: breakpoint, pause/continue, profiler, interpreter, call stack
- Virtual oscilloscope: variable timescales, trigger, cursors, auto-measurements
- Virtual logic analyzer: capture state of all 32 pins at up to 80 Msps with trigger
- Virtual spectrum analyzer: analyze signals in the frequency domain
- Fuzzy logic engine: easily integrate and tune fuzzy logic with the control panel
- Edit Spin variables: use PC-based controls to calibrate and control your project
- Monitor variables: log and analyze variable values over time
- Real time data acquisition: integrate PC for acquisition and control (SCADA)
- Share logged data: as Matlab, text, xls, image files or stream to the Internet
- Low-speed Propeller-to-ViewPort communication rate (115 kbps)
- Simple vision processing



Additional Features for ViewPort Ultimate (#32386; \$149.00):

- High-speed Propeller-to-ViewPort communication rate of 2 Mbps
- Advanced OpenCV powered computer vision: add state-of-the-art vision to your project
- Designer/development kit for customizable instrumentation: layout the rich interface using drag-and-drop

ViewPort can be integrated into any Propeller Spin program. It requires one dedicated Propeller cog and a single line of code at the start of your program. It's easy to get started with plenty of tutorials, videos and documentation in the software. Also, a Propeller Education Kit Lab featuring ViewPort is available for free download from the Propeller forum at www.forums.parallax.com.

Order the **ViewPort Software (Standard or Ultimate)** at www.parallax.com or call our Sales Department toll-free at 888-512-1024 (Mon-Fri, 7 a.m. - 5 p.m., PDT).

Propeller, Spin, Parallax, and the Parallax logo are trademarks of Parallax Inc. Prices are subject to change without notice.

PARALLAX
www.parallax.com

Time-Triggered Systems (Part 2)

A Tool for Automating Analysis

Now that you're familiar with the topic of co-operative scheduling, it's time to move on to the topic of adding a GUI to a co-operative scheduler. The end goal is the development of an interesting multitask audio device, a modern Theremin.

In "Time-Triggered Systems (Part 1): Co-Operative Schedulers 101," I (Mike Smith) described how I rediscovered the idea of controlling multiple embedded processor tasks through the use of a co-operative scheduler (*Circuit Cellar Digital Plus* 225, 2009). One often-stated advantage of these schedulers is that they can be programmed to provide "complete predictability" of when each task is run. This is possible for two main reasons. One reason is that only one interrupt is ever active at the same time within the system. The other reason is the ease with which the developer can specify when each program task would run. This predictability can be put to great use in safety-critical systems where reliability and other known performance issues are important.

In the first article, I also showed that this complete predictability did not occur automatically, even when the system

just involved a series of trivial tasks (e.g., a series of flashing LEDs). The problem was associated with "time jitter." This is the (undesirable) delay in the execution of one task that occurs when two tasks are simultaneously prepared for running within the co-operative operating system. It is analogous to the changes in execution times that occur in a preemptive scheduler when the execution of one task is switched out or interrupted by another task. When the system just involves two minimalistic tasks, the possible presence of time jitter in a co-operative scheduler can be determined using simple "back-of-the-envelope" calculations, as I explained. However, if you are developing a more complicated project that involves a design with many interacting tasks, and if "real" timing and safety concerns are present, a more sophisticated automated tool would be better than a scrap of paper.

In this article, Lizie and I will cover the topic of adding a graphical user interface (GUI) to the co-operative scheduler to obtain detailed timing information on how tasks interact with each other during the development of a multitask audio project. We will explain how to use this tool to check the solution for a project where long tasks have the potential of blocking short tasks when running within a co-operative scheduler.

THE THEREMIN

The Theremin, or aetherphone, is a musical instrument with a number of unique firsts in its background. It is the earliest electronic musical instrument (1913), and the first musical instrument that could be played without being

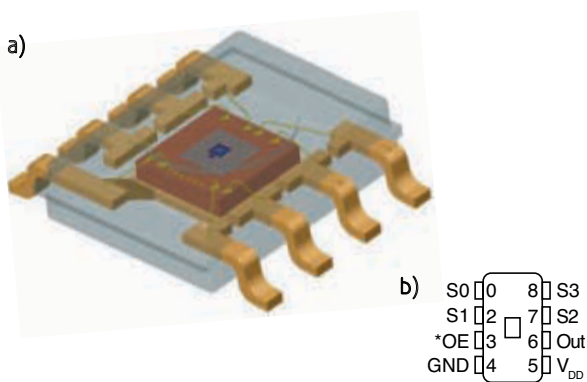


Figure 1a—This is the TAOS TSL230R light-to-frequency converter (www.circuitcellar.com/library/print/1204/Bachiochi173/3.htm).
b—This chip has two pins (S0, S1) to control its light sensitivity over a range of 1 to 100 and two pins (S2, S3) to control its frequency sensitivity over a range of 1 Hz to 1 MHz (<http://pdf1.alldatasheet.com/datasheet-pdf/view/153439/ETC/TSL230R.html>).

touched. Originally, there were two metal antennae used to sense the position of the musician's hands. These antennae were coupled to radio-frequency oscillators used to control circuitry that manipulated the Theremin's sound volume and frequency. Rather than using metal antennae, we implemented two programmable light-to-frequency converters. Changing light levels caused by shadows were used to sense the position of the Theremin-ist's hands.

Circuit Cellar columnist Jeff Bachiochi featured TAOS sensors (see Figure 1) in his 2004–2005 article series about a pulse oximeter design ("Light-to-Frequency Conversion," Parts 1–2, *Circuit Cellar* 173–174). The easy-to-use sensors provide a train (series) of 50% duty cycle square waves, whose frequency is directly proportional to the light level falling on them. Connect the sensor to a general-purpose input/output (GPIO) pin on any processor and you will be able to sense the positions of a musician's hands by the amount of shadow that's cast.

Figure 2 shows the proposed hardware. Two light sensors are connected to a processor's GPIO pins. A speaker is plugged into an evaluation board's analog output channel.

As you learned in the first part of this article series, it isn't difficult to use a co-operative scheduler to prototype a multitask embedded product, such as the Theremin (see Listing 1). First, use `TTCOS_Init()` to initialize the TTCOS scheduler (Line 6). Then add tasks to the scheduler's `todoList` by specifying each task's initial time delay and period with `TTCOS_AddTask()` (Lines 11 to 14). Next, activate the scheduler's single interrupt (timer) `TTCOS_Start()` (Line 16) and send the system into a low-power mode `TTCOS_GoToSleep()` (Line 20). The timer interrupt service (Lines 27–30) routine uses `TTCOS_Update()` (Line 29) to increment the `Run-Me-Now` variable of each task based on the current time and the task's delay and period stored in the scheduler's `to-do` list. With the processor now out of Sleep mode, the `TTCOS_DispatchTask()` routine (Line 21) activates all the tasks that need to run and the processor is given a bite of the poisoned apple and sent to sleep once again (Line 20). For more details about all the code needed to develop a basic co-operative scheduler, see Michael Pont's book *Patterns for Time-Triggered Systems*. (A free download is available at www.tte-systems.com/books.)

CODE INSTRUMENTATION

Building the Theremin required just four tasks to be added to the scheduler's `to-do` list. The `LightSensorVolume()` and `LightSensorFrequency()` tasks provide the timing information necessary to calculate the light sensors' output frequency. These are set to have a period of `RUN_OFTEN`. There is one task `ModifyAudioSettings()`, set to `RUN_AS_NEEDED`, to update the volume and frequency information used to generate the Theremin sound. This information is used by the final task, `OutputSound()`, which is `RUN_MORE_FREQUENTLY`

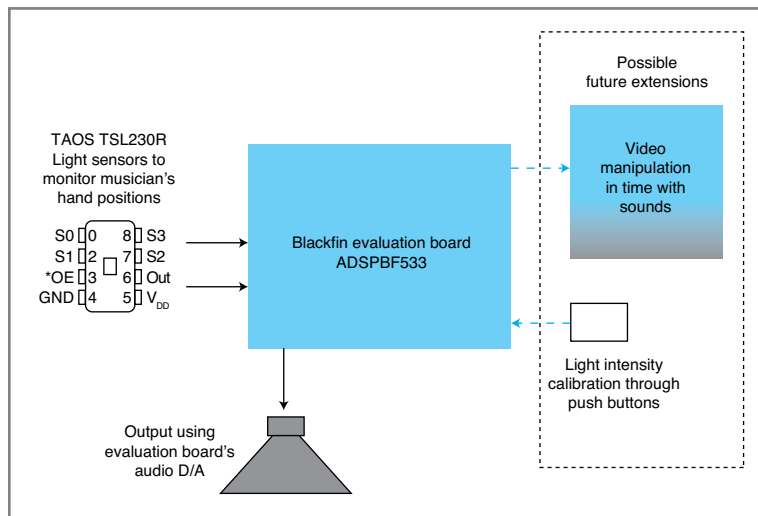


Figure 2—The initial plans are to use the light sensors to control the sound frequency and volume, but video manipulation in time with the music is possible given the evaluation board's capabilities.

to output the audio signal.

For this initial prototype, we chose approximate values for the time periods (`RUN_OFTEN`, `RUN_AS_NEEDED`, and `RUN_FREQUENTLY`) needed in the project based on initial project characteristics. However, this prototype is just the first stage in a full-blown Theremin project. Therefore, it was useful to have a tool capable of validating our design assumptions when the hardware and software actually interacted. The tool had to be able to display a "state history" to show when each task ran and how the various tasks interacted with each other. In particular, we wanted the tool to be able to automatically recognize when one task blocked the expected execution of another task.

To make this tool work, we made some changes to `TTCOS_UpdateTask()`, which was called by the TTCOS scheduler timer ISR (see Listing 2). This function was responsible for determining when a given `taskDelay` had decremented to zero (see Listing 2, Line 55). When this occurred, the task's `RunMeNow` semaphore variable was incremented; this indicated that the task was now in the `READY_TO_RUN` state. Activated by a `RunMeNow` semaphore's value greater than 1, additional software instrumentation (Lines 61 to 64) indicates whether or not the task had been prevented from running by another task:

```
if (RunMeNow > 1)
    StatusHistory(CurrentTime(),
                  taskIndex,
                  TASK_BLOCKED); // Line 61
```

The `StatusHistory()` function (see Listing 2, Lines 38 to 47) added the new task status information into `HistoryInfo`, which is an array of `HistoryStruct` structures arranged as a ring (circular) buffer (see Listing 3).

Information must be recorded to prepare for the time when the task status changes between `READY_TO_RUN`, `TASK_RUNNING`, and `TASK_COMPLETED`. This, together

Listing 1—The proposed Theremin musical instrument code appears as four tasks in the co-operative scheduler's to-do list. Three tasks change the Theremin's volume and frequency based on the shadows cast by the musician's hands. A fourth task uses this information to produce the "eerie sound" that is typical of the Theremin.

```

1  #include "../Theremin.h"
2
3  todoList Tasks[NUMBER_TASKS];
4
5  int main(void) {
6      TTCOS_Init(TICK_TIME_IS_10us); // Init scheduler "to-do" list and scheduler timer
7
8      InitHardware( );
9
10     // Add basic Theremin Tasks, initial delay and task period
11     TTCOS_AddTask(LightSensorFrequency, NO_DELAY, RUN_OFTEN);           // Theremin frequency control
12     TTCOS_AddTask(LightSensorVolume, SHORT_DELAY, RUN_OFTEN);          // Theremin volume control
13     TTCOS_AddTask(ModifyAudioSettings, LONGER_DELAY, RUN_AS_NEEDED);    // Adjust audio parameters
14     TTCOS_AddTask(OutputAudio, NO_DELAY, RUN_MORE_FREQUENTLY);         // Generate and output audio
15
16     TTCOS_Start( );              // Activate scheduler timer interrupt service routine
17                                 // which calls TTCOS_Update() to update task's Run-Me-Now variables
18
19     while (1) {                  // Wait, in low power mode, for an interrupt
20         TTCOS_GoToSleep( );      // Then run all the tasks in the scheduler's 'to-do' list according
21         TTCOS_DispatchTasks( );  // to whether their delays have expired (RunMeNow > 0)
22     }
23
24     return 0;                    // Make compiler happy
25 }
26
27 EX_INTERRUPT_HANDLER(TTCOS_Interrupt) { // Only ISR operating in the co-operative scheduler
28     Acknowledge_TTCOSInterrupt( );
29     TTCOS_Update( );              // Update the Run-Me-Now variables for each task
30 }

```

with statistical information about the minimum and maximum task run times, can be obtained by adding additional software instrumentation to the `TTCOS_DispatchTasks()` function that's responsible for running each task (see [Listing 4](#), Lines 98 to 104).

The software instrumentation code used to store status information and timing statistics in the `TimingInfo` structure can be left in the run-time code for use after the product has been released. It can form the basis of a tool that performs system diagnosis during start-up, or it can be part of a database uploaded over the Internet for more detailed problem-solving analysis—something akin to Microsoft's "Do you want to send in an error report now?"

However, there is always a problem to be faced with any software instrumentation, whether it is used during development or post-release analysis. The actual process of storing (too much) detailed status information could steal

so much time from the processor core that deadlines could be missed. This means that the presence of the instrumentation actually introduces defects into the system's performance.

With many processors, it is possible to minimize this possibility by causing the compiler to always `_inline` the code for the `StatusHistory()` (see [Listing 2](#)) and `RunStatistic()` (see [Listing 4](#)) functions. In-lining improves performance as it reduces many of the redundant operations that occur when subroutines are called (e.g., moving a value from one register to another register by the calling function) only for the value to be moved back to the original register by the called function.

The Blackfin processor has a large amount of internal instruction memory, and the instrumentation does not involve making many inline calls. However, for other processors, the number of processor clock cycles used during the software instrumentation process could be a real issue.

In such situations, the equivalent of the following C embedded assembly language macros could be useful:

```

#define StatusHistory(A,B,C) \
    asm("nop;");
#define RunStatistics(A,B) \
    asm("nop;");

```

When the compiler expands these macros, the software instrumentation is reduced to trivial do-nothing `nop` instructions. These will be removed as redundant code by any half-way-intelligent optimizing compiler, and then the core can run at full speed.

TASK STATUS & TIMING INFO

We've covered the topic of adding software instrumentation to provide task status and timing information. So now that you have all of these details, what are you going to do with them?

Our initial idea was to do something simple, such as add a breakpoint to stop the `TTCOS_Update()` code whenever a task was blocked from running when

Listing 2—The scheduler timer interrupt service routine (Listing 1, Lines 29–31) calls `TTCOS_Update()`. This function is responsible for determining which tasks are now ready to run. Software instrumentation (highlighted) can be added to document the predictability present in the system—in particular, whether one task has been blocked from running by another task.

```

34 volatile int historyIndex = 0;
35 HistoryStruct HistoryInfo[MAXHISTORY];
36
37 #pragma always_inline // Inline "write-to-history" operations to minimize instrumentation impact
38 inline void StatusHistory(unsigned long eventTime, int taskID, int taskStatus){
39     historyIndex++; // Update history index
40     if (historyIndex > MAXHISTORY) // and perform circular buffer operations
41         historyIndex = 0;
42
43     HistoryInfo[historyIndex].eventTime = eventTime; // Enter event status information
44
45     HistoryInfo[historyIndex].taskID = taskID;
46     HistoryInfo[historyIndex].taskStatus = taskStatus;
47 }
48
49 void TTCOS_Update(void) { // Part of scheduler interrupt service routine
50     int taskIndex;
51
52     for(taskIndex = 0; taskIndex < NUMBERTASKS; taskIndex++) {
53         if (Tasks[taskIndex].pointerToTask != DOLIST_ENTRY_EMPTY) { // Is there a valid task in the todoList?
54
55             if (Tasks[taskIndex].taskDelay != 0) // If taskDelay is non-zero then task is not ready to be run
56                 Tasks[taskIndex].taskDelay--; // simply decrement the remaining delay time57
57
58             else { // If the task is READY_TO_RUN
59                 Tasks[taskIndex].RunMeNow++; // then increment the RunMeNow flag
60
61                 if (Tasks[taskIndex].RunMeNow > 1) // Check to see if task has been blocked from running
62                     StatusHistory(CurrentTime(), taskIndex, TASK_BLOCKED);
63                 else
64                     StatusHistory(CurrentTime(), taskIndex, READY_TO_RUN);
65
66                 if (Tasks[taskIndex].taskPeriod != RUN_ONCE) // Prepare periodic tasks to run again
67                     Tasks[taskIndex].taskDelay = Tasks[taskIndex].taskPeriod - 1;
68             }
69         }
70     }
71 }

```

it was designed to run (see Listing 2, Line 61). We figured this breakpoint could be made “intelligent” by using the development environment’s application-programming interface to upload the status information to the host PC automatically. We planned to display the task information there using the graphing capabilities of Excel or possibly by generating a plug-in for use with the Eclipse development framework. However, when we actually got around to developing a “state history” GUI, the amount of work involved did not look so appealing. So, we searched instead for existing shareware capable of performing the same operations. We found an unexpected solution.

We had previously worked with the Analog Devices VDSP integrated development environment (IDDE) that has a state history plug-in (VDKHistory.dll). Running under Windows, this plug-in used to display the performance of Analog Devices’s preemptive VDK scheduler. This

plug-in includes an interface for customizing the performance analysis of multiple threads. There is considerable similarity in functionality needed to analyze preemptive and cooperative scheduling. Thus, the VDK dynamic timing and event information was stored in a history table with a structure equivalent to the `HistoryStruct` entries in Listing 3.

The VDK Help pages indicated that the VDK state history plug-in accessed a global VDK C++ class member variable:

```

HistoryBuffer g_History =
{
    MAX_HISTORY_ENTRIES,
    currentEntry,
    TTCOS_HistoryTable }

```

The VDKHistory.dll plug-in was able to generate the state history shown in [Photo 1](#) when we added this variable to a C++ file linked to our C project.

We encountered typical issues associated with using someone else’s

existing code in an unusual way. We had to redefine display constants, such as `TASK_BLOCKED`, in terms of the constants needed by the VDK plug-in. Analog Devices event type of information can be chosen from a list of possible events equivalent to `taskCreated`, `taskDestroyed`, and `taskSwitched`. Their `taskStatusChange` event type permits the recording of when the task status switches between Ready, Running, Blocked, or Sleeping.

The VDSP development environment handles projects with both C and C++ code. But we wanted to avoid generating such a mixed project; however, there was no documentation available for using the VDK plug-in in this unusual manner. This made things a little difficult.

For example, it is straightforward to access the global VDK C++ `g_History` variable from C and assembly code using the standard name-mangled syntax format of extern History-Buffer `g_History__3VDK`. However,

if you build your own version of the C name-mangled `g_History__3VDK` variable and delete the C++ version, then the plug-in will no longer recognize the history table! The new `g_History__3VDK` label can be seen in the symbol table, but somehow that is not enough to allow the dll to work.

Additional minor annoyances were associated with the lack of access to the VDK plug-in internals. This meant it was not possible to modify the picture legend to display such words as *taskCreated* rather than *threadCreated* (see Photo 1). Nor was it possible to customize the display axis labels to display the actual task names. The first issue can be

Listing 3—Adding this `HistoryStruct` to the code adds all the necessary information to allow the `VDSPHistory.dll` plug-in to generate timing information for the co-operative scheduler. Information stored in the new `TimingStruct` can be used to provide detailed run-time statistics to determine whether the code meets design requirements.

```
#define DOLIST_ENTRY_EMPTY (0)

enum {READY_TO_RUN, TASK_BLOCKED, TASK_RUNNING, TASK_COMPLETED,
TASK_OVERRUN};

typedef struct {
    unsigned long int eventTime;
    unsigned int taskID;
    unsigned int eventType;
    unsigned int taskStatus;
} HistoryStruct;

typedef struct {
    unsigned long int minRunTime;
    unsigned long int maxRunTime;
    unsigned long int lastRunTime;
    unsigned long int taskSpecification;
} TimingStruct;
```

Listing 4—Software instrumentation (highlighted) can be added to `TTCOS_DispatchTasks()` to record the run-time statistics for each task.

```
74 TimingStruct TimingInfo[NUMBER_TASKS];
75
76 #pragma always_inline
77 inline void RunStatistics(int taskID, unsigned long taskDuration){
78     // Record minimum and maximum run times
79     if (taskDuration < TimingInfo[taskID].minRunTime)
80         TimingInfo[taskID].minRunTime = taskDuration;
81
82     if (taskDuration > TimingInfo[taskID].maxRunTime)
83         TimingInfo[taskID].maxRunTime = taskDuration;
84
85     // Indicate that outside of project specifications
86     if (taskDuration > TimingInfo[taskID].taskSpecification)
87         StatusHistory(CurrentTime(), taskID, TASK_OVERRUN);
88 }
89
90 void TTCOS_DispatchTasks(void) { // Dispatches (runs) the next task (if one is ready)
91     int taskIndex;
92     unsigned long int taskStart;
93
94     for(taskIndex = 0; taskIndex < NUMBER_TASKS; taskIndex++) {
95         if(Tasks[taskIndex].RunMeNow > 0) { // Check for a task ready to run
96             // Add status information about running the task
97             // and collect run time information
98             StatusHistory(CurrentTime(), taskIndex, TASK_RUNNING);
99             taskStart = CurrentTime();
100
101             (*Tasks[taskIndex].pointerToTask)(); // Run the task
102                                                    // and record duration
103             StatusHistory(CurrentTime(), taskIndex, TASK_COMPLETED);
104             RunStatistics(taskIndex, CurrentTime() - taskStart);
105
106             Tasks[taskIndex].RunMeNow--; // Indicate that the task has run
107                                           // Remove 'RUN_ONCE' tasks from todoList
108             if (Tasks[taskIndex].taskPeriod == RUN_ONCE)
109                 TTCOS_DeleteTask(taskIndex);
110         }
111     }
112 }
```

solved by “ignoring the problem,” and a Notepad text window can be superimposed on the state history picture to solve the labeling issue in the near-term (see Photo 1).

If you want to build a scheduling GUI for a co-operative scheduler, we would recommend the route of “creative adaptation.” The preemptive scheduler state history display plug-in of your own evaluation kit’s IDDE is certainly a good place to start!

NEW THEREMIN PROTOTYPE

When we planned the four tasks for the Theremin (Listing 1), we assumed that each of the `LightSensor` tasks would have the format in Listing 5 to capture the timings for the beginning and end of the pulses output by the TSL230R sensor. The first two while loops were used to find the leading edge of the light sensor pulse (`time1`). The other while loops were used to find the end of the pulse (`time2`). These edge times could be used to determine the light sensor frequency and allow the `ModifyAudioSettings()` task to update the volume and frequency information of the Theremin.

As you can see in the state history in Photo 1 (and it’s obvious in hindsight), there was an issue with the infrequent `LightSensor()` “wait-for-signal-transition” tasks blocking the more frequent audio `OutputSound()` task (top track showing `EventBlocked` orange). This form of coding scheduler tasks with “wait loops” was not particularly co-operative. The time that the sensor tasks spent waiting for external events continually blocked the more frequent audio event. This meant that the audio stream could be disrupted and the Theremin would sound terrible rather than capable of providing the “eerie” sounds needed for all of our lucrative movie soundtrack contracts.

This issue is common in embedded applications that use co-operative schedulers. For example, how do you put out the contents of a long buffer over a serial peripheral interface (SPI) without blocking other tasks? We found the answer is actually fairly straightforward when we adapted

ideas from Pont’s book.

First, we needed to re-express (re-factor) the task in terms of the following series of states of the input GPIO input signals:

```
enum {HIGH1, LOW1, HIGH2, LOW2};
```

We then recoded the `LightSensor` task to perform either an operation in response to a transition between two states (e.g., input changes from low to high) or simply to exit the task (see Listing 6).

With this new format, two advantages became immediately apparent. One, the waiting for a sensor input to change no longer occurred inside a task—a situation which could possibly cause slow tasks that timed signals to block the faster occurring audio tasks. Two, the “waiting” occurs inside the `TTCOS_GoToSleep()` subroutine with the processor in a low-power mode. This approach offered considerable power savings with a hand-held Theremin product.

Photo 2 shows that this new “no-waiting policy” ensured that the faster tasks were no longer blocked by tasks waiting for slow signal changes. We also show an undocumented feature of the VDK state history plug-in. By using “cycles since the program started” instead of the “number of interrupts” as a time measure in the `HistoryTable`, we got a quick and clear indication of the exact execution time of each task.

CO-OPERATIVE VS. PRE-EMPTIVE

Is this suggested co-operative approach any better than simply measuring times using interrupts generated each time a GPIO pin senses an edge in the light sensor input signal? Why not just use a pre-emptive-type scheduler where tasks, activated by their own external interrupts, interrupt other tasks?

The answer depends on what is meant by *better*. If there is only one light sensor, and thus only one interrupt, the pre-emptive scheduler always will be more accurate as we identify the sensor signal edge precisely as the “exact” time when the input interrupt is triggered. Compare that accuracy with the co-operative scheduler, where an individual task can never provide precision greater than an integer multiple of the timer interrupt.

On the other hand, the uncertainty in timing with every task in the co-operative scheduler remains the same regardless of the number and timing of input signals that are present. This precise knowledge can be put to real advantage in many situations.

Is this level of timing precision a real issue because just how accurately do you need to measure the period of a signal? If it is really necessary, you can set the accuracy to better than any specified $x \mu s$ by simply adjusting the frequency at which the co-operative scheduler timer causes interrupts.

TIMING REQUIREMENTS

Up to this point, we have described the timings of tasks very loosely: `RUN_AS_NEEDED`, `RUN_OFTEN`, and

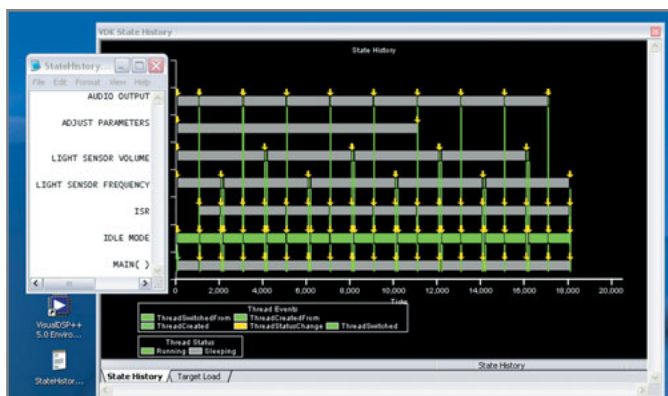


Photo 1—The inclusion of a single global C++ variable was all that we needed to enable an Analog Devices VDK State History plug-in to display the co-operative scheduler’s timing information. We can see that the `AudioOutput()` task (top line) is continually being blocked by the wait-till-done loops in the timing `LightSensor()` tasks.

Listing 5—The most obvious form of the `LightSensorTask()` involves while loops to wait while the input signal changes between four states. In a co-operative scheduler, this waiting would block other tasks that need to execute more frequently.

```
void LightSensorTask( ) {
    while (GPIOPin( ) == HIGH) /* Wait */;
    while (GPIOPin( ) == LOW) /* Wait */;
    // Now at leading edge of the sensor pulse
    time1 = CurrentTime( );

    while (GPIOPin( ) == HIGH) /* Wait */;
    while (GPIOPin( ) == LOW) /* Wait */;
    sensorPeriod= CurrentTime( ) - time1;
}
```

`RUN_MORE_FREQUENTLY`. To get practical definitions of these timing intervals, we need to think in detail of what we are trying to do.

The pianist in our choir often uses “eighth notes,” and when playing “flat out,” plays tunes with 160 beats per minute (quarter notes per minute). Let’s use that number as a starting point and ignore possible carpal tunnel syndrome for our Theremin-ist. This would suggest that we set around six note selections per second as our target for the `LightSensor()` tasks. However, that does not mean that we select 0.13 s (6 Hz) for the `RUN_OFTEN` interval. The `LightSensor()` task must sense four signal transitions for each note selection. These timings directly impact on the note sound, and we don’t want any sour, off-key notes. This indicates a probably required accuracy of around 1%. That means that `RUN_OFTEN` should be around 1.3 ms as a maximum.

While worrying about what value to use for `RUN_OFTEN`, we uncovered a missed design issue. The actual value depends on the settings of the light sensor’s frequency selection pins (see Figure 1). We need the light sensor to put out a signal whose frequency we can accurately measure with the `LightSensor()` task. Our Theremin-ist will probably start a career in a dark sleazy night club before becoming world famous and appearing on the well lit stage at Carnegie Hall in New York. Due to the varying light conditions, we need to add a new task `Adjust-LightSensitivity()` to use the processor’s GPIO output pins to control the S1 and S0 sensitivity select pins on the TAOS TSL230R light sensor (see Figure 1). This task can be set to `RUN_OCCASIONALLY`. We can imagine this task responding to pushes on two buttons (`INCREASE_SENSITIVITY` and `LOWER_SENSITIVITY`) connected to a processor’s GPIO input pins. That suggests that `RUN_OCCASIONALLY` could be set to around 0.1 s (10 Hz) to respond to the user’s input requests in a reasonably responsive manner.

That now leaves us with the audio `OutputSound()` task. For the Theremin application, imagine the flow of this task something like the following. A sound signal is pre-sampled and stored in an array that can be accessed as a circular buffer. For an initial prototype, we stored a single period of a sine wave and had

the Theremin use the sine wave samples to generate a pure musical tone. Each time the `OutputSound()` task is activated, a value from the sound array is sent to the evaluation board’s DAC at a volume determined by the `LightSensorVolume()` task timings. Then the pointer into the sound array is adjusted based on the values recorded by the `LightSensorFrequency()` task. A large adjustment to the pointer position could be made if we want to generate a high-frequency sound from the stored sine wave. A smaller step through the stored sound array will generate lower frequency tones.

We will get a distorted sound signal if we just set `RUN_MORE_FREQUENTLY` = 0.5 ms, which will generate a low 2-kHz audio sampling rate. An eerie distorted sound might just be appropriate for our Theremin. However, this audio sampling rate would probably be insufficient if we want to generate the Theremin sound accompanied by a sampled music signal “beat.” So, we can make a design decision to set `RUN_FREQUENTLY` to handle a 48-kHz audio sampling, requiring a scheduler timer interrupt at 20 µs intervals.

A GOOD DESIGN?

We have gone through the system architecture analysis and described the prototype system. Now we are ready to

Listing 6—We can avoid the `LightSensor()` routine from blocking audio tasks by replacing the slow code of Listing 5 that “waits-till-some-input-value-changes.” Instead we need fast code that operates under the principle: “If no input value has changed, I’m leaving as quickly as possible and going elsewhere to do something more important!”

```
void LightSensorTask( ) {
    static int currentState = HIGH1;
    int newState = currentState;
    int pinState = GPIOPin( ); // Input level
    switch (currentState) {
        case HIGH1: /* Test for sensor going low */
            if (pinState == LOW)
                newState = LOW1;
            break;
        case LOW1: /* Test for sensor going high */
            if (pinState == HIGH) {
                newState = HIGH2;
                // Now at leading edge of light sensor pulse
                time1 = CurrentTime( );
            }
            break;
        case HIGH2: /* Test for sensor going low */
            if (pinState == LOW) {
                newState = LOW2;
            }
            break;
        case LOW2: /* Test for sensor going high */
            if (pinState == HIGH) {
                newState = HIGH1;
                sensorPeriod = CurrentTime( ) - time1;
            }
            break;
    }
    // Update the task state
    currentState = newState;
}
```

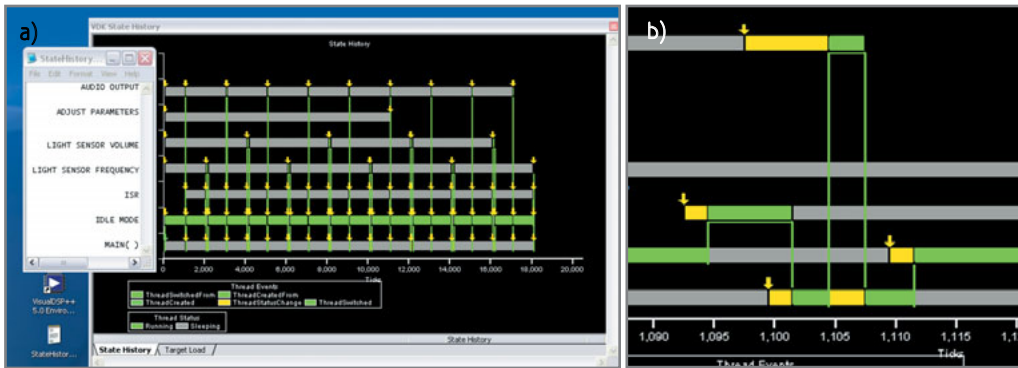



Photo 2a—We can get a clear indication of the execution time of each task by using “cycles since the program started” rather than the “number of interrupts” time measure normally used by the VDK State History display plug-in. We can even use, at no extra expense, the plug-in’s zoom feature for really close-up shots with great timing detail. **b**—On the right, we see the processor switching from the low-power idle mode (track 2) into the ISR (track 3), where the AudioOutput() task (top track) is prepared to run. The processor drops out of its idle mode to run the loop in main() (bottom track). The AudioOutput() task is then dispatched (top track) before the processor goes back to sleep (track 2).

the 1% accuracy needed for the LightSensor() timings. However, we now had an internal time interrupt at 50 kHz servicing an external device that acquired samples at a 48-kHz rate. That made for rather tight design constraints.

There were two obvious solutions. The first was to speed up the scheduler timer to provide 100-kHz interrupts (10 μ s). The second was to switch the co-operative scheduler to use the external audio interrupts, rather than the internal timer. Both approaches have pros and cons.

code for a finished product, or are we? What practical issues have we overlooked?

The 20- μ s scheduler interrupt necessary to handle the audio sampling rate looked easily capable of providing

Using an internal 100-kHz timer interrupt ensured that we could meet a timing guarantee that the audio chip

Listing 7—In this working code for a prototype Theremin, many problems were solved by adding a task (Line 9) to switch control of the co-operative scheduler from the internal timer interrupt to the external audio interrupt. In this project, tasks were added to enable the recording and playback of “musical beats” to accompany the Theremin music (Lines 19 to 26). The DSPTasks code is used to automate the match between the beat tempo of the recorded music and the beat of the music generated by the Theremin. This software task must meet strict timing requirements if it is not to block the other, more hardware-oriented, tasks.

```

3  int main(void) {
4      TTCOS_Init( ORIGINAL_TICK_TIME_IS_10us);    // Start the scheduler using internal timer interrupts
5
6      InitHardware( );
7
8      // Switch from internal timer interrupts to external audio interrupts to control the scheduler
9      TTCOS_AddTask(SwitchToAudioInterrupt, NO_DELAY, RUN_ONCE);
10
11     // Wait a while to ensure the system is stable
12     // Play Theremin music on one channel and 'recorded' background beats on another
13     TTCOS_AddTask(AudioTask,                HUNDRED_MS_DELAY,    EVERY_TICK);
14
15     // Read Light Sensors that control Theremin volume and frequency
16     TTCOS_AddTask(ReadLightSensors,          HUNDRED_MS_DELAY + 1,  EVERY_TICK);
17
18     // Use switches SW1 and SW2 to control recording and playback of background beats to accompany music
19     TTCOS_AddTask(RecordSW1Button,           HUNDRED_MS_DELAY + 2,    EVERY_20_MS);
20     TTCOS_AddTask(PlayBackSW2Button,         HUNDRED_MS_DELAY + 3,    EVERY_20_MS);
21
22
23     // Use switches SW3 and SW4 to provide coarse adjustment of background beat tempo
24     // Display information on evaluation board's LEDs
25     TTCOS_AddTask(Down_Shift_SW3Button,      HUNDRED_MS_DELAY + 4,    EVERY_20_MS);
26     TTCOS_AddTask(UpShift_SW4Button,         HUNDRED_MS_DELAY + 5,    EVERY_20_MS);
27     TTCOS_AddTask(DisplayLEDShift,           HUNDRED_MS_DELAY + 6,    EVERY_20_MS);
28
29     // DSP tasks to automate the beat tempo in response to the Theremin music being played
30     // and linearize the light sensor reading
31     TTCOS_AddTask(DSPTasks,                   HUNDRED_MS_DELAY + 7,  EVERY_10_MS);
32
33     TTCOS_Start( );                                // Activate scheduler timer interrupt service routine
34                                                    // which calls TTCOS_Update( ) to update task's Run-Me-Now semaphore
35
36     while (1) {
37         TTCOS_GoToSleep( );                        // Wait, in low power mode, for an interrupt
38                                                    // The interrupt service routine calls TTCOS_Update( )
39         TTCOS_DispatchTasks( );                    // Run all the tasks in the system according
39                                                    // to whether their delays have expired
40     }
41
42     return 0;                                       // Make compiler happy
43 }

```

input value would be updated every 20 μ s. However, it could have been argued that having this higher interrupt rate meant that we were “wasting battery power” as we were often unnecessarily waking the processor from its `TTCOS_Sleep()` low-power mode.

In some ways, worrying about power consumption seems superfluous. After all, if we have to drive current consuming audio amplifiers using an external power supply, why is there a need to run the processor itself on batteries?

However, it is actually a good design point to consider. Suppose we want to run “Theremin music” as part of a stand-alone hand-held game application using head phones as output. Ignoring other power consumption issues, by using faster interrupts in this situation, we would be consuming more of the battery capacity than is really necessary to power the processor core.

In Listing 7, a screen dump of a running Theremin musical instrument, we solved this problem by adding a `RUN_ONCE` task that switches the system from using the internal timer interrupt to drive the scheduler to using the external audio (Line 9). However, that alone is rather an unsafe design decision, as we have swapped from using a highly reliable internal interrupt for a less reliable external interrupt. If that external interrupt vanishes, the whole system would hang!

In the current situation, having the system lock up because the audio interrupt dies is not a practical concern. We need the audio chip to work in order to play the Theremin music. If this chip dies, we still have to pack up our gig and go home, regardless of whether the system is running (and happily printing out error messages) or has totally locked up!

However, this does bring up a valid point. Just how do you handle error conditions with a co-operative scheduler? This, as they used to say in the cliff-hanger movies, will be covered in the next exciting episode. There we will detail data acquisition issues using co-operative schedulers. ■

Mike Smith (mike.smith@ucalgary.ca) has been contributing to Circuit Cellar since the 1980s. He is a professor in computer engineering at the Schulich School of Engineering, University of Calgary, Canada. Mike's main interests are developing new biomedical engineering algorithms and moving them onto multicore and multiple-processor embedded systems in a systematic and reliable fashion. He recently became a convert to the application of Agile Methodologies in the embedded environment. Mention “test-driven development” and his eyes light up. In 2008, Mike had his Analog Devices University Ambassadorship renewed for the eighth straight year.

Lizie Dunling-Smith is a fourth-year undergraduate student at the University of Alberta, Canada. She will graduate in 2009 with a B.Sc. in engineering physics, specializing in electrical engineering.

SOURCES

ADSP-BF533 Blackfin processor

Analog Devices, Inc. | www.analog.com

TTSL230R Sensors

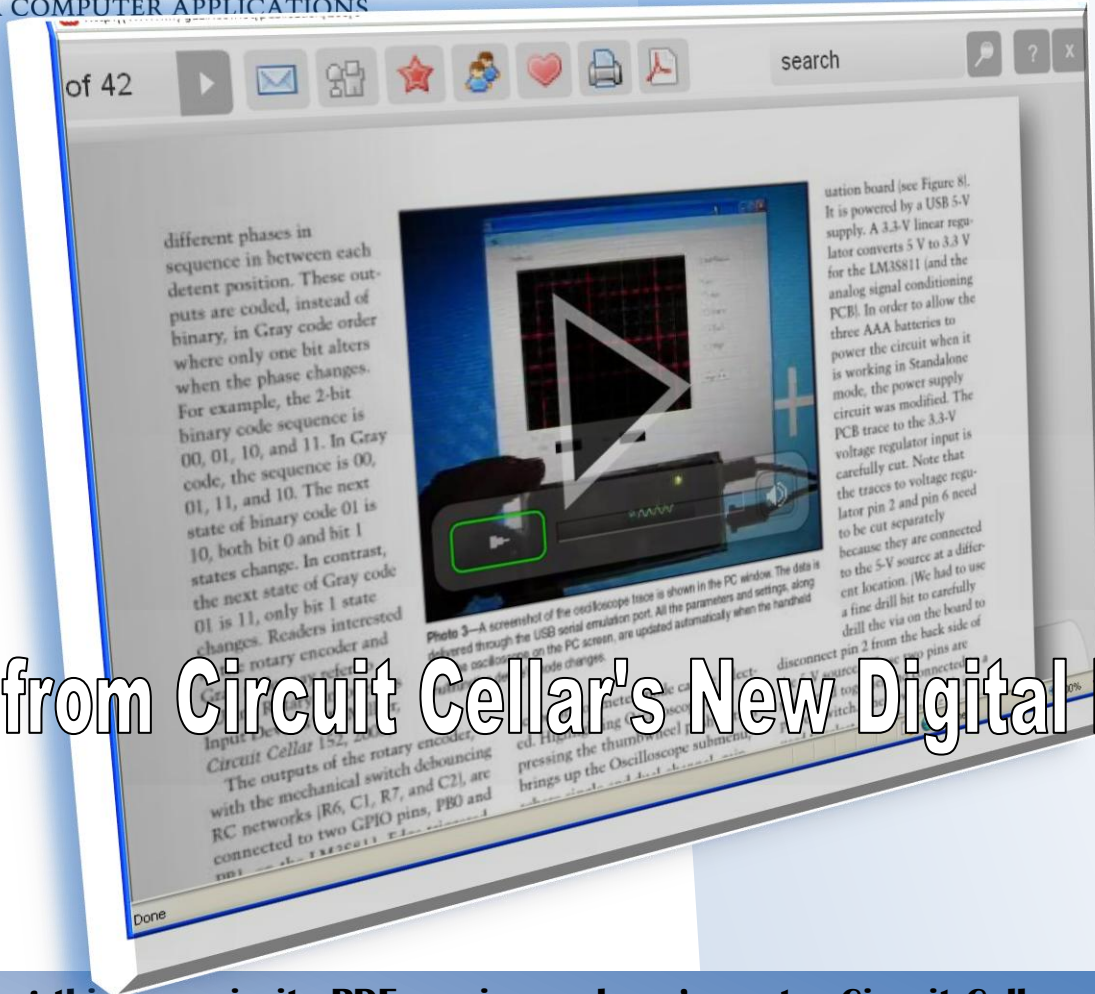
TAOS, Inc. | www.taosinc.com

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

***SUBSCRIBE or
RENEW TODAY!***

Video from Circuit Cellar's New Digital Plus



If you're seeing this issue in its PDF version and you're not a Circuit Cellar subscriber, you may be missing out on audio and video enhancements that are included with Circuit Cellar's flip book version of Digital Plus. For an example of an editorial video enhancement, **[CLICK HERE](#)** for courtesy access. (URL links to Circuit Cellar's flip book hosted by trusted source at mygazines.net.)

By maintaining a Circuit Cellar Digital Plus subscription, you ensure that you're among the first to view Circuit Cellar magazine each month. And you'll get all the benefits of multimedia material in the flip book version of the magazine along with a PDF version for your archives.

Always be among the first to have access to Circuit Cellar each month. And guarantee that you'll see it all. Renew or Subscribe to Digital Plus today! **[CLICK HERE](#)** for subscription options.