New SoC Technology, p. 22 • Practical Parallel CRC Generation, p. 38 • MCU-Based Theremin Design, p. 62

www.circuitcellar.com

CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

#234 January 2010

EMBEDDED APPLICATIONS

Scalable Capacitive Touch Sensing

Platform-Independent Stepper Control

Build a Teletext-Based TV Interface

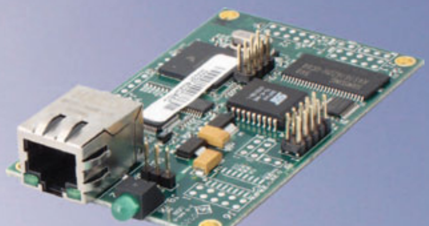
Ethernet Control System Interface

Debugging and Testing 101

0 74470 75349 0 01>

$5.95 U.S. ($6.95 Canada)

SSH ENCRYPTED SERIAL TO ETHERNET SOLUTIONS



Device P/N: SB70LC-100CR
Kit P/N: NNDK-SB70LC-KIT

$47 Qty. 1000

SB70LC
2-port serial-to-Ethernet server

Device P/N: SB700-EX-100CR
Kit P/N: NNDK-SB700EX-KIT

SB700EX
2-port serial-to-Ethernet server
with RS-232 & RS-485/422 support

$129 Qty. 1000

Device P/N: CB34-EX-100IR
Kit P/N: NNDK-CB34EX-KIT

$149 Qty. 1000

CB34EX
industrial temperature grade
2-port serial-to-Ethernet server
with RS-232 & RS-485/422 support
and terminal block connector

Instantly network-enable any serial device

Works out of the box - no programming is required

Customize to suit any application with low-cost development kit

256-bit encryption protects data from unauthorized monitoring

Features:

10/100 Ethernet

TCP/UDP/SSH/SSL modes

DHCP/Static IP Support

Data rates up to 921.6kbps

Web-based configuration

Need a custom solution?

NetBurner **Serial to Ethernet Development Kits** are available to customize any aspect of operation including web pages, data filtering, or custom network applications. All kits include platform hardware, ANSI C/C++ compiler, TCP/IP stack, web server, e-mail protocols, RTOS, flash file system, Eclipse IDE, debugger, cables and power supply. The NetBurner Security Suite option includes SSH v1 & v2 support.

NetBurner
Networking in One Day!

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

freescale™
Alliance Member

5 Competitive Advantages

Overseas Manufacturing

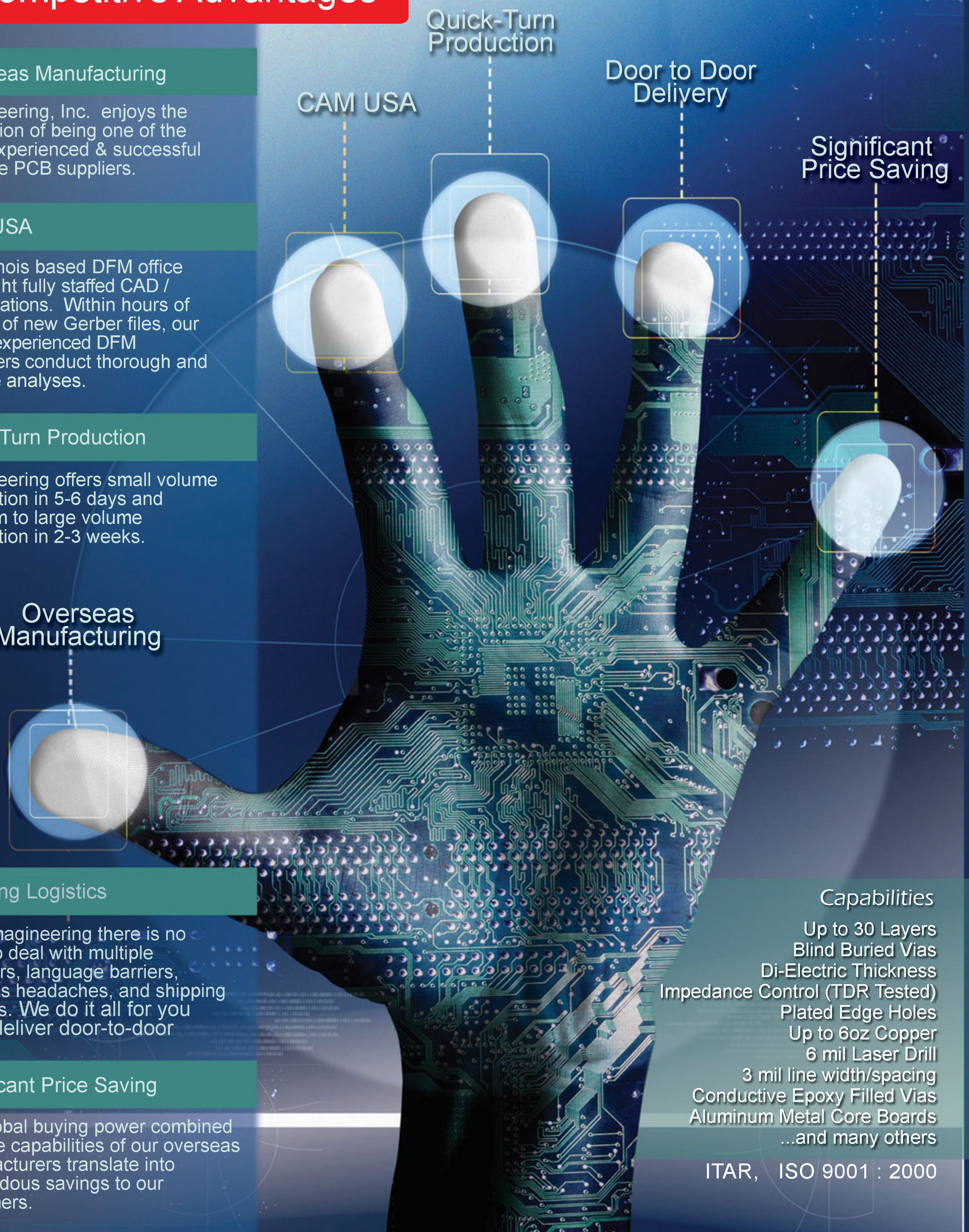
Imagineering, Inc. enjoys the reputation of being one of the most experienced & successful offshore PCB suppliers.

CAM USA

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

Quick-Turn Production

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



Shipping Logistics

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

Significant Price Saving

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

Capabilities

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

ITAR, ISO 9001 : 2000

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

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

23 YEARS IN BUSINESS...AND STILL GOING STRONG

WIZnet 2010 iMCU DESIGN CONTEST

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

Awaken the Wonderful Wizard in YOU!

iMCU DESIGN CONTEST

NOV. 2009 ~ JUN. 2010

Your creativity and design project idea could win you a share of $15,000 in Cash Prizes and Recognition in Circuit Cellar magazine.

For details, visit www.WIZwiki.net

FUTURE ELECTRONICS

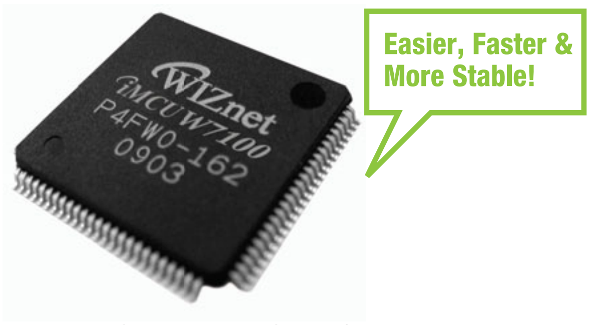
iMCU7100EVB Contest Special Price: 49.00 USD

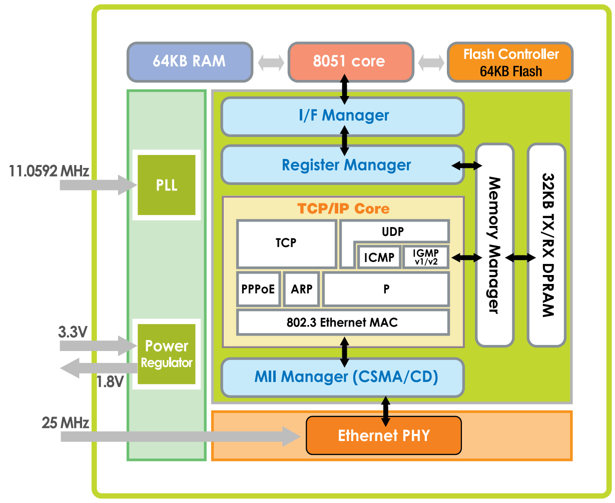
Co-sponsor Official Sample Purchase: www.FutureElectronics.com

Internet Embedded MCU

New Paradigm of MCU for Internet Devices

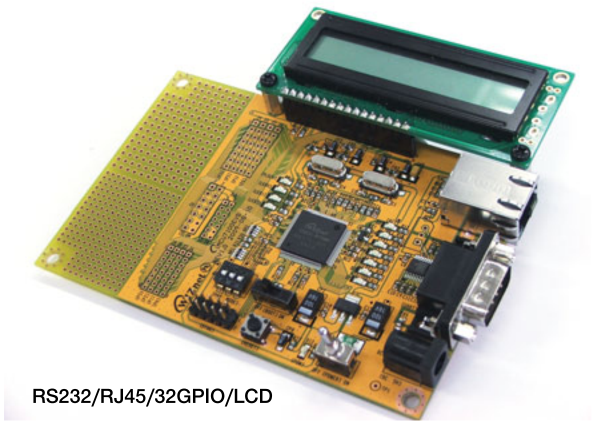
iMCU W7100





- H/W TCP/IP + MAC + 10BaseT/ 100Base TX Ethernet PHY
- Fully software compatible with standard 8051
- Pipelined architecture with standard 8051
- 64K Bytes e-Flash memory
- 64K Bytes SRAM memory
- 100-LQFP Lead-free package
- Single Chip Serial-to-Ethernet Gateway

iMCU7100EVB



Open Source Codes

- TCP lookback
- UDP lookback
- DHCP Client
- DNS Client
- Serial-to-Ethernet Gateway
- Internet LCD display
- HTTPC
- Telnet

Available at www.WIZwiki.net

WIZnet

www.WIZnet.co.kr
www.WIZwiki.net

TASK MANAGER

More Than Tinkering

In a recent *Wall Street Journal* article, Justin Lahart posited how the down economy has led to the resurgence of tinkering in America ("Tinkering Makes a Comeback Amid Crisis," November 12, 2009). Apparently, at a time when new jobs are scarce and credit is tight, "hackerspaces" are rapidly popping up all over the U.S. It's in these communal work spaces that tinkerers (many of whom are novice designers) can "share tools and ideas" as they effort to invent the next top-selling electronic widgets, create affordable versions of costly electronic devices, or simply pass time between job interviews. DIY electronics projects are all the rage.

I'm not surprised that we've arrived at this point, particularly because during the past two decades we've seen DIY "revolutions" in areas ranging from website design, to day-trading stocks, to real estate "flipping." So, when you combine the DIY mentality with the average American's love for everything electronic, it makes sense that the general public would eventually catch the electronics engineering bug. Add to that a little economic uncertainty and the bug quickly becomes a fever. Fortunately, with entry-level kits, step-by-step online design videos, and a little perseverance, just about anyone can start designing and programming on a tight budget. And that's exciting. The more people engaged in design and programming, the better!

As with past electronics-related revolutions, *Circuit Cellar* readers are ahead of the curve. Our job is to help you remain there by providing you with new articles, code, and product information each month. Don't worry, we've got you covered. In this issue, we do what we do best so you can do what you do best. We present articles by serious designers who aren't your average tinkerers and hackers. You can use the articles as starting points for your new embedded designs and computer apps.

In "Teletext-Based TV Interface," Hans-Dieter Paul explains how to use a TV's built-in teletext decoder for displaying colored text and pseudo-graphics (p. 14). On page 30, Ralph Stirling describes his CtrlBox, which is a motion control system that provides an interface between an FPGA board and a host computer. Turn to page 38 to learn about the cyclic redundancy check (CRC). Evgeni Stavinov describes a practical method of generating Verilog or VHDL code for the parallel CRC. In "Multichannel Touch Sensors" on page 46, Matt Oppenheim explains how to implement scalable capacitive touch sensing. Check out Miguel Sanchez's "Three-Axis Stepper Controller" for a concise description of how to build a platform-independent driver board (p. 54).

As for our columnists, Tom Cantrell presents the PSoC 3 and PSoC 5. If you liked the first PSoC, you're in for a treat (p. 22). On page 62, Jeff Bachiochi describes how he built his own version of the Theremin. George Martin closes the issue with some helpful debugging techniques (p. 70). With sound debugging skills in your arsenal, you'll have the confidence to tackle any design project. It's time to get started!

cj@circuitcellar.com

C. Abate

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

MANAGING EDITOR
C. J. Abate

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Jeff Bachiochi
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Gary Bodley
Ken Davidson
David Tweed

PUBLISHER
Hugo Vanhaecke

MARKETING
Shannon Barraclough

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNERS
Grace Chen
Carey Penney

STAFF ENGINEER
John Gorsky

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

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

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

Cover photography by Chris Rakoczy—Rakoczy Photography
www.rakoczyphoto.com
PRINTED IN THE UNITED STATES

CONTACTS

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

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

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

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

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Vernon, CT 06066. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $29.95, Canada/Mexico $34.95, all other countries $49.95. Two-year (24 issues) subscription rate USA and possessions $49.95, 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.

Let your geek shine.

Meet Steven Kennedy, SparkFun customer and high school teacher. In one of his classes, Steven helps his students discover the world of physical computing by encouraging them to explore all the facets of engineering – whether it is electrical, mechanical, or aeronautical. Using SparkFun products, Steven's class has created projects ranging from a homemade pick-and-place machine to a robotic arm.

Whether you're looking for tutorials to get started in electronics, or a way to inspire your students, the tools are out there. Create an environment of invention, and let your geek shine too.

sparkfun™ ELECTRONICS

Sharing Ingenuity

WWW.SPARKFUN.COM

INSIDE ISSUE 234

BONUS CONTENT
Arduino Internet Clock

January 2010 • Embedded Applications

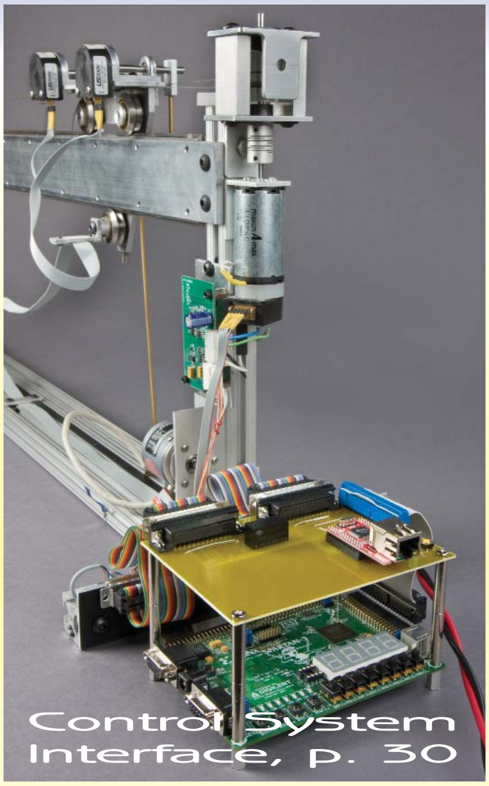
14 **Teletext-Based TV Interface**
Hans-Dieter Paul

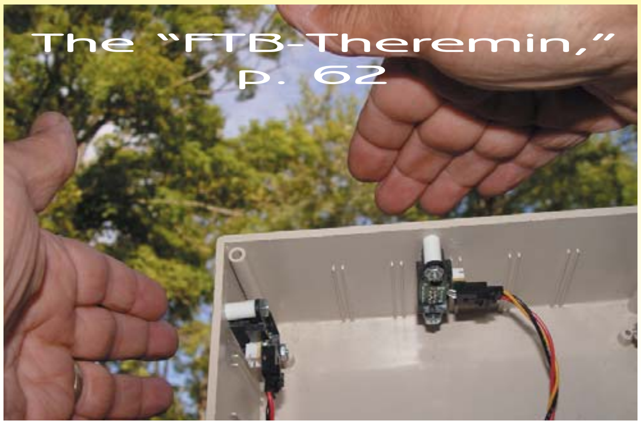
30 **The CtrlBox**
An Ethernet Control System Interface
Ralph Stirling

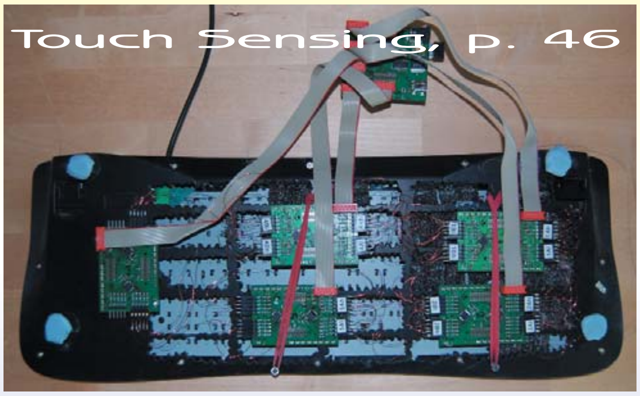
38 **A Practical Parallel CRC Generation Method**
Evgeni Stavinov

46 **Multichannel Touch Sensors**
Implement Scalable Capacitive Touch Sensing
Matt Oppenheim

54 **Three-Axis Stepper Controller**
Miguel Sanchez







22 **SILICON UPDATE**
SoC with a Capital "P"
A Look at the PSoC 3 and PSoC 5
Tom Cantrell

62 **FROM THE BENCH**
Good Vibrations
Wave Shaping and Theremin Design with an MCU
Jeff Bachiochi

70 **LESSONS FROM THE TRENCHES**
Debugging Techniques
George Martin

TASK MANAGER **4**
More Than Tinkering
C. J. Abate

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

TEST YOUR EQ **13**

CROSSWORD **74**

INDEX OF ADVERTISERS **79**
February Preview

PRIORITY INTERRUPT **80**
A Handshake and a Future
Steve Ciarcia

NEW CYCLONE IV FPGAs

The new **Cyclone IV FPGA** family adds support for mainstream serial protocols while offering an optimal balance of low cost, low power, and a rich supply of logic, memory, and DSP capabilities. The family offers two variants. Cyclone IV GX devices have up to 150,000 logic elements, up to 6.5-Mb RAM, up to 360 multipliers, and up to eight integrated 3.125-Gbps transceivers supporting mainstream protocols (e.g., GbE, SDI, CPRI, V-by-One), and it has hard IP for PCI Express (PCIe). With low-power consumption and packages as small as 11 mm × 11 mm, these devices address cost-sensitive, small form-factor applications in the wireless, wireline, broadcast, industrial, and consumer markets. Cyclone IV E devices deliver an unprecedented combination of low cost and high functionality, and lower power by up to 25% compared to previous generation Cyclone products.

Integrating transceivers eliminates external component costs and reduces power consumption up to 30% compared to previous generation Cyclone products. This power savings also reduces cost by eliminating the need for heat-dissipation hardware. The devices require only two power supplies, which significantly simplifies PCB design and reduces board space and cost. With a focus on low cost, the smallest Cyclone IV GX device is the industry's smallest FPGA with transceivers.

Production shipments of the EP4CGX15 and EP4CE115—the first Cyclone IV GX and Cyclone IV E devices, respectively—will begin in the first quarter of 2010. Budgetary pricing for the smallest devices—the EP4CE6 and the EP4CGX15—will start as low as **$3** and **$6** respectively for 250,000-unit quantities.

Altera Corp.
www.altera.com

ENERGY-EFFICIENT MICROCONTROLLER

The new 32-bit **EFM32 Gecko microcontroller** family is based on the ultra-efficient ARM Cortex-M3 microcontroller architecture. The EFM32G has been proven to extend battery life by a factor of four, consuming a quarter of the energy required by existing 8, 16, or 32-bit microcontrollers.

Proven to consume less than 180 µA per megahertz while executing real-life code from flash memory, the EFM32G achieves the lowest active mode current consumption of any microcontroller. Its standby current consumption is also the lowest, at typically 900 nA, while running a real-time clock, power-on reset, brown-out detector, full RAM and CPU retention, and less than 20 nA in its deepest sleep mode. Furthermore, the microcontroller's start-up time of less than 2 µs is the industry's fastest. Specific highlights in the EFM32G's rich feature set include: a 4 × 40 segment LCD controller; an eight-channel 12-bit 1-Msps ADC; a brown-out detector; a 32-kHz real-time counter; and a UART capable of 9,600 bps at 100 nA.

Twenty-two different EFM32G microcontroller products will become available over the next few months, in a variety of packages, including QFN32, QFN64, QFP100, and BGA112. The EFM32G operates from a single supply rail of between 1.8 and 3.8 V. The microcontrollers provide up to 128-KB flash memory and up to 16 KB of RAM.

The first products are being offered in QFN64 and BGA112 profiles and are currently sampling with lead customers. Pricing for the initial 32-pin devices starts at **$1.55** in 100,000-piece quantities.

Energy Micro AS
www.energymicro.com

IC DRIVES DIMMABLE CFLs

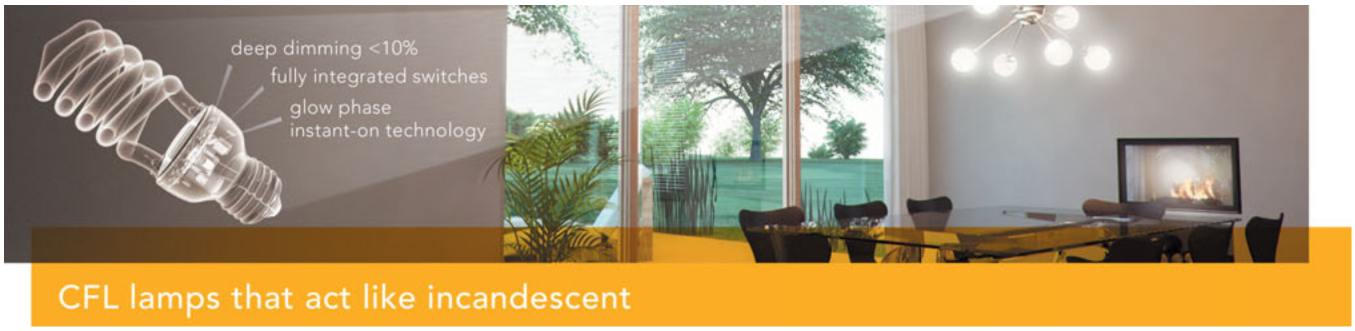
The **UBA2028** is a fully integrated high-voltage power IC that drives dimmable compact fluorescent lamps (CFLs). The UBA2028 enables lighting manufacturers to create high-quality CFLs that rival incandescent bulbs, with quick warm-up times and unprecedented dimming capabilities down to 10% of full light intensity. This driver IC features the highest level of integration available on the market today, enabling more compact designs, highly efficient power conversion, and extended CFL lifetimes beyond 15,000 hours.

The UBA2028 introduces "true dimming" capabilities for a smoother dimming experience without flickering and noise. Further, its high level of integration enables more compact CFL sizes that can fit smaller lampshade designs than ever before.

The device is a 600-V power IC that can drive and control dimmable CFL applications up to 21 W, and features 3-Ω integrated switches. The IC includes a half-bridge power circuit, a dimming function, a high-voltage level shift circuit, an oscillator function, a lamp voltage monitor, a current control function, a timer function, and detection and protection circuits.

The UBA2028 is available with a complete application kit including samples, a datasheet, an application note, and a demonstration board. The UBA2028 costs **$1.75** in 1,000-piece quantities.

NXP Semiconductors
www.nxp.com


deep dimming <10%
fully integrated switches
glow phase
instant-on technology
CFL lamps that act like incandescent

STM32 FAST-START DEVELOPMENT PLATFORM

The **STM32-comStick** is a low-cost, full-featured development platform in a USB stick that enables fast application design for STM32 Connectivity Line microcontrollers. These microcontrollers, including the STM32F105 and STM32F107 families, add Ethernet, USB Device/Host/OTG, CAN, and audio-class I2S peripherals to the advanced ARM Cortex-M3 core, providing low-power 32-bit processing and real-time behavior for embedded applications requiring networking and communications capabilities.

The STM32-comStick includes an unlimited Hitex HiTOP5 IDE/debugger and TASKING C compiler, a built-in GUI allowing for application control, and a 72-MHz STM32F107VC MCU with 256-KB flash memory. Developers can plug the device directly into a PC USB port to evaluate the MCU communication peripherals and build, program, and debug their own applications. No external components, software, or power supply are required.

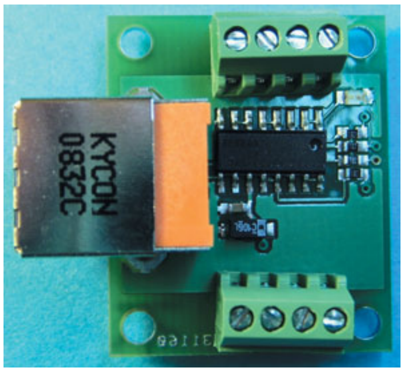
To speed up development, source-code examples demonstrate the microcontroller's Ethernet and USB peripherals. Sample applications include an embedded web server running on a TCP/IP stack, a USB Host mass-storage peripheral, and a file system demonstrating file storage functionality.

In addition to the STM32-comStick, the STM32-PerformanceStick is also available for STM32 developers using 72-MHz Performance Line devices with up to 512-KB on-chip flash memory. The STM32-comStick is available with a resale price of **$69**.

STMicroelectronics
www.st.com

USB SOLID-STATE RELAY MODULE

The **JSB27x series** of USB 2.0-controlled modules are available with two or four solid-state relays. Each solid-state relay is rated for 1 A at 50 VAC or DC. Screw-type terminal blocks are included on the module to provide a quick-and-easy connection to user hardware. Solid-state relays provide several advantages over their mechanical counterparts like increased lifetime, clean bounce-less operation, and silent operation. An easy-to-use software interface includes a Windows driver and a DLL or Class Library API. A Linux support disk is also available.



This small form factor module replaces internal PC-based plug-in cards for use in various test, control, monitor, and measurement applications.

Single-unit pricing for the two relay module starts at $75.

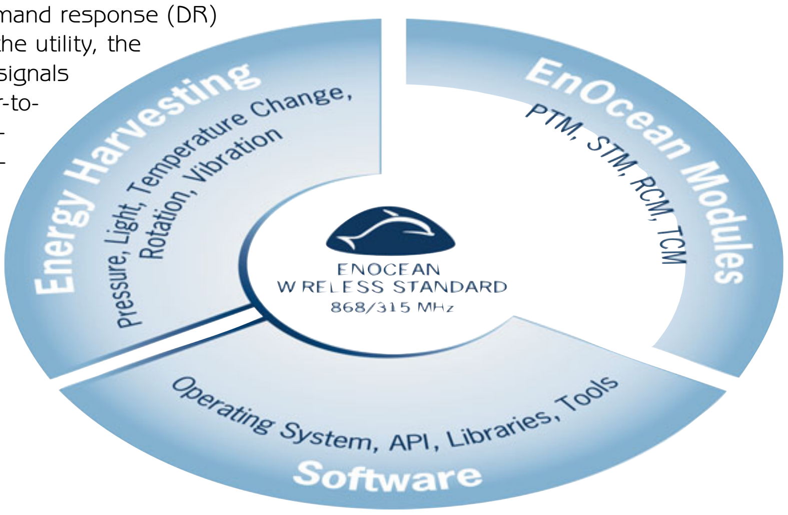
J-Works, Inc.
www.j-works.com

ASIC MODULES DESIGNED FOR ENERGY HARVESTING

A third-generation suite of energy-harvesting wireless modules—based on the **Dolphin ASIC**—is the world's first platform that supports self-powered two-way wireless communications, ultra-low-power sleep modes, and the ability to self-power actuators such as water valves and air vents. Dolphin-based modules consume approximately one-tenth the power of common low-power radio modules. The platform allows OEMs to create energy-autonomous controls able to draw power from multiple ambient sources, such as solar, linear motion, and thermal energies. Integrators can now also develop their own firmware using the new Dolphin Studio—support software for custom firmware development, RF packet monitoring, and C-based code sampling.

The new Dolphin platform enables manufacturers to rapidly develop solutions to suit today's "green" economy. OEMs can now create solutions that transform structures into energy-efficient, responsive, and sustainable buildings. For example, Dolphin-based controls simplify the installation of building automation systems and also provide a flexible and effective way to collect and disseminate utility information such as demand response (DR) events and meter consumption. Once received from the utility, the battery-less controls can seamlessly disseminate DR signals throughout a building while managing day-to-day, hour-to-hour, and minute-to-minute energy consumption. Self-powered two-way communications also enable integrators to build thermostats that react to occupancy and window sensor data in addition to regulating room temperatures.

The EDK 300 development kit is available to help you get started with this new architecture. Please contact EnOcean for pricing.


Energy Harvesting
Pressure, Light, Temperature Change, Rotation, Vibration
EnOcean Modules
PTM, STM, RCM, TCM
ENOCEAN
WIRELESS STANDARD
868/315 MHz
Operating System, API, Libraries, Tools
Software

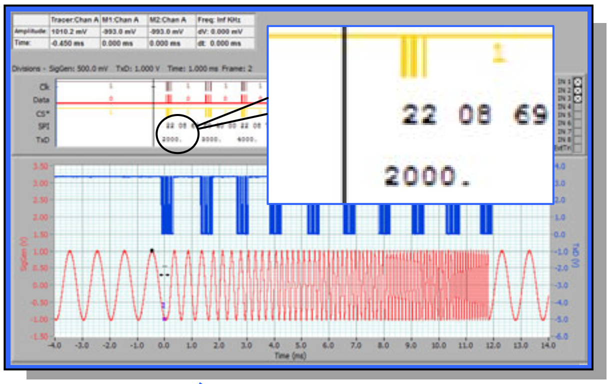
EnOcean, Inc.
www.enocean.com

NPN

cleverscope

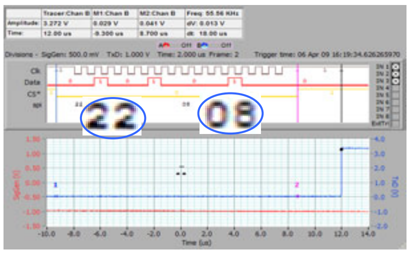
100 MHz MSO 8M Samples 14 bit

Protocol Decoding



Example:
Decode 28 ms of SPI data with 10ns resolution.

Tracking graph - track and zoom:



- **+ Two mixed signal triggers**
- **+ Protocol decoding**
- **+ Spectrum analysis**
- **+ Symbolic maths**
- **+ Custom units**
- **+ Copy & paste**
- **+ Signal generator**
- **+ USB or Ethernet**
- **+ 4 or 8M samples storage**
- **+ 100 MHz sampling**
- **+ Dual 10, 12 or 14 bit ADC**
- **+ Ext Trigger, 8 Digital Inputs**
- **+ 1 MSa/sec charting**

Protocol Decoding.
Decode SPI, I2C, or RS232 based data streams 4M samples long. The decoder is fully configurable, and can save time stamped results to a text file. Write your own plug-in decoder – you can use our open source ones as a starting point.

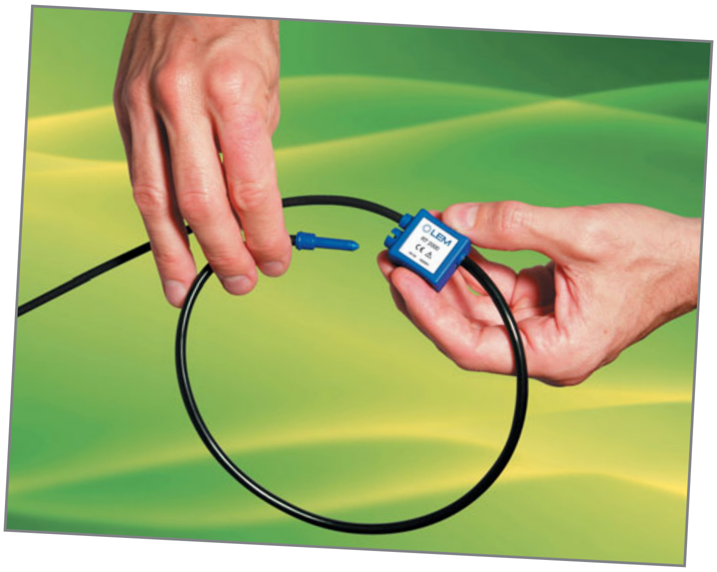
Check out full buffer decodes on our Examples page on the website. Navigating the captured signal is fast and easy; the zoomed Tracking graph follows the Scope graph tracer to see the big picture and the detail simultaneously.

In the USA call:
Saelig com
unique electronics
1-888-7SAELIG info@saelig.com

Our users at Harvard, Stanford, MIT, Caltech, UCLA, UCI, UCSD, UCB, Colorado State, Rice University, U of Washington, NorthEastern .. know you can trust a Cleverscope.

www.cleverscope.com

HIGHLY ACCURATE ROGOWSKI COIL SENSORS



After calibration, the new **RT series** of coil sensors achieves an absolute accuracy of better than 0.65%—including the position error—making them the first split-core Rogowski coils to be suitable for use in Class 1 power devices. An imperfect coil structure induces an unbalanced geometry and increases sensitivity to the position of the measured conductor within the sensor or to the proximity of external electric cables. The RT series of sensors overcomes the problem of asymmetry resulting from discontinuity at the sensor opening, which is inherent in conventional split-core Rogowski coils. LEM engineers have invented a unique, patent-pending magnetic coupling technology that allows a perfect extension of the magnetic flux at the loop opening that compensates for coil asymmetry. In addition, an advanced coil-winding process produces exceptionally regular windings to further enhance sensor symmetry, accuracy, and immunity to electromagnetic interference.

The thin, light, and flexible format of RT sensors enables them to be fitted into applications for which traditional current transformers are typically too heavy and bulky, especially when measuring high currents. Their split-core construction allows them to easily wrap around the conductor without dismantling cables or shutting down operation. The benchmark accuracy of RT series sensors provides enhanced performance in current and power monitors as well as energy meters. The 5-mm gauge RT sensors are currently available with sensing apertures with diameters of 55 mm or 125 mm. They have a five-year guarantee.

Pricing for the coils starts at **$50** each.

LEM
www.lem.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

Distributors of Low-Power RF modules, Zigbee, Bluetooth, WiFi, GPS, GSM/GPRS

MURS Modules
•MULTI-USE RADIO SERVICE•
LICENSE FREE BAND
AVAILABLE RIGHT NOW!

✓ High-Performance ✓ Serial Interface
✓ Low Power ✓ Very Long Range

NEW! QPX1
2 Watt Multi-channel VHF Transceiver

NEW! QPT1
VHF Narrow Band FM 2 Watt Multi-channel transmitter

151.820 MHz, 151.880 MHz, 151.940 MHz, 154.570 MHz, 154.600 MHz

manufactured by RADIOMETRIX

LEMOS INTERNATIONAL
www.lemosint.com
866.345.3667
sales@lemosint.com

ULTRA-LOW-POWER ZigBee-BASED TRANSMISSIONS

The **JN5148** can perform multiple packet transmissions from only 100 µJ of energy harvested from an electromechanical switch. Jennic is the first wireless microcontroller vendor to show ZigBee communications are possible with such low-energy consumption. A single-switch press is all that is required to enable the JN5148 to carry out system startup and initialization, followed by a succession of packet transmissions to increase the probability of packet delivery to the receiver.

A previously announced series of energy-harvesting technology demonstrators utilize thermal, vibrational, RF, and solar energy-harvesting techniques to power end devices in a wireless sensor network. By employing a powered wireless backbone that contains permanently active proxy server routers, energy-constrained end devices can broadcast data when they are able, relying on the backbone to intercept the data messages. This device enables the use of the same approach but with even lower energy budgets.

The JN5148 wireless microcontroller integrates a high-performance, 32-bit RISC CPU core with mixed-signal peripherals and an IEEE 802.15.4, 2.4-GHz transceiver. The 98-dB link budget supports indoor communication over distances of up to 50 m, while 128-bit AES encryption ensures a high level of security. It offers industry-leading current consumption of just 15 mA when transmitting, 18 mA when receiving, and 200 nA in sleep mode.

The JN5148 is priced at **$5.99** each in quantities of 500 pieces.

Jennic
www.jennic.com

NPN

CIRCUIT CELLAR Test Your EQ

Edited by David Tweed

Problem 1—*Whatever happened to "Bubble Memory"? It seems to have disappeared without a trace.*

Problem 2—*What's the difference between an Ethernet hub and an Ethernet switch, and what are the key tradeoffs?*

Problem 3—*How exactly does a multi-master I^2C bus work?*

Problem 4—*An alkaline AA cell has a nominal operating voltage of 1.5 V and a capacity of about 2,000 mA-h. It weighs about 23 grams. By how much does its weight (mass) change between being fully charged and fully discharged?*

a. Not at all
b. 0.12 nanogram
c. 41 micrograms
d. Other

For reference, the electron charge-to-mass ratio is about 1.76 x 1,011 C/kg, and the speed of light is about 3 x 108 m/s.

Contributed by David Tweed

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

Got Serial, Need Network?

Volume Discounts Available

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

PCBCORE

China PCB Supplier
(Prototype thru Production)

✓ 1-layer up to 30-layer
✓ Cost and quality
✓ On time delivery
✓ Dedicated service
✓ Instant Online Quote & Order
..... Day and Night

No minimum quantity - 1 piece is welcome
Check our low price and save big $$$...

86(571)86795686 sales@pcbcore.com

www.pcbcore.com

by Hans-Dieter Paul

Teletext-Based TV Interface

This design uses a TV's built-in teletext decoder for displaying colored text and pseudo-graphics on the screen. It serves 15 pages that are continuously sent to the TV along with a video signal. If you don't live in Europe where teletext is used, you can apply these design techniques to your own server development and graphics-related projects.

Many TVs in Europe have a built-in teletext (TT) decoder. And even some of the newest widescreen LCD TVs are compatible with the decade-old TT standard. I use a TT decoder to display text and graphics in color on my TV screen. In this article, I'll describe what I call my "Home Teletext Server," which currently "serves" 15 pages (see Photo 1). I intend to upgrade the application over time.

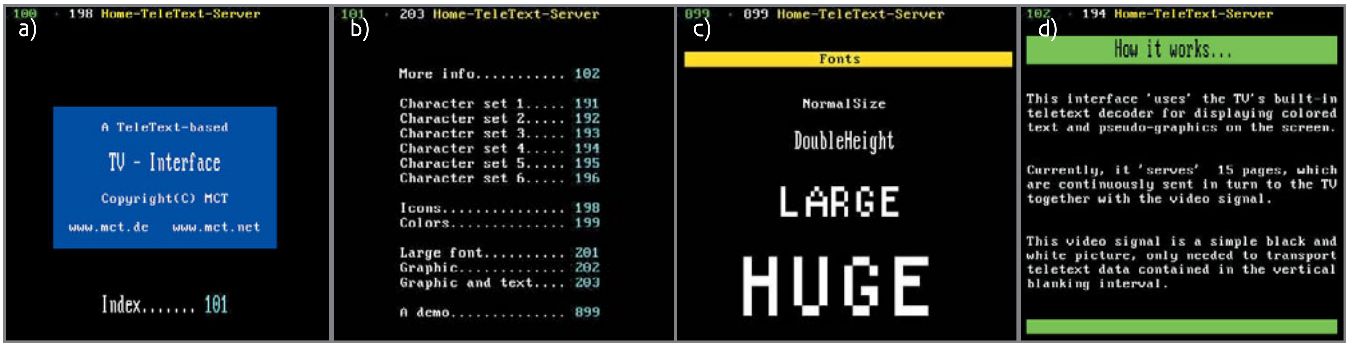
Note that the software for this project requires a PAL standard TV with a built-in TT decoder. TT is only used in Europe. It is not defined for NTSC, and it exists only in areas using the PAL video-encoding standard. If you live in Europe, you'll find this project immediately applicable. If you don't, I'm confident you'll find the details thought-provoking and useful.

THE HISTORY

The NXP Semiconductors LPC2138 is an excellent—perhaps my *favorite*—controller. It has an ARM core, 60-MHz clock, and powerful peripherals. In addition to a 32-bit timer system, a synchronous serial port (SSP), and a 10-bit ADC—which are also found on other LPC21*xx* family members—the LPC2138 incorporates a 10-bit DAC, which extends the range of possible applications.

My first application was a TV terminal that simply displayed characters on a screen. I had to generate an analog signal with at least three levels: sync, black, and white. So what I needed was a DAC—low-resolution, indeed, but yet a DAC. It's easy to build a simple, low-resolution DAC with general-purpose I/O (GPIO) and some resistors. But I am less skilled at soldering than at programming, so I've always tried to do without external "hardware." Small wonder that I started my project with an LPC2138.

A software character generator and a simple interrupt service routine (ISR) was all that I needed to complete my TV terminal without any external components. But the resolution was poor. For each pixel, the DAC had to be set to the pixel's "color" value (black or white). Even on a fast CPU this takes some time and limits the number of



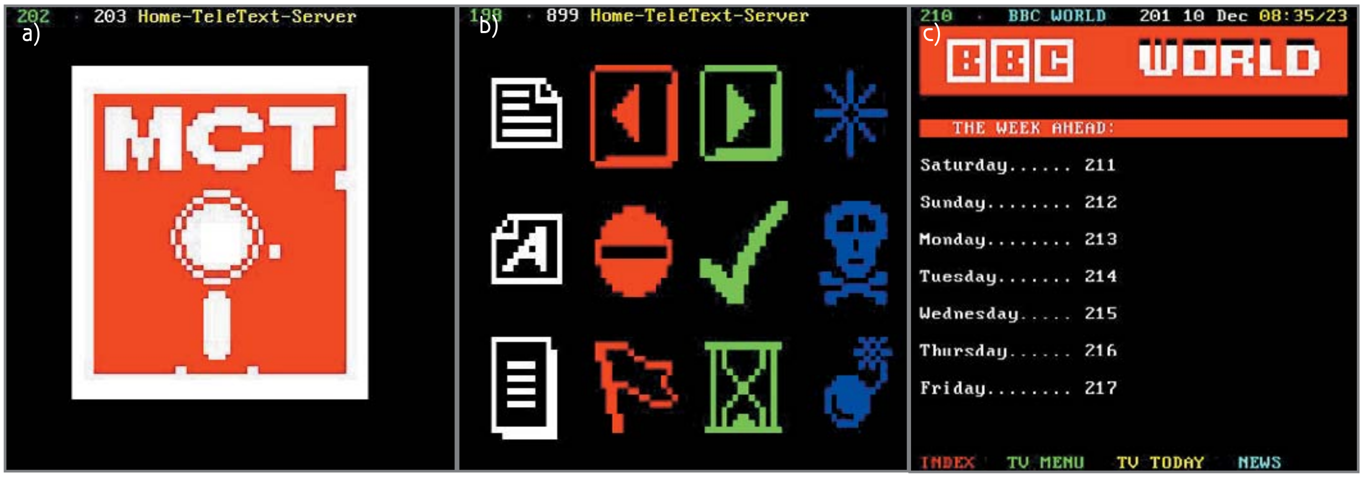
```
100   198 Home-TeleText-Server

      A TeleText-based
      TV - Interface
      Copyright(C) MCT
   www.mct.de   www.mct.net

      Index....... 101
```

```
101   203 Home-TeleText-Server

    More info........... 102

    Character set 1..... 191
    Character set 2..... 192
    Character set 3..... 193
    Character set 4..... 194
    Character set 5..... 195
    Character set 6..... 196

    Icons............... 198
    Colors.............. 199

    Large font.......... 201
    Graphic............. 202
    Graphic and text.... 203

    A demo.............. 899
```

```
899   899 Home-TeleText-Server
                Fonts

             NormalSize
            DoubleHeight
               LARGE
               HUGE
```

```
102   194 Home-TeleText-Server
          How it works...

This interface 'uses' the TV's built-in
teletext decoder for displaying colored
text and pseudo-graphics on the screen.

Currently, it 'serves'  15 pages, which
are continuously sent in turn to the TV
together with the video signal.

This video signal is a simple black and
white picture, only needed to transport
teletext data contained in the vertical
blanking interval.
```

Photo 1—Here you see the Main page (a), Index page (b), Text demo page (c), and a short introduction (d).



a)
202 · 203 Home-TeleText-Server
MCT
b)
198 · 899 Home-TeleText-Server
c)

```
210  ·  BBC WORLD    201 10 Dec 08:35/23
BBC   WORLD
 THE WEEK AHEAD:
Saturday...... 211
Sunday........ 212
Monday........ 213
Tuesday....... 214
Wednesday..... 215
Thursday...... 216
Friday........ 217
INDEX   TV MENU   TV TODAY   NEWS
```

Photo 2—Here you see pseudo-graphics demo pages (a and b), as well as an example of a commercial teletext page (c). (Photo 2c Source: The BBC)

pixels per line.

In the second version of my TV terminal design, I used the SSP to shift out the white pixels, but I still had the DAC create the sync and black levels. Unfortunately, this method required some external "hardware": a diode between the DAC and the SSP outputs. I was lucky enough to find a 1N4148 at the bottom of my desk drawer, and decided that I could live with this hardware extension. Somehow, I managed to fit the diode into the cable (needed anyway to connect the TV). The resolution improved a lot. I also added some functions to the software that enabled me to display text in different sizes and simple graphics like lines, circles, and bitmaps. (TV terminal examples are available at www.mct.net. Click "Download" and then "ECO-C-arm sample programs.")

I should have been happy. It's not hard to guess why I wasn't: I wanted color! While searching the Internet for a solution, I learned something. A black and white video signal is one thing; a color video signal is another. For color, a quadrature modulated subcarrier is needed. A few cycles of this color carrier have to be included in the signal (burst) to synchronize the color decoder's oscillator. Besides, the PAL standard requires phase-shifting between lines.

A really useful application couldn't be realized without dedicated hardware support. As soldering became nearly inevitable, I started thinking hard about a way to make the TV display color without such a complex signal.

THE SOLUTION

I live in Germany, where PAL is the TV encoding standard. In addition to the TV programming, the stations offer an extra service, called Videotext (or TT), as in other European countries. This service provides information of all kinds based on colored text and pseudo-graphics (see Photo 2).

Virtually every TV sold in Germany has a built-in TT decoder. TT information is sent together with the TV program, but it is "hidden" in the video signal. The TT decoder shows this information only on demand; it's usually initiated using the TV's remote control.

Did I say colored text? Yes, TT encoding allows RGB colors with a color depth of 1 bit per component, which makes eight colors, including black and white (see Photo 3). And here's an interesting fact: TT doesn't require a color video signal. A simple black-and-white video signal can be used to transport TT data. Even a signal producing a black screen (i.e., with no "picture" information at all) can contain TT data, which the decoder makes visible. In other words, the TT decoder creates the picture specified by the coded TT data—in color, if desired. The decoder is the TV interface's "external hardware." Such external hardware has some

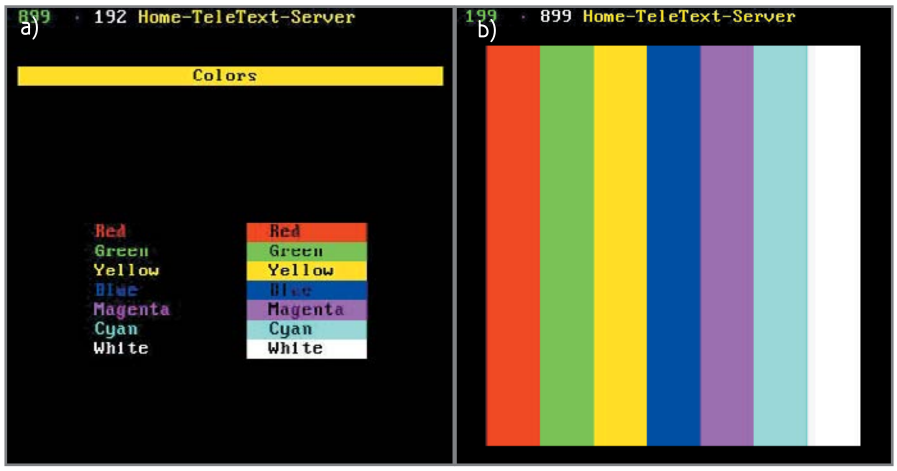


Photo 3—Teletext encoding allows for RGB colors with a color depth of 1 bit per component. Here you see demo pages with foreground and background colors displayed horizontally (a) and vertically (b).

advantages: it's ready-to-use at no extra cost, and, of course, it doesn't need soldering. Years ago, these decoders were hardware options for TVs. But in the days of TV-on-a-chip, you can't buy a TV without a (software) TT decoder (in Germany at least).

I consulted the TT specification for the encoding details. What followed were some late nights of programming and debugging.

THE IMPLEMENTATION

According to the specification, even a "simple" black-and-white video signal is a "wild" mix of pulses. A simplified form is shown in Figure 1. This represents one TV screen line. One full picture has 625 lines, written interlaced in two fields of 312.5 lines each. A vertical sync signal (composed of small pulses in the first few lines of a field) indicates the beginning of a new field and defines its starting position on the screen.

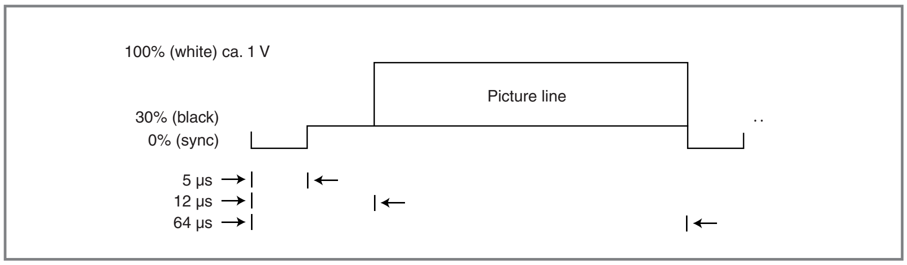


Figure 1—This is a simplified video signal (PAL norm, no color).

The software uses a much simpler form: noninterlaced mode (i.e., the same field written twice to the same position). To keep timing as close as possible, 312 lines per field are used (which results in a slightly higher vertical frequency) and the vertical sync pulse simplifies to a long low pulse in the first two lines of each field. Luckily, TVs are tolerant. They can "swallow" this without a hiccup.

A black-and-white picture requires at least three levels: sync, black, and something between black and white. The DAC provides the sync and black levels. The SSP is used for pixel output. It is a kind of high-performance serial peripheral interface (SPI). Both interfaces are based on shift registers, but the SSP can be clocked at higher rates. In contrast to the SPI, the SSP has built-in FIFOs allowing to create continuous bit streams back-to-back without gaps.

The video signal is generated in a timer ISR, which activates every 64 µs. The DAC is then set to the required levels for a black line, and the first two lines of each field are held on the sync level (0). The pixel data (0 for black and 1 for white) comes from the SSP MOSI1, which must be connected to the DAC output using a diode (see Figure 2). When a bit is 1, the DAC output is simply pulled high. The diode blocks 0 bits to prevent them from pulling the signal down to the sync level. So, the 0 bits are held on the black level coming from the DAC.

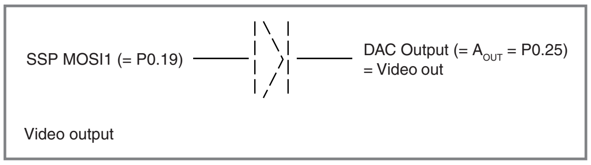


Figure 2—Video output. The pixel data (0 for black and 1 for white) comes from the SSP MOSI1, which must be connected to the DAC output using a diode.

When I first tested this concept by displaying some lines of text entered from the serial interface, I was disappointed. Although the text was readable, the lines were jittering! I found the solution for this problem in Ilya Mamontov's article, "TV-Based Oscilloscope" (*Circuit Cellar* 196, 2006). The jittering was caused by varying interrupt latencies. Most of the time, the main program does nothing but remain "busy waiting." When an interrupt occurs, the CPU completes the current instruction prior to servicing the interrupt request. As these processes run asynchronously, the CPU is interrupted at different stages during instruction execution.

To put it simply, the interrupt is at times serviced a little earlier and sometimes a little later. Even though I'm talking about nanoseconds, these varying delays become visible as jitter. The solution is as simple as it is effective. Instead of remaining busy waiting, the CPU enters Sleep mode. This makes interrupt latencies constant (it always takes the same time to get out of this idle state). When the jitter is gone, the screen lines are aligned vertically. For TT lines (as I'll describe soon), interrupt latencies are

"I find it comfortable to sit back, watch TV, and occasionally switch to TT. I can select pages—which are nicely organized in magazines—like the real TT. I can do all this via remote control. It's just like accessing the program guide, the latest sports results, or the weather report from a real station. It's just another channel. I call it my 'Home-TeleText Channel.'"

The Next Generation of In-Circuit Debugging

The NEW MPLAB® ICD 3

The MPLAB ICD 3 In-Circuit Debugger is Microchip's most cost effective high-speed debugger for Microchip Flash PIC® Microcontrollers (MCU) and dsPIC® Digital Signal Controller devices. It debugs and programs PIC MCUs and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (IDE).

- In-Circuit Debugging for PIC MCUs and dsPIC DSCs
- Full-speed, real-time emulation
- Source debugging, stopwatch, complex breakpoints and in-circuit programming
- MPLAB IDE compatible
- Firmware upgrade via MPLAB IDE
- Overvoltage and undervoltage protection
- High Speed USB 2.0 (480 Mbps)
- Target power, up to 100 MA
- Internal 1 MB memory buffer for increased download speed

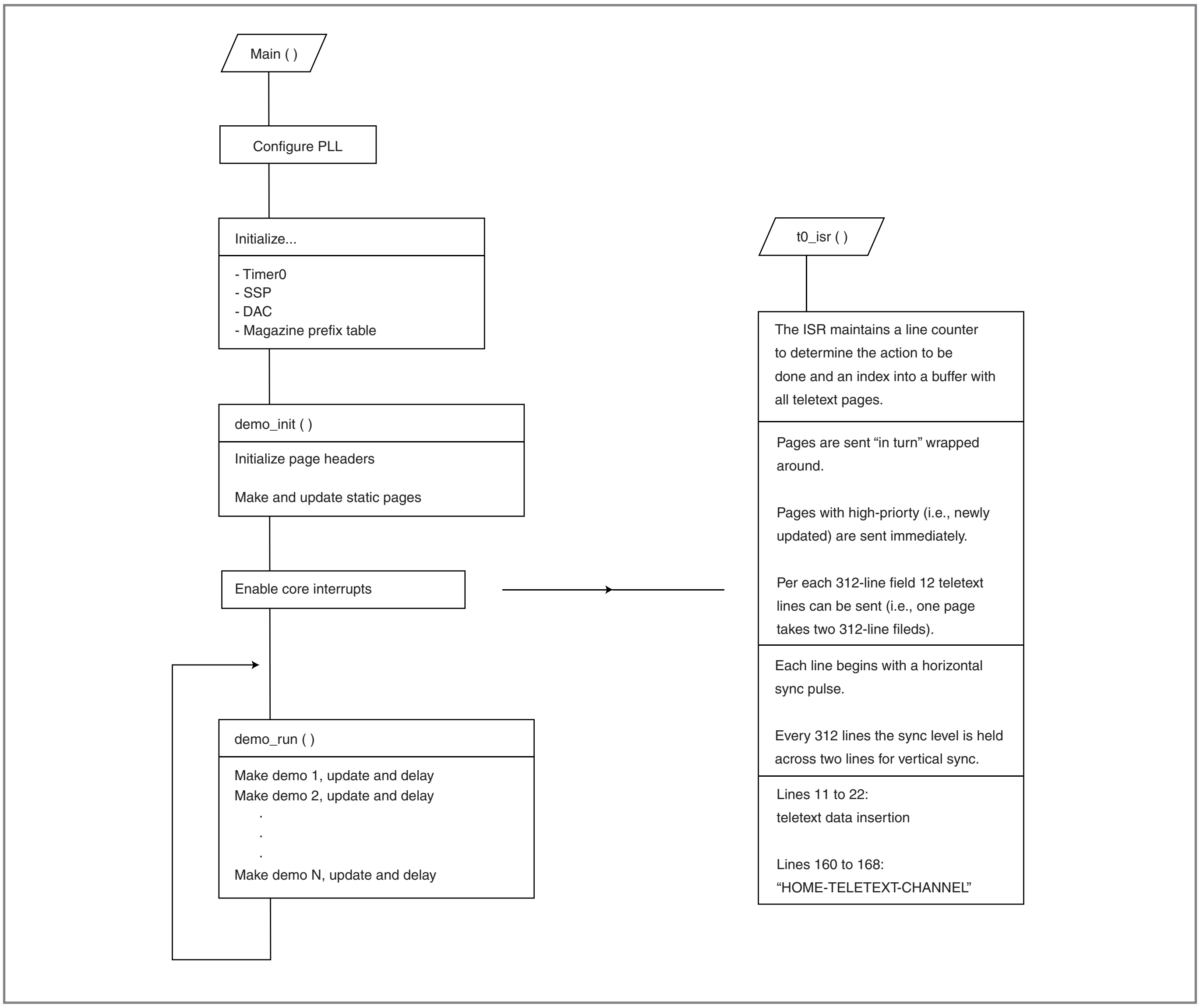
MPLAB® ICD 2 RECYCLE

Return your old MPLAB ICD 2 and receive 25% off the new MPLAB ICD 3, MPLAB REAL ICE or PICkit™ 3 Debug Express. For more information on this offer, please visit:

www.microchip.com/ICD2recycle

www.microchip.com/ICD3

MICROCHIP



Main ( )
Configure PLL
Initialize...
- Timer0
- SSP
- DAC
- Magazine prefix table
demo_init ( )
Initialize page headers
Make and update static pages
Enable core interrupts
demo_run ( )
Make demo 1, update and delay
Make demo 2, update and delay
.
.
.
Make demo N, update and delay
t0_isr ( )
The ISR maintains a line counter to determine the action to be done and an index into a buffer with all teletext pages.
Pages are sent "in turn" wrapped around.
Pages with high-priorty (i.e., newly updated) are sent immediately.
Per each 312-line field 12 teletext lines can be sent (i.e., one page takes two 312-line fileds).
Each line begins with a horizontal sync pulse.
Every 312 lines the sync level is held across two lines for vertical sync.
Lines 11 to 22: teletext data insertion
Lines 160 to 168: "HOME-TELETEXT-CHANNEL"

Figure 3—This is the program's straightforward structure.

less important. Such lines are not visible. And besides, the decoder synchronizes to the start of teletext data within a tolerance window at the beginning of each TT line.

How is TT squeezed into this obviously packed scheme? The TV cannot display all the lines. Some of them lie in the vertical sync phase—which is called the vertical blanking interval (VBI)—and are not visible. The reason for this dates back to the days of the cathode ray tube (CRT). But these lines can be used to transmit additional information. And this is the exact place where TT data is hidden.

A full explanation of how TT works is beyond the scope of this article. But let's review some basic ideas.

Some of the unused invisible lines transmitted in the VBI contain coded data instead of picture pixels. Each line contains 45 bytes. This equals a data rate of almost 7 Mbps (6.9375 Mbps to be precise). Forty bytes hold "real" TT data. The remaining 5 bytes are for syncing and addressing. Per picture, 24 TT lines (12 lines per half picture) are transmitted. This gives a 40 × 24 character TT page including a header with some control bytes used by the decoder (e.g., hide header and character set). The pages are organized in "magazines" (1 to 8) with 100 pages each. And the header data is partially hamming-coded. The rest of the page consists of 7-bit data with odd parity. All TT data is sent LSB first.

What the decoder makes appear on the TV screen in the end is a rather difficult thing to determine. It depends on the header control bytes and on the sequential character attributes. "Sequential" means that attributes (e.g., size and color) are also stored in the page, just like normal characters. Most of them get displayed as "space," and take effect only for the following characters, until the end of the line.

The TT lines get inserted into the video signal in the timer ISR. In fact, the ISR has nothing else to do because no picture information is needed. Yet, I decided to include one picture line, displaying the text "HOME-TELETEXT-CHANNEL" centered on the TV screen. This looks better than a black screen, and gives the user a hint to switch on his TV's TT function.

Each TT line starts with two sync bytes and a start code, followed by the magazine prefix (the hamming-coded magazine and row addresses of this line). Then the 40 bytes of data are transmitted (7 bits with odd parity). Line 0 of each page, the header, contains the magazine, page and subpage numbers, and the page control bits. Subpage numbers are not used and are always 0. Only the "hide header" control bit (suppressing the header line on the screen) and the 3-bit character set field (allowing six plus two reserved character sets) are used. This part of the header is also hamming coded, and the remaining 32 bytes can be used as a page title (7 bits with odd parity, like the rest of the page).

All data has to be prepared by the main program to be ready to send for the ISR. There is no time left to add parity, for example, on the fly. But there's still a big problem: the TT data rate. As I already mentioned, the TT data rate is 6.9375 Mbps. This clock cannot be derived from the 60-MHz system clock. So the quartz had to be changed. Using a 13.875-MHz crystal solves the problem. Multiplied by four through the LPC2138 on-chip PLL gives a 55.5-MHz system clock. The SSP is configured to divide this clock by eight—and voila, exactly 6.9375 MHz. My LPC2138 board has a socket for the quartz. And, as I used to keep some spare crystals in my desk drawer, that was easy.

The very last problem occurred during program development. The timing uses a fine-tuned delay loop. Sometimes, after having modified the program, I had to readjust the delay values. But this was more an annoyance than a problem. It was uncomfortable. Finally, I remembered that the LPC2138 accesses the flash memory through the memory accelerator module (MAM) allowing it to execute code from flash memory really fast. (In fact, in contrast to other controllers, there is no need to run code from RAM for higher speed.)

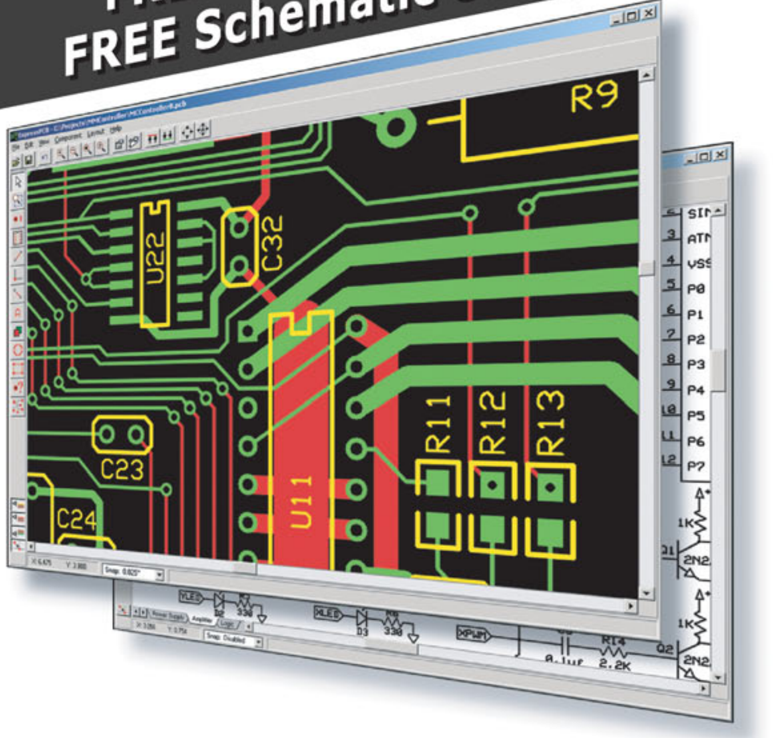
But this kind of pipelining or caching also makes code execution time position-dependent and therefore unpredictable. As I found by experimenting, aligning the loop on a 16-byte boundary followed by a NOP instruction made the delay values independent of the rest of the code. Also, it made the ISR (located immediately after the loop) perform best. So I have reason to believe that my theory came close the truth (and might also be relevant for other programs, which require some "tuning"). This alignment and the NOP are, by the way, the only two assembler lines in the code. (Of course, the macro ENABLE_INTERRUPTS, which is provided by a system include file, also hides some assembler lines because this cannot be done in C.)

PROGRAM STRUCTURE

The software is divided into several files. The tt_serv.c file is for the TT server. It contains `main()` and provides some functions for page update and format conversion—and, of course,

$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

the timer ISR, which is the central part. The tt_full_demo.h file is a sample application, showing all program features. The tt_demo.h file is a short demo (skeleton for applications). The vt.h file is a set of functions to display text and graphics (virtual terminal). Both font6x9.h and font8x9.h are font files, and icons.h includes sample icons.

The sources are well documented. The program structure is straightforward (see Figure 3).

How are pages made? The answer is in the tt_full_demo.h file. The numerous examples show how to create text, simple graphics, and bitmaps. Pages are composed "offline" in a separate buffer. When a page is complete, `update()` is called, which copies it to the transmit buffer at the same time converting it to the TT format. (The bit order of bytes is reversed and odd parity added.) The page gets marked high-priority, and the ISR sends it immediately.

THE REALIZATION

Completing this project requires any board with an LPC2138 (see examples at www.mct.net/product/lc2138.html). Replace the quartz with a 13.875-MHz crystal and connect a diode between MOSI1 (P0.19, anode) and Aout (P0.25, cathode). Then connect Aout and GND to a TV's video input. On the SCART connector, these are pins 20 (video) and 18 or 21 (GND). Switch the TV to video input and activate TT.

Of course, the program must first get into the controller. The compiled program is supplied as a hex file (tt_full_demo.hex), which you can program serially to the flash memory using a freely available utility (www.flashmagictool.com).

There is also a binary image (tt_full_demo.img) that you can program, using the demo version of the ECO-C-arm compiler. After installation, first set the used COM port under Terminal and then choose Download File.

The source of the full demo (tt_full_demo.h) is too large to be compiled with the ECO-C-arm compiler demo. Only the short demo (tt_demo.h) can be compiled and it also leaves room for modifications. Note, however, that programs compiled with the ECO-C-arm compiler demo cannot be run stand-alone. They can be started only from the PC. The short demo gets included in tt_serv.c. Those who have the full version of the ECO-C-arm compiler can include tt_full_demo.h instead.

To compile the program, make a new project, copy all the files from the "src" folder into the project folder, set the correct target (LC2138), and add tt_serv.c to the project file list. (Go to "Getting Started" under Help.)

THE FUTURE

What can you do with the TV interface? Many applications need a display connected to a microcontroller. A small LCD is often sufficient or desired. But sometimes such a display is chosen only because the controller lacks a graphic adapter. A large color display usually requires more effort.

What is this interface good for? Lots of inexpensive LCD computer monitors are widely available, offering far more resolution. Why not design an interface for VGA?

Having a computer monitor—inexpensive or not—in the living room is not for everyone. It is an additional screen that needs energy, and it has to be operated, with a keyboard or similar kind of control.

Why not put an extra computer in the living room?

I find it comfortable to sit back, watch TV, and occasionally switch to TT. I can select pages—which are nicely organized in magazines—like the real TT. I can do all this via remote control. It's just like accessing the program guide, the latest sports results, or the weather report from a real station. It's just another channel. I call it my "Home-Tele-Text Channel."

Currently, I'm working on an Ethernet module with a SPI. The prototype is ready to link my home TT server design to the LAN. This will enable my office computer to continuously update the pages with current information. But I am sure it will take more than one late-night session to get that software working. ◘

Author's note: I thank the BBC for granting me permission to use Photo 2c.

Hans-Dieter Paul (hd.paul@arcor.de) studied computer science at the Technical University of Berlin (TU–Berlin). In 1984, he founded MCT Paul & Scherer Mikrocomputertechnik GmbH with his partner Erhard Scherer (www.mct.net). Since then, he has been developing hardware and software for industrial applications.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/234.

RESOURCES

Institute for Radio Technology (IRT), "Technical Guideline ARD/ZDF No. 8R4 Teletext Specification," Munich, Germany, 1986.

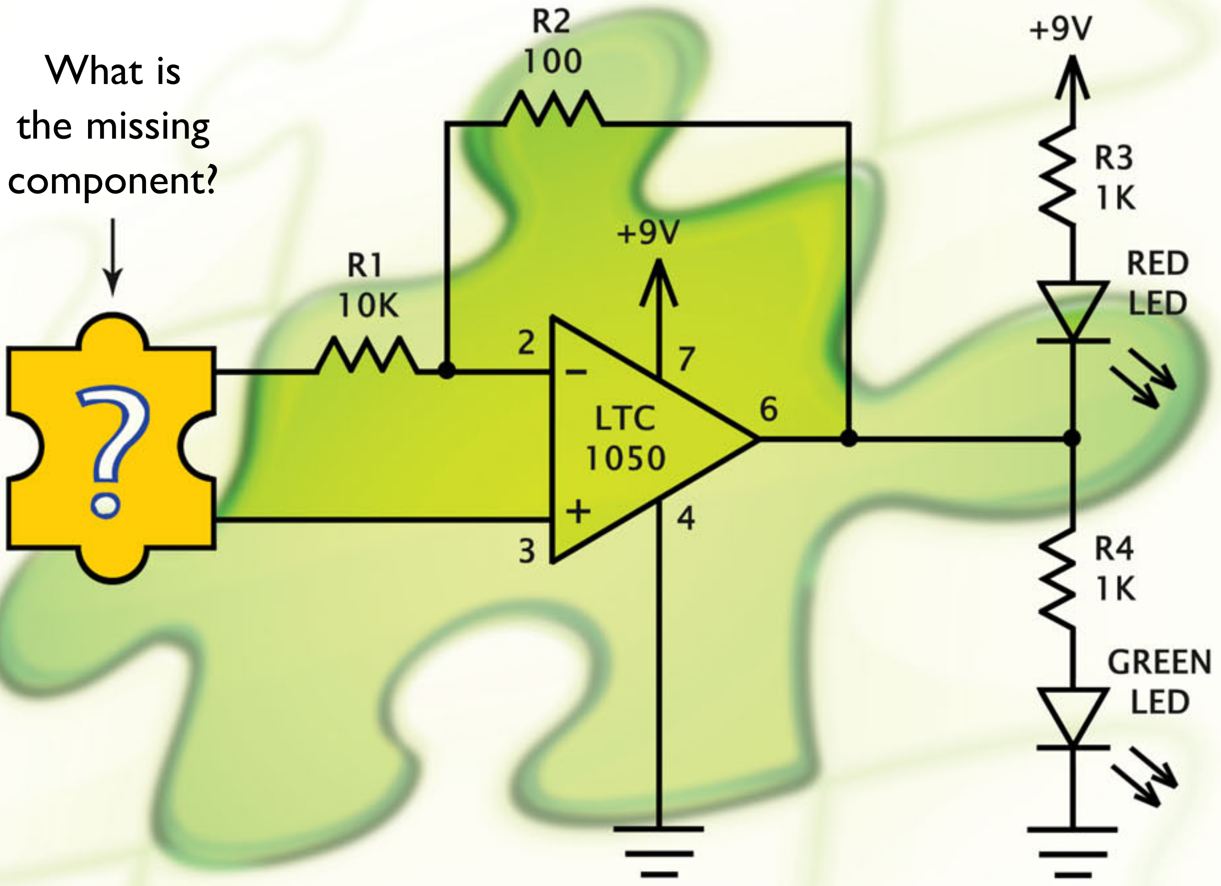
I. Mamontov, "TV-Based Oscilloscope," *Circuit Cellar* 196, 2006.

SOURCES

ECO-C-arm compiler
MCT | www.mct.net

LPC2138 Microcontroller
NXP Semiconductors | www.nxp.com

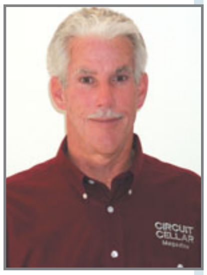
Are you up for a challenge?



Industry guru Forrest M. Mims III has created a stumper. Video game designer Bob Wheels needed an inexpensive, counter-clockwise rotation detector for a radio-controlled car that could withstand the busy hands of a teenaged game player and endure lots of punishment. Can you figure out what's missing? Go to www.Jameco.com/unravel to see if you are correct and while you are there, sign-up for our free full color catalog.

JAMECO®
ELECTRONICS
1-800-831-4242

by Tom Cantrell



SoC with a Capital "P"

A Look at the PSoC 3 and PSoC 5

If you thought the first PSoC was great, wait until you get your hands on the PSoC 3 and PSoC 5. These affordable new PSoCs—with which you can define your own programmable logic functions—operate over a wide voltage range and are extremely easy to use. Now you just need to choose an application.

Has it really been almost nine years since I first covered the Cypress Semiconductor PSoC ("SoC Hop," *Circuit Cellar* 128, 2001)? The PSoC's combination of a flash MCU, programmable logic (notably including analog), powerful yet easy-to-use configuration software, and low price made it revolutionary at the time. And surprisingly, for pretty much the same reasons, the PSoC remains revolutionary to this day. There's still nothing quite like it, a situation in which would-be competitors' losses have meant gains for the PSoC. According to Cypress, their "little chip that could" has done just that, with a skyrocketing share of the 8-bit MCU market, blowing past any number of larger and more experienced contenders.

The word on the street had been that Cypress had big plans for a PSoC upgrade in the works. Now it's official with their recent formal announcement of next-generation parts that bring PSoC panache to the 8051-based PSoC 3 (available now) and ARM Cortex M3-based PSoC 5 (available Q1 2010) cores (see Figure 1).

Piggybacking on two of the most popular MCU architectures can only add fuel to the PSoC fire. But there's more to the latest versions than just different opcodes. Let's take a closer look at the '51-based PSoC 3 to see how it retains the essence of the original PSoC while adding significant new capabilities.

DÉJÀ VU MCU

By now, most embedded designers are familiar with the '51, at least those who haven't been on another planet for the last 30 years. For those who need a history lesson, check out a column I wrote back in 2004 ("'51 Flavors," *Circuit Cellar* 163) in which I reminisced with John Wharton, one of the original '51 architects.

Although I'm a bit of a cynic about ostensible "advances" in computer architecture, it would be unrealistic to say the '51 is as "good" a design as more modern MCUs, such as the Atmel AVR or Renesas Technology H8. Indeed, the '51's continued success proves that, in the 8-bit MCU market, "good enough" is the benchmark.

At the same time the '51, warts and all, is arguably a step up from the original PSoC's proprietary M8 core. Indeed, I noted back in "SoC Hop" how the M8 reminded me of the 6502 (of Apple-I fame), one of the few 8-bit architectures that actually predates the '51.

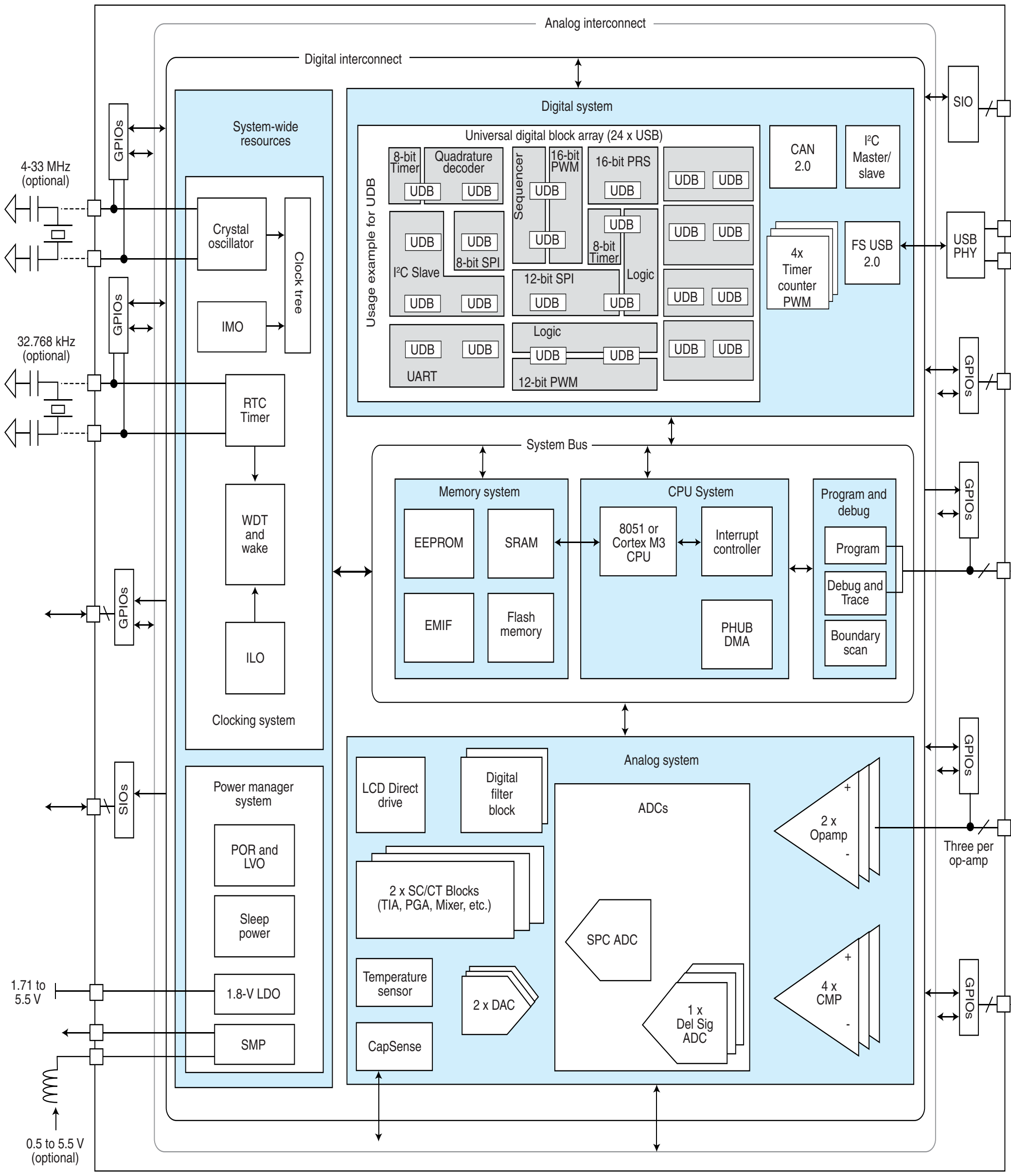


Figure 1—It was the "P" (hardware and software programmability) in "PSoC" that made the original a hit. The new PSoC 3 and PSoC 5 take the concept even further with higher-performance cores ('51 and ARM Cortex-M3) and the ability for users to define their own programmable logic functions.

The '51 story is less about the bits and bytes and more about the fact that the '51—with its design essentially in the public domain (ironic given Intel's stance on intellectual property)—is a de facto "People's Micro." The Cypress strategy becomes clear when you realize the '51 and ARM Cortex-M3 are "open" architectures supported by a wide base of silicon suppliers, in stark contrast to the M8 and other "sole source" alternatives.

Even as I downplay the technical aspects, keep in mind that the PSoC 3's '51, like other modern incarnations (e.g., NXP, Atmel, and Silicon Labs), is quite improved from the disco-era original. For instance, the old-school 12 clock-per-instruction (CPI) design was put to rest long ago with hopped-up implementations delivering tens of MIPS instead of just one. Another example is the way the PSoC 3 overcomes the original's reluctance to venture off chip with the addition of an extra data pointer register and a relatively high-performance (up to 24-bit address, 8- or 16-bit data) external memory interface.

Rather than get too fancy with the '51 architecture itself—and risk a "New Coke" fiasco in the process—Cypress upgrades the basic model with some sexy chrome and tailfins. Adding a high-speed 24-channel DMAC and low-latency vectored interrupt controller is a better way

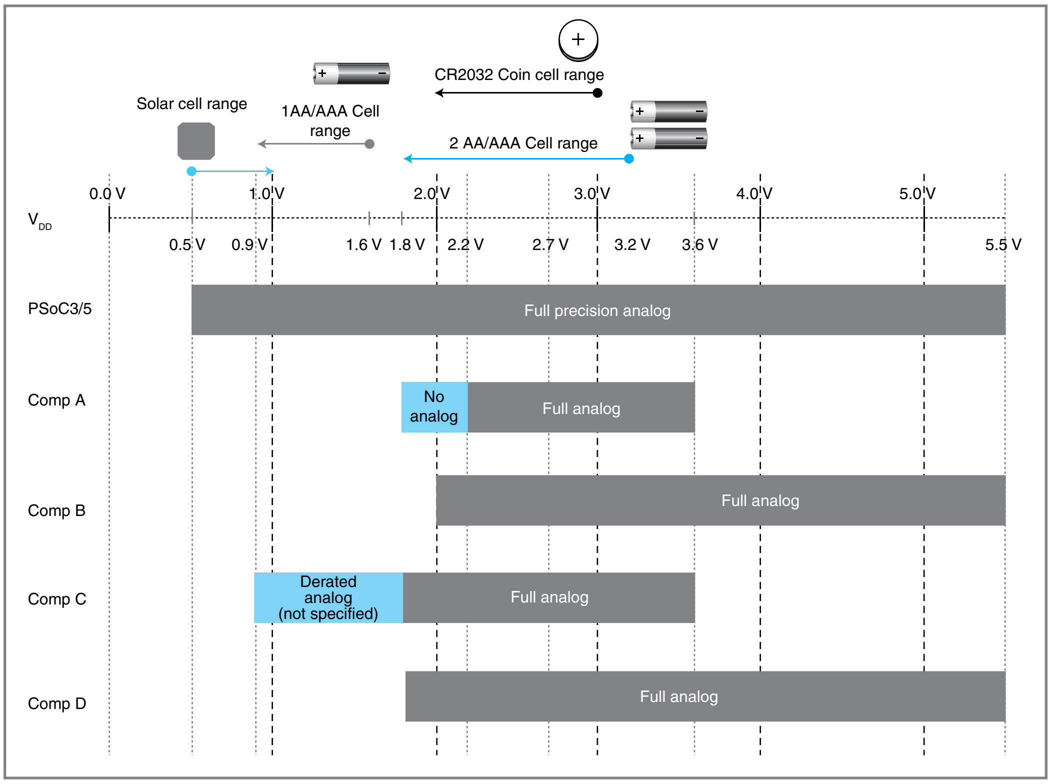


Figure 2—Thanks to an on-chip voltage booster, the new PSoCs are capable of operating over a very wide voltage range, encompassing single-cell (i.e., 0.9-V) applications to 5-V legacy interfaces. Better yet, the chips remain surprisingly functional (e.g., clock rate, analog accuracy) across the entire range. (Source: Cypress Semiconductor)

to deliver real-world performance than fooling around with the ’51 recipe itself.

GLUE BE GONE

The programmable nature (i.e., MCU and analog/digital hardware) of the PSoC 3 is the star and deservedly gets most of the attention. But, just like the latest box office hit, behind every big star is an army of crew, assistants, and extras that brings the magic to the big screen.

In the real world of deeply embedded apps, the mundane matters as much as the bells and whistles. Fancy features are all for naught if they bring along baggage of extra chips, more board space, higher power, and bigger budgets.

I’m making a point of digressing because as I perused the PSoC 3 datasheet I found myself appreciating all manner of details that, though seemingly small, reflect an understanding of the “gotchas” that challenge designers in the trenches. For instance, the trend towards ever-lower-voltage parts is all well and good, but there are downsides, including reduction in dynamic range for attached analog sensors and the fact many embedded apps still need higher voltage interfaces, up to 3.3 V or even 5 V.

The PSoC 3 delivers the best of both worlds. Thanks

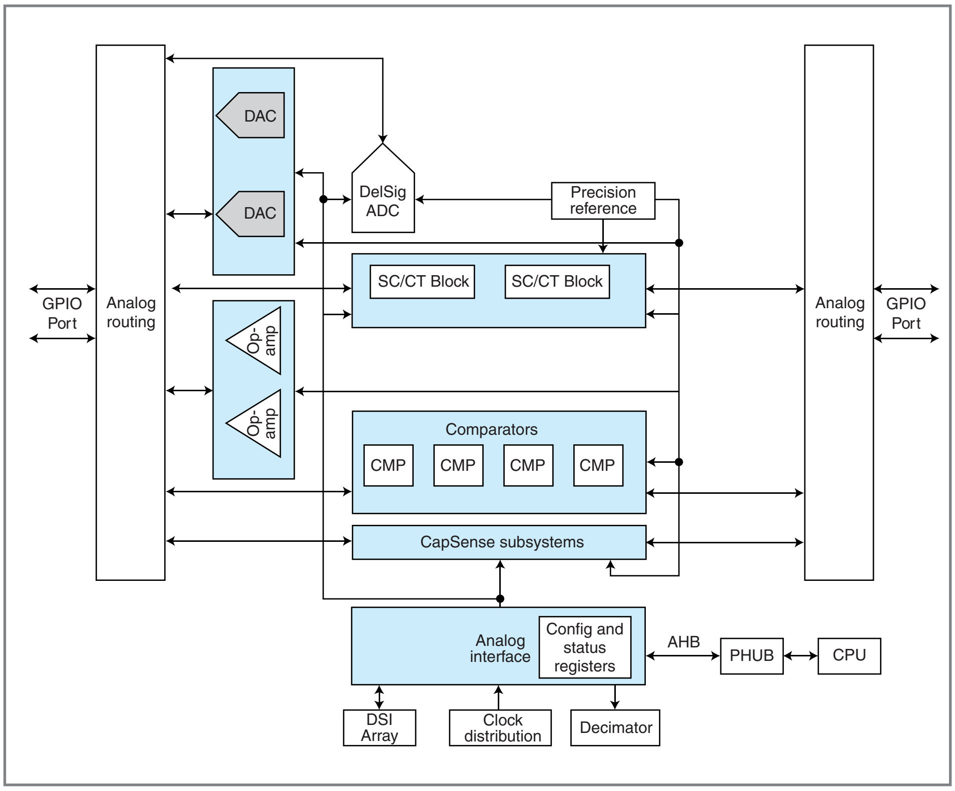


Figure 3—Ones and zeros are well and good, but when it comes to adding value to blue-collar chips, “It’s the analog, stupid.” The PSoC obliges with a programmable analog subsystem comprising a collection of fundamental building blocks.

to an on-chip voltage booster, it will run on anything between 0.5 and 5.5 V! That's impressive; but better yet, it's not an either/or proposition. Different voltages (internally generated or supplied externally) can be allocated around the chip and pins. So, for example, you could run the PSoC 3 off a single 0.9-V cell, internally boosted to 1.8 V for the core, yet still support a mix of higher voltage analog and digital interfaces.

Like most modern MCUs, the PSoC 3 can generate its own clock (1 to 67 MHz) with a combination of an on-chip oscillator and PLL. The on-chip oscillator is calibrated at the factory and guarantees ±1% across the voltage and temperature (–40° to 85°C) range. Ditto for the 1.024-V on-chip analog voltage reference, which is good for ±0.9% across voltage and temperature.

PSoC 3 variants that include USB 2.0 ports ("Full Speed," i.e., 12 Mbps) can clock the CPU with the USB clock, thereby achieving PPM timing accuracy without the need for an extra crystal. There are also built-in low-speed (1, 33, and 100 kHz) clocks that, although low accuracy, are useful for housekeeping tasks, such as periodic wakeup or running the processor at super-low speed (and super-low power).

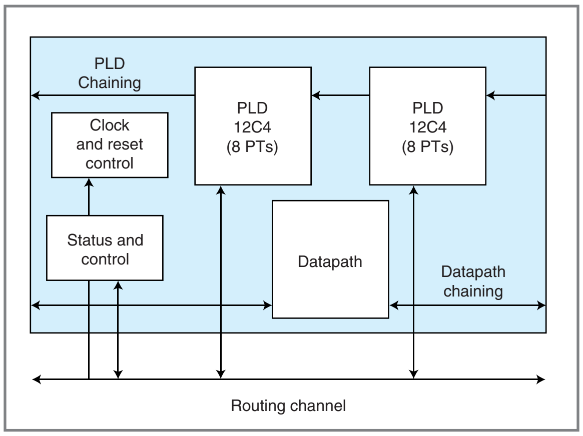


Figure 4—For the first time, Cypress exposes the secrets of the Universal Digital Block and gives users the ability to define their own programmable logic functions.

Many MCUs tout their excellent features on the front page of the datasheets. But look in the fine print and you'll typically find specs (e.g., clock frequency, analog accuracy) that are compromised at temperature and voltage extremes. Cypress makes a point of noting that their new PSoCs—thanks in large part to the step-up voltage booster—work well in all conditions (see Figure 2).

The PSoC 3 roadmap includes parts in a range of packages from 48 to 100 pins. Those who've struggled with juggling I/O will appreciate the fact most PSoC 3 pins (i.e., except those used for special functions like USB or I^{2}C) are freely routable to any internal digital or analog function. The general-purpose I/O lines are quite configurable (e.g., open drain, pull-up, pull-down, slew rate, Schmitt trigger, etc.) and any pin can serve as an interrupt input. The GPIOs also can be called to duty as

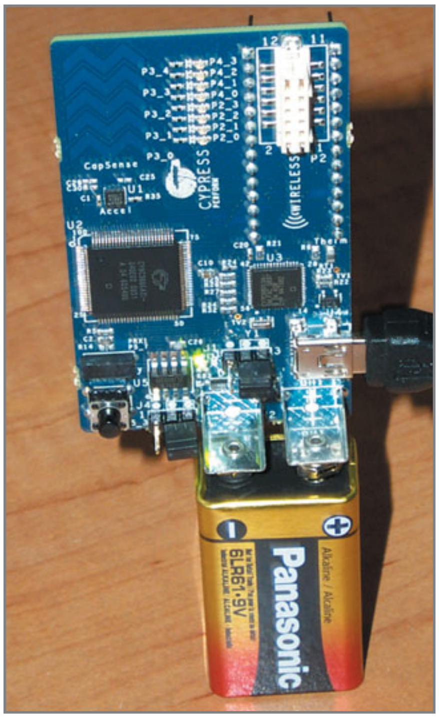
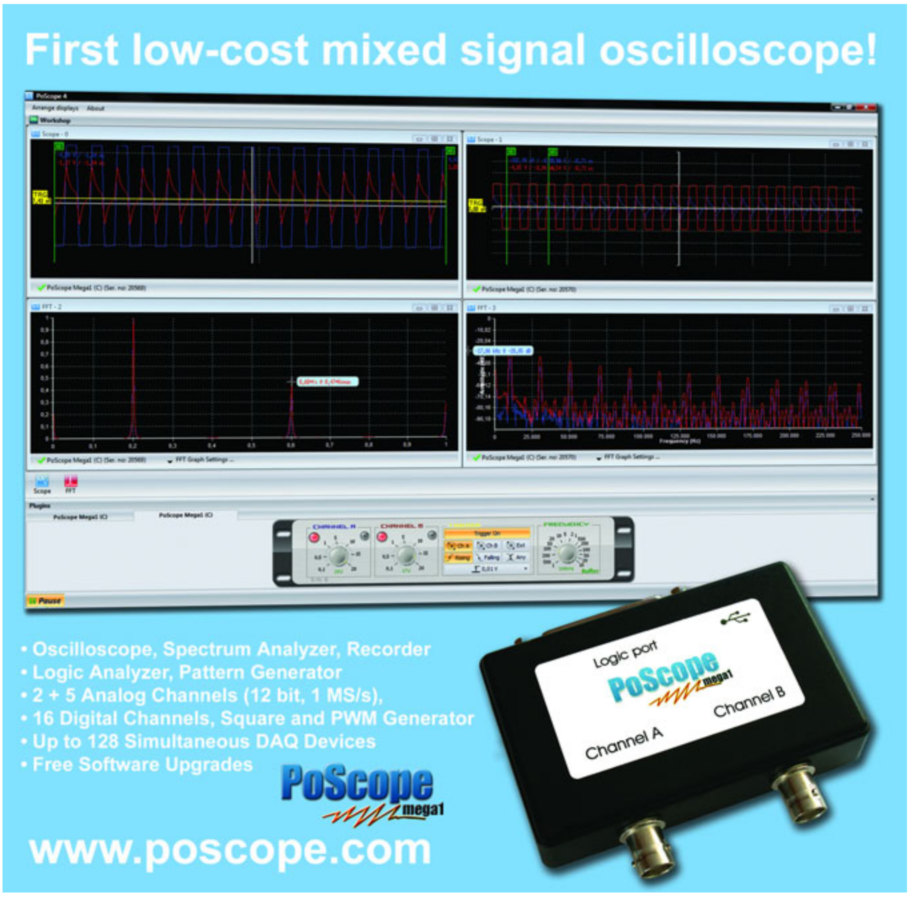


Photo 1—Even if you don't have an immediate application, I'd suggest checking out the PSoC "First Touch" starter kit. Forty-nine dollars is a small price to pay to get up to speed with the truly unique PSoC way of doing things.


First low-cost mixed signal oscilloscope!
• Oscilloscope, Spectrum Analyzer, Recorder
• Logic Analyzer, Pattern Generator
• 2 + 5 Analog Channels (12 bit, 1 MS/s),
• 16 Digital Channels, Square and PWM Generator
• Up to 128 Simultaneous DAQ Devices
• Free Software Upgrades
PoScope mega1
www.poscope.com
Logic port
Channel A
Channel B

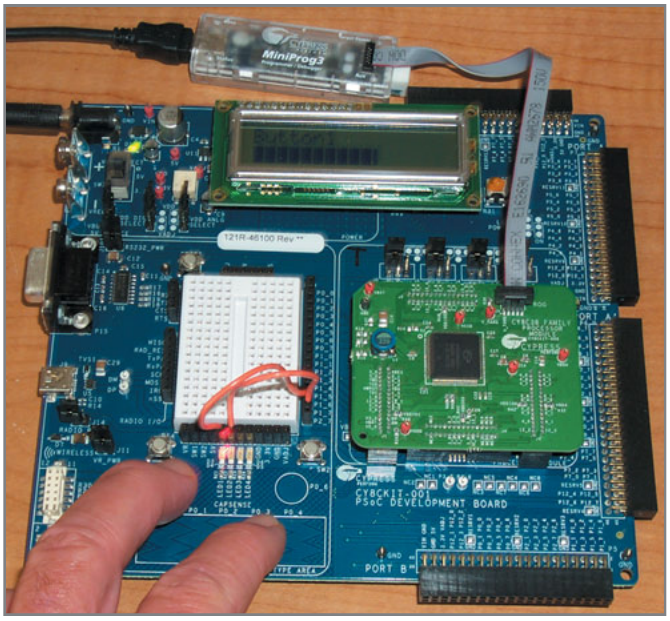


Photo 2—The PSoC development board is a platform that supports any PSoC, past or present. It's suitable for full-fledged development with a MiniProg programmer/debugger, full access to the pins, and the ability to configure various voltage (digital, analog, I/O) options.

LCD segment drivers or CapSense capacitive-touch sense inputs. Here's a nice touch: the entire pin state is stored in on-chip flash memory and initialized automatically at power-on reset so you come to life with the pins in a known state.

I used to laugh when I saw outfits pushing flash memory MCUs with "10-year" data retention specs into embedded apps, many of which have longer lifetimes. I just imagine some hapless designer squirming on the witness stand as the plaintiff's prosecutor grandstands: "You admit you designed this chip in with clear warning it could fail?"

By contrast, the PSoC 3 guarantees 20-year data retention, 100,000 cycles for program flash memory, and 1M cycles for EEPROM. Further reassurance comes with the ability to allocate a portion of the flash memory to error-correction code (ECC) that can fix single-bit errors and detect 2-bit errors (per 8-byte fetch).

A full spectrum of flash security options ranges from password protecting a single flash page up to locking the entire chip irrevocably forever. Talk about attention to detail, even the register that locks the chip uses a "majority voting" scheme for the critical throw-away-the-key bits. Better to be safe than sorry, and a penny's worth of extra silicon is a small price to pay for peace of mind.

PUTTING THE "P" IN PSoC

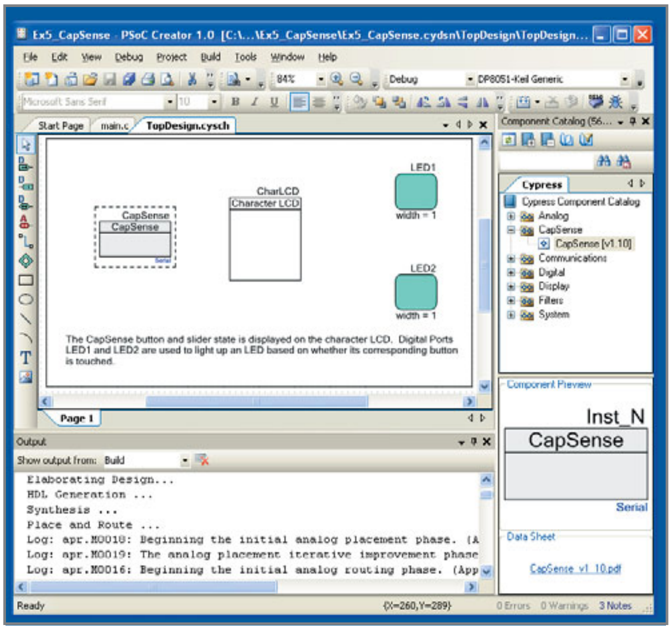
So far with the PSoC 3 we've got a first-class '51 flash memory MCU that's quite competitive with anything on the shelf. But, of course, being competitive with what's already out there isn't that compelling. What sets the PSoC apart from the rest, as it has since day one, is digital and analog hardware programmability.

As with the original PSoC, the PSoC 3 analog subsystem (see Figure 3) starts with a collection of building blocks including: a high-resolution sigma-delta ADC, current/voltage DACs, comparators, op-amps, voltage reference, and so-called "SC/CT" (switched capacitor, continuous time) blocks. The latter, combining an op-amp with capacitor and resistor arrays, is a versatile signal conditioner that can be configured as a PGA, buffer, mixer, modulator, and so on. New additions to the analog capabilities include special support for segment LCD and "CapSense" capacitive touch-sensing applications. For both of these, the ability to freely route pins in order to facilitate connector layout will make life easier for PCB designers.

Higher-end members of the lineup take analog capabilities further with a digital twist in the form of a "Digital Filter Block," essentially a mini-me 24-bit DSP. The DSP is itself quite configurable, with the ability to apportion the overall signal-processing bandwidth in terms of the number of the filter channels, filter complexity, sampling rate, and so on.

While the analog subsystem has better specs and new features, it's similar in principle to the original. By contrast, the digital subsystem is different in obvious, and arguably profound, ways. In the first PSoC, the exact nature of the digital programmability was rather limited and indeed somewhat hidden. The designer could choose some options, but certainly had no ability to get under the hood. Arguably, that was a good thing since making programmable logic easy was a big reason PSoC took the market by storm.

With the new PSoCs, Cypress is popping the hood, at least partway, on the digital subsystem. Now you get to see inside the Universal Digital Block (UDB) that populates the digital logic array. As shown in Figure 4, each



The CapSense button and slider state is displayed on the character LCD. Digital Ports LED1 and LED2 are used to light up an LED based on whether its corresponding button is touched.

```
Elaborating Design...
HDL Generation ...
Synthesis ...
Place and Route ...
Log: apr.M0018: Beginning the initial analog placement phase. (A
Log: apr.M0019: The analog placement iterative improvement phase
Log: apr.M0016: Beginning the initial analog routing phase. (App
```

Photo 3—The easy-to-use "PSoC Creator" software is a one-stop shop for PSoC hardware and software development. On the screen, users can take advantage of the familiar "schematic entry" interface. Those who wish to dig a little deeper can take advantage of a full-fledged HDL synthesis regime.

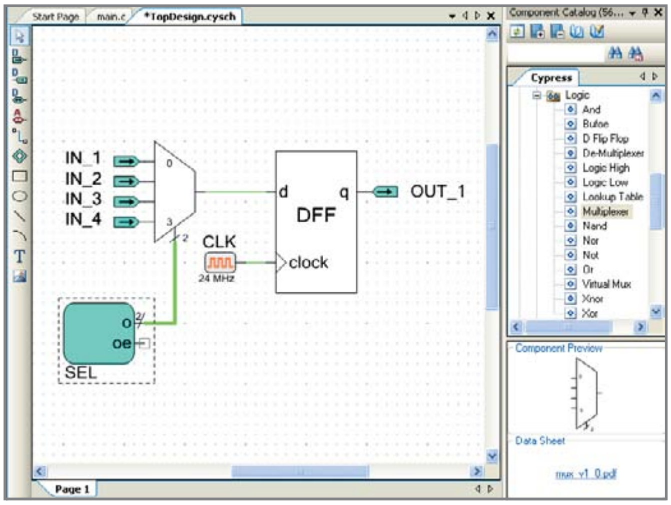


Photo 4—It may be old-school, but the simple schematic entry approach that PSoC Creator uses is something most embedded designers are familiar with.

UDB contains a unique pairing of datapath and product term logic. More than a matter of satisfying your curiosity, Cypress now allows, indeed encourages, you to create your own custom logic, a significant new capability that has implications all the way up the tool chain.

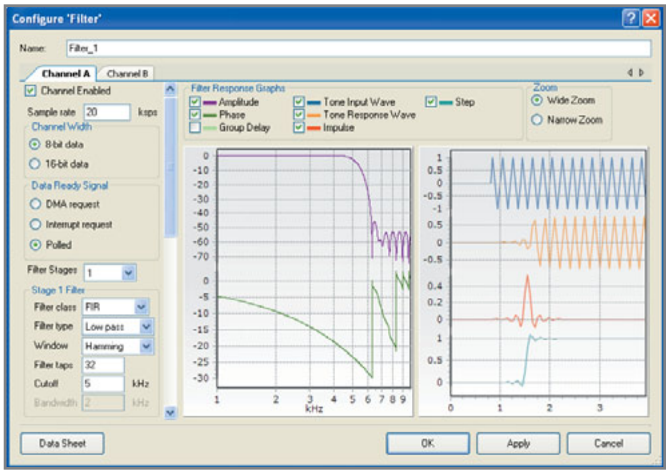
IN THE BEGINNING

Speaking of tools, the best way to see what's new is to give PSoC 3 a test drive. Cypress makes it easy with two distinct hardware options to choose from.

The PSoC 3 FirstTouch starter kit offers an easy way to kick the tires and is practically an impulse buy at just $49 (see Photo 1). Out of the box, it includes demo programs that take advantage of an on-board accelerometer, temperature sensor, the requisite blinking lights (i.e., bank of eight LEDs), and a small CapSense touchpad. There's also a connector to add a Cypress Semiconductor CyFi 2.4-GHz radio module if you're so inclined.

Cute demos aside, the starter kit is more than a toy. Headers on the back of the board provide access to 24 of the PSoC 3's I/O lines, all the more useful in light of the chip's ability to map practically any internal digital or analog function to them. Better yet, a USB interface provides dedicated access to the PSoC 3 on-chip debug logic, supporting full-featured download, debugging, and flash programming without the need for an external emulator pod.

As you can see in Photo 2, the PSoC development board is a higher-end option, although still reasonably priced at $249. The key difference from the starter kit is that the development board provides comprehensive support for any PSoC (i.e., the original PSoC and the new PSoC 3 and PSoC 5) by utilizing a plug-in module


Configure 'Filter'

Name: Filter_1

Channel A | Channel B

[✓] Channel Enabled
Sample rate 20 ksps
Channel Width: [✓ 8-bit data] 16-bit data
Data Ready Signal: DMA request, Interrupt request, [✓ Polled]
Filter Stages 1
Stage 1 Filter
Filter class FIR
Filter type Low pass
Window Hamming
Filter taps 32
Cutoff 5 kHz
Bandwidth 2 kHz

Filter Response Graphs: [✓] Amplitude, [✓] Phase, [ ] Group Delay, [✓] Tone Input Wave, [✓] Tone Response Wave, [✓] Impulse, [✓] Step
Zoom: [✓ Wide Zoom] Narrow Zoom

Photo 5—Various "Wizards" make it easy to play "what if" while exploring built-in PSoC capabilities such as the Digital Filter Block.

Project Boxes
Are In Stock
for Immediate Delivery
in Any Quantity
at DesignNotes!

DesignNotes.com
What Your Electronic Hobby Stores Used To Be
Celebrating Our 10th Year Anniversary
1-800-957-6867 www.DesignNotes.com

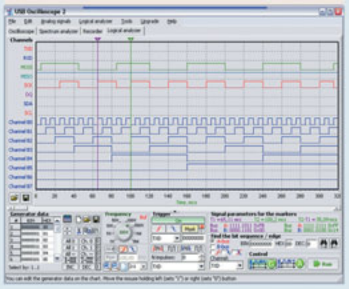
PIC-SERVO
MOTION CONTROL

MOTION CONTROLLERS FOR
BRUSH, BRUSHLESS AND
STEPPER MOTORS.

- controller chips
- controller boards

www.picservo.com
JEFFREY KERR, LLC

USB Oscilloscope for $169.50
Logic and Spectrum Analyzers, Generator.
www.HobbyLab.us



for the MCU. It also comes with a MiniProg3 program/debug adapter that works with any PSoC old or new. Other notable enhancements include an LCD, breadboard area, extensive pin access via headers, and the ability to exercise the full range of voltage options (i.e., digital, analog, I/O).

All well and good, but not essentially different from existing Cypress boards (or competitors' for that matter). What really sets the new chips apart is the "PSoC Creator" tool suite (see Photo 3). Compared to the earlier "PSoC Designer," an obvious difference is the need to accommodate software development for the new '51 and ARM cores. To that end, Creator comes with compilers for both the PSoC 3 (Keil) and PSoC 5 (GNU, courtesy of CodeSourcery). To fully exploit the advantage of these "open" cores, Creator uses a plug-in approach that can host other popular compilers (e.g., ARM RealView for the PSoC 5).

As I alluded to earlier, by far the most significant difference with Creator derives from the new freedom users are given to craft their own programmable logic add-ons. The difference may not seem significant, but in fact calls for major changes tool-wise. To that end, Creator enhances the visual interface with a schematic entry look and feel (see Photo 4).

In fact, dig deep enough under the hood and Creator reveals more EDA-in-drag aspects, namely a full-fledged Verilog synthesis regime. OK, it's there for those who need it (and those who think they need it but really don't). But it's important to understand that most users need never grapple with a single line of Verilog. As with PSoC Designer before, PSoC Creator supports design at a very high level with an extensive library of proven functions built in. Each comes with a "datasheet" that describes its use, performance, AC and DC specs, and so on, just like a real chip.

With PSoC Creator, the arrows in your design quiver go beyond the usual suspects with Wizards to manage the hard logic too. For example, the "Digital Filter Block" (i.e., 24-bit DSP) mentioned earlier is fronted with a very cool filter Wizard that allows you to play "what if" with key parameters (filter type, complexity, sample rate) and see the filter response right on the screen (see Photo 5). Other Wizards help configure MCU built-ins, such as the interrupt controller, DMA controller, peripherals (e.g., I^{2}C, USB, CAN) and system functions. All the Wizards automatically generate driver code and an API for easy access by your application program.

ONCE MORE, WITH FEELING

The new PSoC 3 and PSoC 5 are going to be a big hit, for exactly the same reasons the original PSoC was: digital and analog programmable logic, low-cost ($2 to $10 across the PSoC 3 and PSoC 5 lineups), and a very powerful yet easy-to-use high-level design tool. The popular '51 and ARM cores are really tasty icing on the cake, boosting the range of possible applications by an order of magnitude. And the new ability to define custom programmable logic enables creation, sharing, and perhaps even a market for value-added IP.

Why doesn't every MCU supplier offer something like the PSoC? Maybe the answer is because it's hard. The closest thing you'll find to the PSoC concept is in the FPGA soft-core space. The difference, of course, is that the PSoC hard cores offer far superior price, performance, and power consumption compared to any soft-core.

There was nothing like the original PSoC when I first covered it all those years ago in "SoC Hop." And there's nothing like PSoC 3 and PSoC 5 today. It was totally unique then, and still is. If you missed hitching a ride on the PSoC the bandwagon the first time, now's your chance. Hop on! ■

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

SOURCE

PSoC 3 and PSoC 5
Cypress Semiconductor Corp. |
www.cypress.com

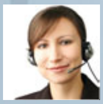
The New High-Performance Catalog

Industry's only full-featured online catalog

•Browse •Search **•Check Stock** •Buy

Try It Now at www.mouser.com

WARNING: Designing with Hot, New Products May Cause A Time-to-Market Advantage.



Introducing the new, enhanced, high-tech online catalog. Allowing you to browse, search, check stock, buy, and more — An industry rst And only online catalog to offer all these features without ever having to leave the catalog!.

MOUSER® ELECTRONICS

a tti company

www.mouser.com (800) 346-6873

by Ralph Stirling

The CtrlBox

An Ethernet Control System Interface

This well-designed motion control interface features a WIZnet W5100 that provides an interface between a Xilinx Spartan3 FPGA board and a host computer. The FPGA counts pulses from four quadrature shaft encoders and generates PWM for a motor amplifier. You can use the host PC to perform control calculations. Here the system is used in an inverted pendulum apparatus.

It seems as though I've spent much of my professional career redesigning data acquisition and control systems for new computer interfaces. I've used parallel, serial, SCSI, Multibus, ISA 8-bit, ISA 16-bit, and PC/104. I've contemplated EISA, VME, Sun Sbus, Apple NuBus, PCI, FireWire, and USB 1.1-2.0.

My customers for the last 20 years have been professors and students who work with digital control systems. For example, a few years ago, I built the "DataBox," which was an ambitious custom data acquisition system designed for use in a lab. I call my newest design the "CtrlBox," which is a feedback and control system interface for an inverted pendulum (see Photo 1 and Figure 1). In this article, I'll describe the design. This project can give you a base for designing your own real-time control interface.

INVERTED PENDULUM

The inverted pendulum is a popular apparatus that many academics—such as the instructors at Walla Walla University where I work—use for control systems education. A broomstick or pencil balanced on your hand is a human-controlled inverted pendulum. Segway scooters and rockets are digitally controlled inverted pendulums. The inverted pendulum is a classic example of an unstable system.

In our laboratory apparatus, the angle of the freely swinging

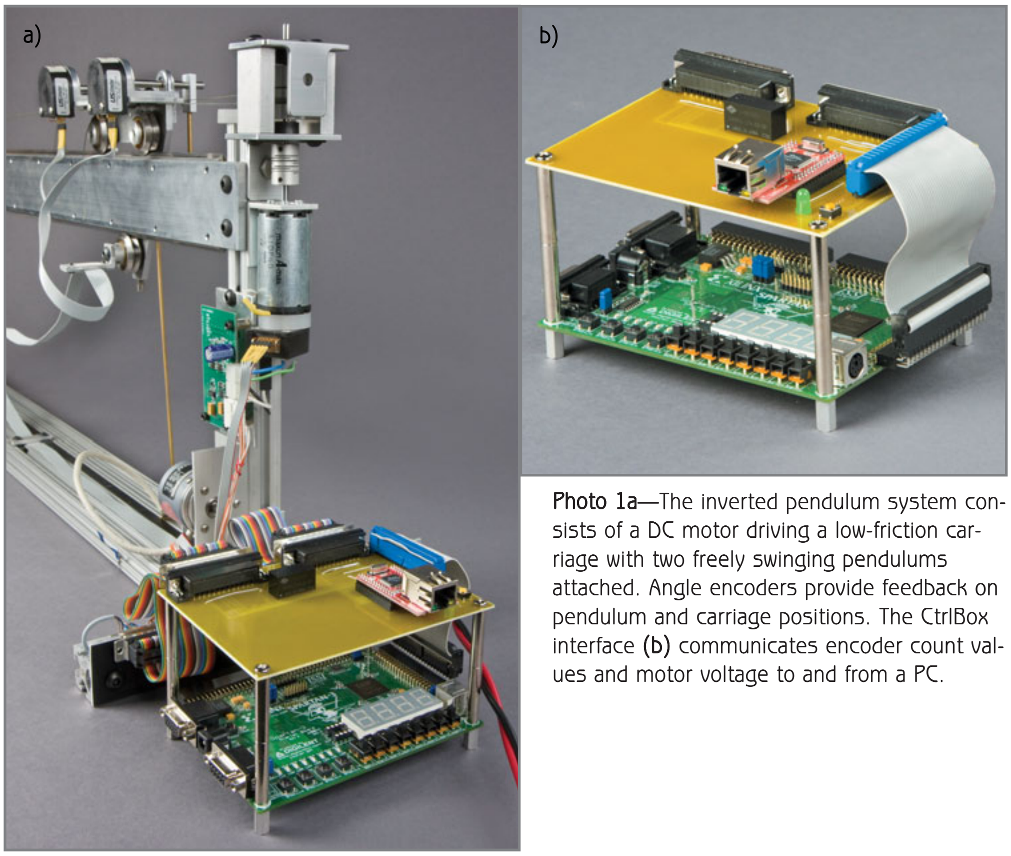
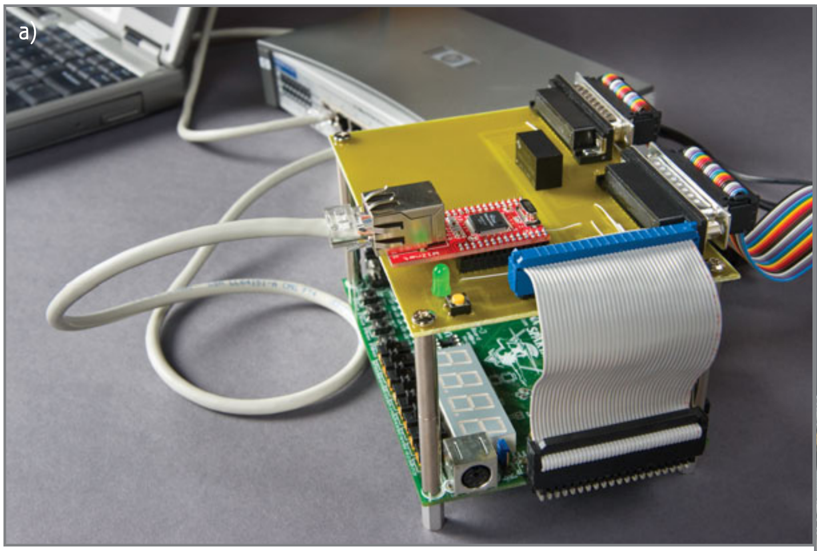


Photo 1a—The inverted pendulum system consists of a DC motor driving a low-friction carriage with two freely swinging pendulums attached. Angle encoders provide feedback on pendulum and carriage positions. The CtrlBox interface (b) communicates encoder count values and motor voltage to and from a PC.



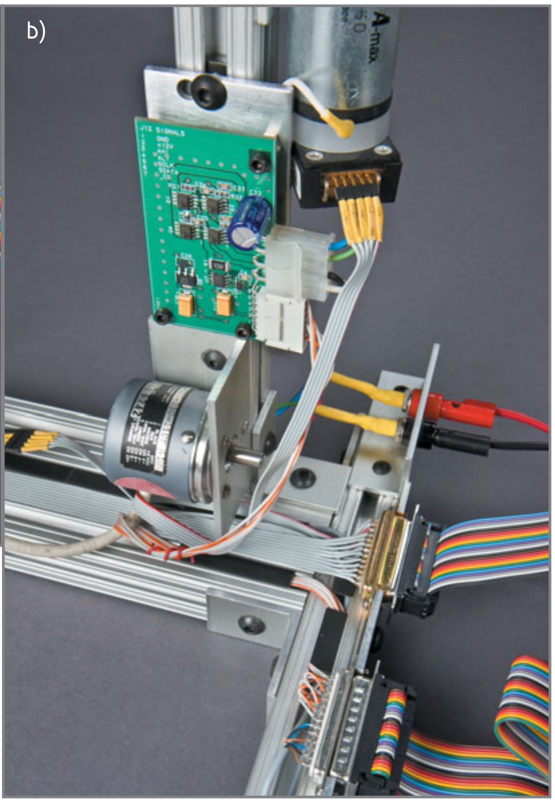


Photo 2a—The CtrlBox communicates with the control PC using 100-Mbps Ethernet through a switch or router. It obtains an IP address by DHCP. b—A custom H-bridge motor amplifier drives the DC motor in our system. Any motor amplifier that converts a PWM duty cycle to motor voltage would also work.

pendulum and the position of the carriage that moves its base must be measured at a rate of at least 100 Hz, and the new motor voltage must be computed quickly, in order to keep the pendulum balancing. The motor drives the carriage through a low-friction pulley and bearing system.

There are single, twin, and stacked inverted pendulums. The apparatus shown in this article is a twin inverted pendulum, which is equivalent to balancing two pencils (of different lengths) on your hand at the same time. The two rods are constrained to rotate in only one axis. The stacked pendulum has one rod stacked on top of another, with one encoder to measure the relative angle between them and another encoder to measure the angle between the bottom rod and the carriage.

DIGITAL CONTROL

There are endless numbers of commercial digital control systems, but none met my specific requirements: control algorithms written and tested in the well-known MATLAB environment on a familiar operating system (OS); a minimum of four quadrature encoder inputs and one high-frequency pulse-width modulation (PWM) output; a minimum 100-Hz sample rate with minimal delay between reception and transmission (i.e., send a short packet every 10 ms and receive a short packet immediately thereafter); guaranteed no sample loss; a simple PC configuration for use as a controller (no complicated drivers); and low enough cost to make multiple systems affordable.

My attempts at using SBCs with a real-time OS did not satisfy my users. Tricky Windows real-time extensions made for a complicated system set-up process and MATLAB compatibility issues. The requirement for four encoder inputs precluded simple USB data acquisition modules.

In 1986 I implemented an Ethernet interface on a system for an employer. Design time and budget limitations precluded a full-custom TCP/IP stack implementation, so I purchased Multibus cards from Communications Machinery Corp. for $3,000 each. The boards provided full TCP/IP communications, with a simple register and ring-buffer interface to my system. Ever since, I've thought about the number of communications problems that are solved with TCP/IP over Ethernet. Ethernet has a high signaling rate and TCP/IP provides guaranteed delivery. Every OS and computer produced today supports Ethernet and TCP/IP. If used with a network switch to isolate the control system and controlling PC from other network traffic,

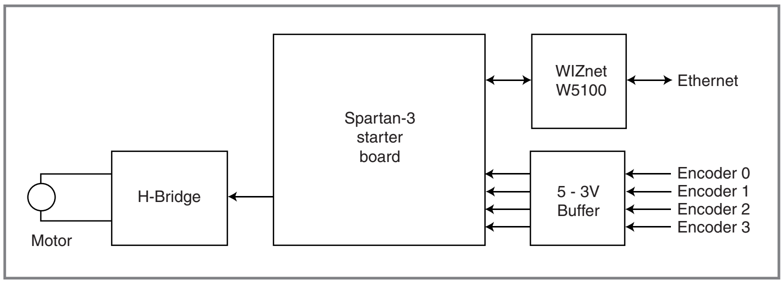


Figure 1—The Digilent Spartan-3 starter board has three 40-pin I/O connectors. The custom CtrlBox board connects to one of these headers with a ribbon cable. This board contains only connectors, the WIZnet module, and a 5- to 3.3-V level converter for the angle encoders.

communication delays are low and relatively predictable.

So, in an effort to find the perfect interface, I investigated nearly every Ethernet TCP/IP device introduced during the last 15 years or so. I tried low-cost modules like uCsimm, SitePlayer, Lantronix Xport, DigiConnect, Dallas TINI, and Brightstar ipEngine, as well as expensive products like UEI PowerDaq. None of them gave me what I wanted. Most of the products could not deliver the sample rate and reception/transmission response time I needed.

Finally, about two years ago, WIZnet introduced the first TCP/IP hardware solution, the W3100. The chip promised me the functionality of the 80-square-inch $3,000 CMC network card in a 2-square-inch $20 module—with better performance! After some initial experimenting with the W3100 and its successor, the W3150, I decided to use a WIZnet module with a Xilinx Spartan3 field-programmable gate array (FPGA) as my control systems interface (see Photo 1). The current implementation pairs a Digilent Spartan-3 starter board with a WIZnet WIZ810MZ module, level converters, and connectors to interface an ordinary PC running Windows and MATLAB to my inverted pendulum apparatus (see Photo 2). The Digilent board contains a 200,000-gate Spartan-3 FPGA, a configuration PROM for loading the FPGA with logic at power-up, SRAM, buttons, switches, LEDs, and connectors.

FPGA-BASED SYSTEM

The distinction between hardware and software is rather blurred in an FPGA-based system like mine. The sample rate counter, encoder counters, and PWM generator are all built as custom logic in the FPGA.

I used a Xilinx PicoBlaze processor softcore to implement the state machine for reading the encoders, updating the PWM, and communicating via the WIZnet interface. A static RAM device on the Spartan-3 board stores transmit and receive packets. The PicoBlaze processor is programmed with a clean, simple assembly language. About 1,000 of 1,024 (maximum) words of instruction memory are used (see Listing 1).

The complete system uses 28% of the slices on the 200,000-gate Spartan-3 FPGA. The logic is implemented in the VHDL and synthesized with the free Xilinx 10.1 Webpack tools (see Figure 2).

INITIAL DESIGN DECISIONS

One of the first design decisions I had to make was which WIZnet interface mode to use (see Figure 3). The W5100 has three interface modes: Direct, Indirect, and SPI serial. Direct mode uses an unacceptable 26 pins. SPI mode uses only three pins (a scarce commodity on the FPGA board). Unfortunately, it requires 32 clock cycles at 70 ns each for a basic register read or write operation. The master clock for the Spartan-3 starter board is 50 MHz, and the PicoBlaze processor uses two clocks per instruction for a cycle time of 40 ns. Each INPUT and OUTPUT instruction takes the same 40 ns. This means 2.56 µs would be required for each WIZnet

Easy Embedded Linux

OMNI - EP

- 200 Mhz ARM9 CPU
- 10/100 Mb Ethernet
- 32 MB RAM
- 16 MB Flash
- 16 Digital I/O Lines
- 2 Ports of USB 2.0
- SPI Bus
- AC97 Amplified Audio
- Battery Backed Clock
- 2 Serial Ports
- Low Power Consumption
- RoHS Compliant

Our newest ARM9 Linux controller, the OmniEP doesn't cost an arm and a leg. It delivers removable storage, amplified audio, ethernet and serial RS232 communication ports in a rugged and attractive enclosure. Models without enclosure and LCD available.

The OmniEP comes preloaded with Linux to jumpstart your development process, with LCD and pushbutton drivers supplied. Large capacity USB drives can be easily mounted in the USB port.

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

JK microsystems, Inc.

International Orders Welcome

MADE IN U.S.A

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

VISA MasterCard PayPal IPC MEMBER ASSOCIATION CONNECTING ELECTRONICS INDUSTRIES BBB

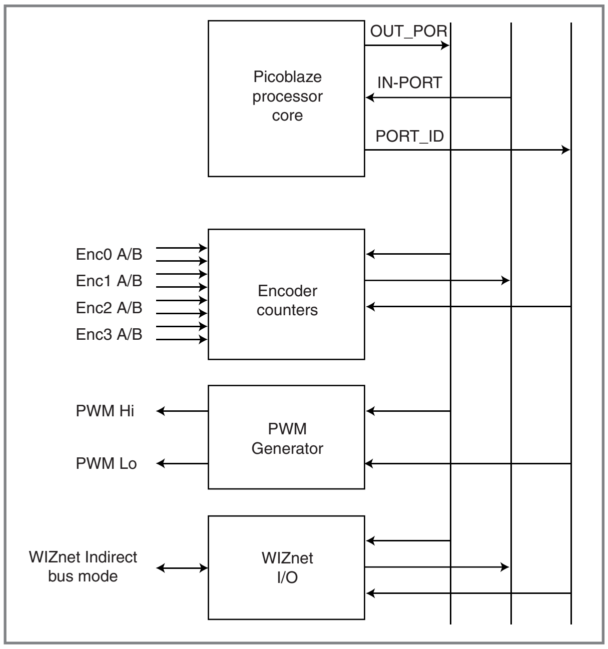


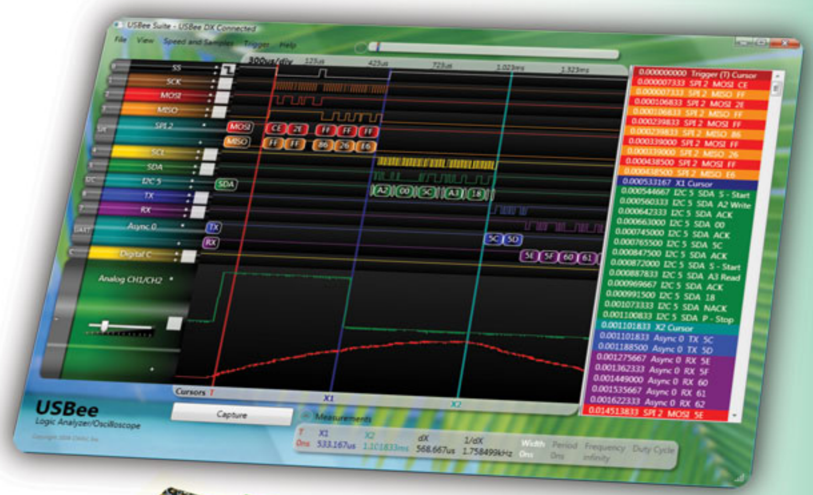
Figure 2—The Spartan-3 FPGA logic consists of the PicoBlaze softcore connected to three main modules: the 32-bit encoder counters, a 16-bit PWM generator, and the WIZnet interface. Other modules not shown include the sample clock counter, configuration PROM reader, debugging UART, and LED and button connections.

register read or write operation, seriously limiting system performance. Indirect mode is a compromise between the other two modes. It requires eight data lines, two address lines, and three control lines, 10 pins more than the SPI mode. Each WIZnet register or memory read or write operation takes two 70-ns clock cycles for the first read or write, and just one for each additional sequential address. The control signal timing for Indirect mode was implemented as a part hardware/part software state machine that actually takes eight instructions per read or write, or 320 ns per cycle. This level of overhead has proved to be acceptable for my application.

The encoders used in my inverted pendulum apparatus have a resolution of 4,096 counts per revolution. Although the pendulums never need to count beyond one revolution clockwise or counterclockwise, the carriage position needs to be tracked from one end to the other. A Maxon DC motor drives the carriage through a pulley and taut fishing line. An encoder attached to the motor shaft then gives 55.33 counts per millimeter of carriage travel. The maximum length of travel is 0.44 m from end to end, or 22,585 counts. All the counts would fit in 16-bit integers, but I used 32-bit integers anyway. Other projects required 32-bit counts. Matlab was no faster at processing 16-bit integers than 32-bit integers. Overhead for an Ethernet TCP/IP packet negated any performance improvements for going from 32-byte packets to 16-byte packets, and there was plenty of room in the FPGA for 32-bit counters. Each encoder counter consists of a glitch-removing filter for the two encoder inputs, and a simple state machine to increment or decrement the 32-bit counter whenever the encoder inputs change values. The state machine is implemented as a look-up table.

PWM for the motor drive is implemented as a free-running counter with a comparator to switch the output off at the desired duty cycle value. This would make it easy to add additional PWM outputs without duplicating the counter. The number of bits in the free-running counter determines the resolution and frequency of the PWM signal.

The sample clock that paces the system is a 32-bit counter. The sample period loaded into this counter is transmitted along with the PWM duty cycle and PWM Enable flag from the host computer as four 32-bit integers.



See It & Solve It
with
USBee Test Pods

- Logic Analyzers
- Oscilloscopes
- Signal Generators
- Protocol Analyzers
  - I2C, SPI, ASYNC, CAN
  - 1-Wire, PS/2, USB
  - I2S, SMBus, Serial

- Configurable
- Programmable
- High Speed USB
- PC-Based
- And start at $139

USBee™
USB based Electrical Engineer

www.USBee.com

The first integer value in the packet is the PWM value. The second is blank (for future use). The third is a one or zero to indicate whether the motor drive should be enabled or disabled. The fourth is the sample period. A value of zero in the sample period indicates the sample rate should be left untouched. It seems wasteful to use 32-bit integers where only 1 or 16 bits are needed, but the additional transmit time is insignificant, and the reduction in complexity by using the same size parameters is nice.

The inverted pendulum has two limit switches to detect carriage end of travel. These are industrial Hall effect sensors, whose outputs are converted to logic-level values with optocouplers. The limit switches disable the PWM drive regardless of the state of the host control program. Because inexperienced students are writing the control programs, I don't want to depend on either their understanding of safety code or the reliability of the control link. To restart after an out-of-range limit event, the carriage is pushed back to the center and the rods are lifted back to vertical.

IMPLEMENTING DHCP

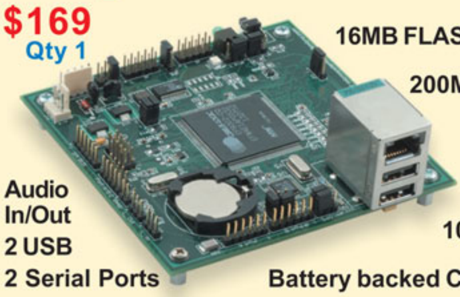
The initial implementation of this system used hard-coded Internet protocol (IP) addresses. This was sufficient for a standalone PC and the CtrlBox system (or even one connected to a simple network). But due to the ever-growing complexity of our campus network, the CtrlBox needs to support dynamic IP address assignment. Most networks use a procedure called the dynamic host configuration protocol (DHCP) to issue IP addresses to clients. With a WIZnet-provided sample DHCP client implementation in C—which was targeted for an Atmel processor—and some descriptions of the DHCP protocol—which I found on the 'Net—I created a minimal DHCP process for the CtrlBox.

DHCP consists of four basic phases. In the first phase, the client broadcasts a DHCP discovery packet of about 300 bytes, using the user datagram protocol (UDP). One DHCP server responds with a DHCP offer (more than one can potentially respond). The client extracts the IP address, Subnet, and Gateway fields out of the offer packet, slightly modifies the discovery packet, and resends it as a DHCP request. The server acknowledges the formal request with a DHCP Acknowledge. More complete DHCP implementations would keep track of the time limit specified by the server and request a renewal of the IP address at the appropriate time, but I did not include this feature. My network always assigns the same address to a device whenever it appears on the same subnet, so renewal isn't necessary. The CtrlBox normally remains on for the duration of a lab period, and the DHCP renewal will certainly be longer than that. I could add automatic renewal, but at a cost in complexity, instruction memory space, and FPGA gates.

To implement the DHCP protocol, I created a template for the DHCP discovery packet and stored it in the configuration PROM after the FPGA configuration bitstream. When the FPGA finishes

Easy Embedded Linux

$169
Qty 1



16MB FLASH / 32MB RAM
200Mhz Arm9 CPU
16 Digital I/O
Watchdog
10/100 Ethernet
Battery backed Clock/Calendar
Audio In/Out
2 USB
2 Serial Ports

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

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

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

JK microsystems

DESIGNCON® 2010
The **BRIGHTEST** Minds in Electronic Design

Conference: February 1–4, 2010
Exhibition: February 2–3, 2010
Santa Clara Convention Center
Santa Clara, California

If there is one industry event you attend this year, make it the definitive event for electronic design experts.
Make it DesignCon.

www.designcon.com/2010

Listing 1—The PicoBlaze softcore in the FPGA is programmed to set up the WIZnet module and process encoder counts and PWM values. This C-like pseudocode documents the main loop and the receive routine (as an example of a lower level routine). The actual implementation is in PicoBlaze assembly language.

```
// Top level routine
main()
{
   pwmdisable();          // disable pwm right off

   promreset();           // initialize from config prom
   promcopy();            // copy config prom data to sram

   wizrst();              // reset Wiznet W5100
   wizinit();
   wizsocket();
   wizudpsetup();         // set up for UDP
   wizdhcpdisc();         // DHCP Discovery phase
   wizdhcpoffer();        // wait for DHCP Offer
   wizdhcpreq();          // DHCP Request phase
   wizdhcpack();          // wait for DHCP Acknowledgement

   wiztcp();              // setup for TCP server

   while (1)
   {
       // if packet has been received, process

       if (dataready())   // dataready implies connected
       {
           wizreceive(); // copy packet out of W5100
           pwmcopy();    // copy pwm data to pwm registers
           setsmplclk(); // set sample clock if value changed
       }

       // if sample period has completed, transmit encoder values

       if (smplclk() && connected())
       {
           filltxbuff(); // copy encoder data to buffer
           wiztransmit();// copy buffer to W5100
       }

       // check TCP status, go back to listening if disconnected

       wizstat();
   }
}

// wizreceive
//
// Setup:  Rcounter is receive byte count constant (max 32 bytes)
//     Mrxbuff is scratch pad receive buffer

wizreceive()
{
   Rcounter = 0;
   while (Rcounter < Rcnt)    // block until we have enough bytes
       Rcounter = S_RX_RSR;
   Rcounter = Rcnt;
   Rp = S_RX_RR;
   while (Rcounter > 0)
   {
       Mrxbuff(i++) = *(0x6000+sockbase+(Rp & 0x07FF));
       Rp++;
       Rcounter--;
   }
   S_RX_RR = Rp;
   S_CR = RECV;
}
```

the configuration process, my PicoBlaze program searches through the PROM for a unique "sync" word and then copies the bytes that follow to the SRAM. The DHCP routines then manipulate and transmit the template.

Note that the W5100 does not come with a preassigned MAC address. There are three possible ways of coming up with a valid address. The official method is to buy a block of addresses from the IEEE. If you are going to make a large number of devices, you can purchase an organizationally unique identifier (OUI). This will cost you $1,650 and give you a block of 12 million MAC addresses. If that is more than you need, you can purchase an individual address block (IAB) of 4,096 addresses for $550. If you can't justify those prices, you have two other choices.

The second method is to collect old network adapter cards and copy their MAC addresses. Discard the cards, and you'll have globally unique addresses to use. The drawback to this method is that your device will be treated as 3Com, Intel, Dell, or another vendor's hardware based on the OUI you borrowed.

The last method, which I chose, is to use locally administered addresses (LAA). If the next-to-least significant bit of the most significant byte of the MAC address is a one, the address is an LAA. The least significant bit of the most significant byte must be zero for standard unicasting hardware (like mine). You get to choose all the other bits. There is absolutely no guarantee that the address you pick out of the blue won't collide with one someone else picks, so you must not use an LAA on the open Internet. Within the confines of our campus—where I am the only one building custom Ethernet hardware—it isn't hard to pick an address block I feel safe with.

CONTROL SYSTEM OPERATION

Once the CtrlBox obtains an IP address through the DHCP mechanism, it configures itself as a TCP server and waits for the host PC to connect. A MATLAB control program on a Windows PC connects and sends the desired sample rate, using the "pnet" function from the MATLAB central user program exchange site. There may be different methods of performing TCP socket operations on other operating systems. The CtrlBox begins sending encoder count values at the specified sample rate. Matlab computes new motor voltage values (PWM duty cycle) and transmits them back after each received encoder count packet. A variety of control algorithms could be used with the CtrlBox, but I needed to be sure it would work with the specific type of algorithms used in our feedback and control classes.

The type of control algorithm we use for the inverted pendulum problem is a model-based, state-space method. The inverted pendulum apparatus was carefully weighed and measured, and a model of its behavior was created. The particular algorithm used in our Digital Control class is called a "Reduced Order Observer." Explaining state-space control theory is beyond the scope of this article. State-space control differs from the more familiar PID control theory by its reliance on modeling the system rather than tuning the controller. The reduced order observer control loop consists of seven matrix multiplies and five matrix additions. Part of this math is to calculate the velocity of the carriage and pendulums from successive position samples (encoder counts).

One of my concerns when I started this project was latency due to the overhead of the TCP and the host operating system. The control algorithm expects to be given the current pendulum angle and carriage position, not from some time in the past. Even more important is to never lose a sample completely. With some older feedback and control interfaces, the operating system could do its own thing long enough to miss a sample. When this happened, the control algorithm would think the carriage and pendulum moved twice as far during one sample period. This caused a rather violent reaction to bring the velocity back into proper range.

The CtrlBox does not have this problem because the TCP guarantees the delivery of every packet. The operating system may be a bit slow in reading the TCP socket buffer on occasion, but all the samples are present, so all the calculated velocities should be correct. When the operating system completes a task that slows the response time, the control program quickly "catches up" with the buffered data.

For this introductory control systems course, the control parameters are derived with the assumption that a continuous-time system is used rather than a discrete-sampled system. For this approximation to be accurate, the encoder-read/compute/PWM-update time must be a small fraction of the total sample time. It is difficult to measure the individual components of this latency. I measured the time from the transmission of an encoder data packet to the reception of a PWM packet using a Tektronix TLA5201 logic analyzer. Over the course of 6,000 samples at a sample rate of 100 Hz, the delay from the transmission of encoder data to the reception of PWM data varied from 0.2 to 0.6 ms. I used a Dell D600 laptop with a 1.8-GHz Pentium M processor running Windows XP as the host to run the MATLAB control

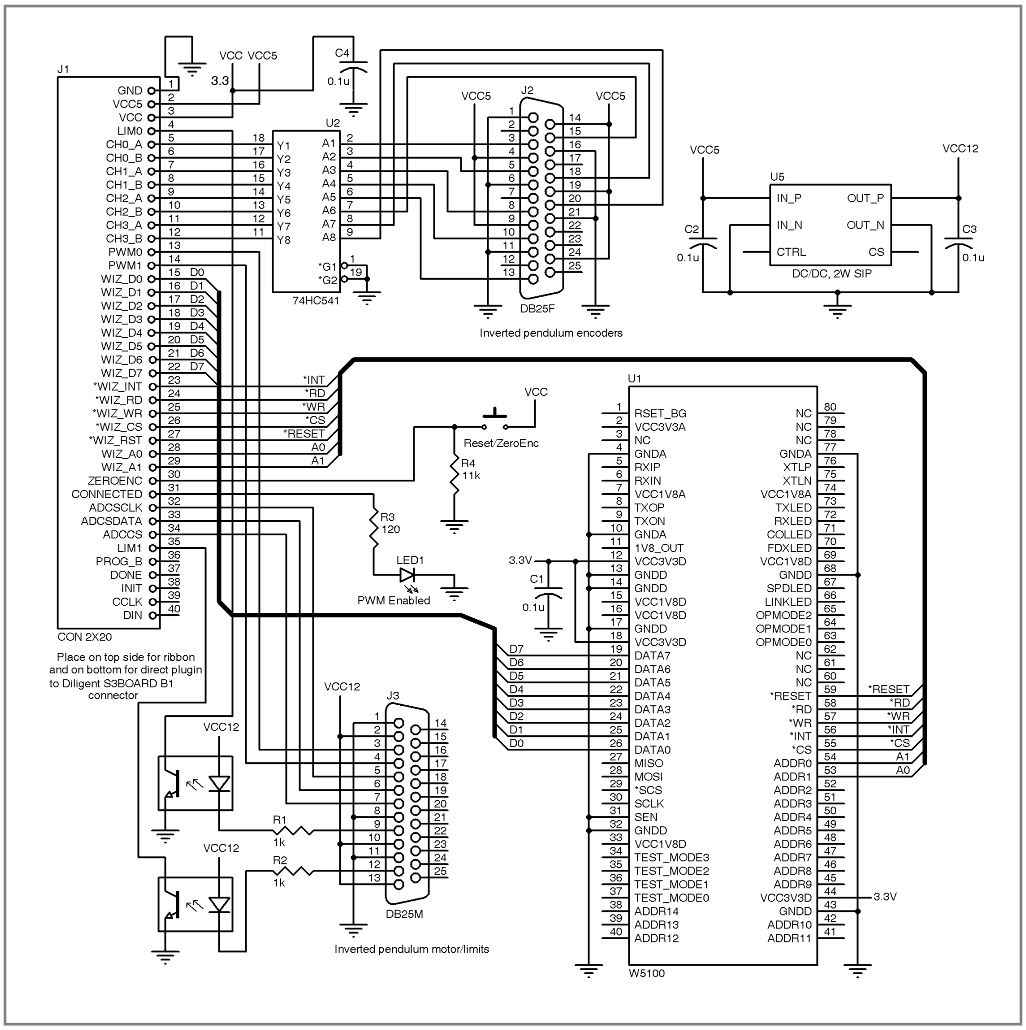


Figure 3—The custom CtrlBox PCB contains only a few components. A 74HC541 buffer converts the 5-V encoder output levels to the 3.3-V levels compatible with the FPGA. A DC-DC convertor supplies the 12 V necessary for circuitry on the motor amplifier and the Hall-effect limit switches. The WIZnet module consumes the majority of the I/O pins.

algorithm. The PC and the CtrlBox were connected with a Hewlett-Packard eight-port, 100-Mbps switch. Removing the matrix calculations does not reduce the response time, so the operating system overhead is likely the primary component of this delay. At a 100-Hz sample rate, a delay time of 0.6 ms is an acceptable 6% of the sample period.

NOISE SUPPRESSION

My original CtrlBox interface had an Apex SA56 single-chip motor driver device on the same board with the WIZnet module. I had problems with motor brush noise affecting the system's behavior. It caused erroneous fault conditions, reset the FPGA, and sometimes corrupted the logic. I treated the problem with shielding, ferrite beads, and digital isolators, with limited success, and then I moved the motor amplifier off-board.

For the present version, I have a new motor amplifier mounted right next to the motor. The 24-VDC power for the motor amp never comes to the CtrlBox. The new motor amp uses discrete MOSFET transistors. It has better specs, it's more robust, and it costs a bit less than the integrated SA56 module. Because I wanted to move the motor amp off-board anyway, the small amount of additional real estate required for the discrete amplifier didn't matter.

Although the off-board amplifier reduces noise issues, it turns out that the FPGA board itself is a rather significant noise radiator. Without driving a motor, the CtrlBox can wipe out 2-m amateur radio communications in an adjacent office. My liberal use of ferrite bead noise-suppression products from Ferroxcube significantly reduces the RFI. Installing the entire controller in a shielded box would be helpful, but I want to keep it exposed so students can see the different parts. Plus, it's useful to keep the buttons and LEDs on the Digilent FPGA board accessible.

ALTERNATIVE APPROACHES

The project is finished. Was it worth it to go with an all-FPGA approach?

It was nice to create my own peripherals that worked exactly the way I wanted (with no extra clutter or complexity), but I found it tedious to work with the PicoBlaze assembly language and long compile/download cycles. I used nearly all of the PicoBlaze CPU's instruction memory space. I thought about partitioning the design into multiple PicoBlaze modules, but it wasn't necessary in the end.

I could have used several other approaches to design the CtrlBox. A more powerful processor softcore (e.g., the Xilinx MicroBlaze or OpenCores ZPU) could have been used in the FPGA, which would have provided C programming tools and much more instruction memory space. I could have used a Spartan-3 starter board for all the encoder counting and PWM generation, along with something like a simple 8051 processor between the FPGA and the WIZnet module. Doing so would have made it easier to write and debug the communications code. I even could have used an ordinary processor (e.g., a Silicon Laboratories 8051 derivative) with the WIZnet module and four LSI Computer Systems LS7366 encoder counters.

Another alternative would have been to use two Microchip Technology dsPIC33F32MC302 processors, which have two on-board encoder counters. One dsPIC would run the WIZnet communications and PWM generation, as well as read two encoders. The other would simply handle the two remaining encoders.

In the end, I went with the all-FPGA approach. When it comes to academic design projects, I usually choose a new technology that I want to learn about and put into the hands of students. Learning to integrate an FPGA softcore and the WIZnet module was a valuable experience. ■

Ralph Stirling (ralph.stirling@wallawalla.edu) has degrees in Electrical Engineering from Walla Walla University and Purdue University. He has been designing instrumentation and motion-control applications for a couple of decades, while helping students with projects and faculty with lab development at Walla Walla University, where he teaches a class in manufacturing automation. Ralph is an amateur radio operator and enjoys woodworking, remodeling, and attempting to use his Xtracycle cargo bicycle for all his local transportation.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/234.

RESOURCES

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

R. Stirling, Twin Inverted Pendulum – CtrlBox Interface Video, Walla Walla University, www.youtube.com/user/stirlingengr.

Xilinx, "Reading Data from Configuration PROMs," XAPP694, v1.1.1, 2007, www.xilinx.com/support/documentation/application_notes/xapp694.pdf.

SOURCES

Spartan-3 Starter board
Digilent, Inc. | www.digilentinc.com

ZPU 32-bit CPU
OpenCores | www.opencores.org

WIZ810MJ Ethernet module
WIZnet | www.wiznet.co.kr/en/

PicoBlaze and Spartan-3 FPGA
Xilinx | www.xilinx.com

by Evgeni Stavinov

A Practical Parallel CRC Generation Method

Do you understand the mechanics of the cyclic redundancy check (CRC) well enough to build a customized parallel CRC circuit described by an arbitrary CRC generator polynomial? This article covers a practical method of generating Verilog or VHDL code for the parallel CRC. The result is the fast generation of a parallel CRC code for an arbitrary polynomial and data width.

Most electrical and computer engineers are familiar with the cyclic redundancy check (CRC). Many know that it's used in communication protocols to detect bit errors, and that it's essentially a remainder of the modulo-2 long division operation. Some have had closer encounters with the CRC and know that it's implemented as a linear feedback shift register (LFSR) using flip-flops and XOR gates. They likely used an online tool or an existing example to generate parallel CRC code for a design. But very few engineers understand the mechanics of the CRC well enough to build a customized parallel CRC circuit described by an arbitrary CRC generator polynomial. What about you?

In this article, I'll present a practical method for generating Verilog or VHDL code for the parallel CRC. This method allows for the fast generation of a parallel CRC code for an arbitrary polynomial and data width. I'll also briefly describe other interesting methods and provide more information on the subject.

So why am I covering parallel CRC? There are several existing tools that can generate the code, and a lot of examples for popular CRC polynomials. However, it's often beneficial to understand the underlying principles in order to implement a customized circuit or make optimizations to an existing one. This is a subject every practicing logic design engineer should understand.

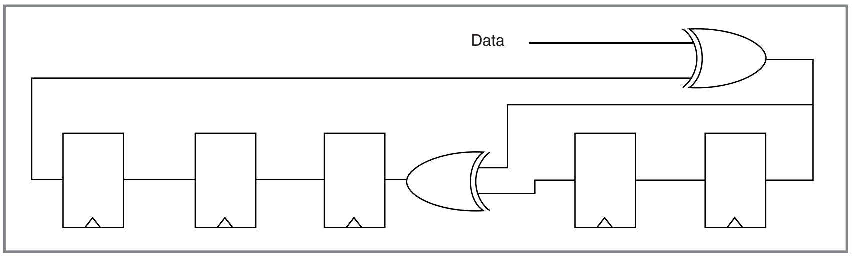


Figure 1—This is a USB CRC5 implementation as LFSR using generator polynomial $G(x) = x^5 + x^2 + 1$.

CRC OVERVIEW

Every modern communication protocol uses one or more error-detection algorithms. CRC is by far the most popular. CRC properties are defined by the generator polynomial length and coefficients. The protocol specification usually defines CRC in hex or polynomial notation. For

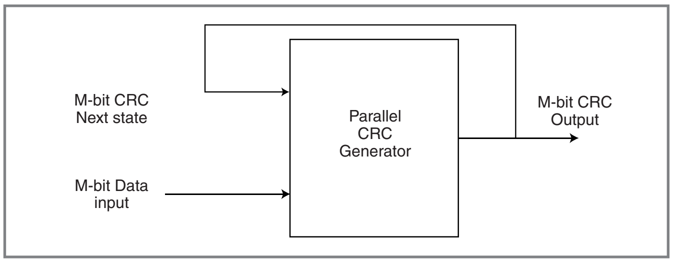


Figure 2—This is a parallel CRC block. The next state CRC output is a function of the current state CRC and the data.

example, CRC5 used in USB protocol is represented as 0x5 in hex notation or as $G(x) = x^5 + x^2 + 1$ in the polynomial notation:

$$\text{Hex notation 0x5} \leftrightarrow \text{polynomial notation } G(x) = x^5 + x^2 + 1$$

This CRC is typically implemented in hardware as a linear feedback shift register (LFSR) with a serial data input (see Figure 1).

In many cases the serial LFSR implementation of the CRC is suboptimal for a given design. Because of the serial data input, it only allows the CRC calculation of one data bit every clock. If a design has an N-bit datapath—meaning that every clock CRC module has to calculate CRC on N bits of data—serial CRC will not work. One example is USB 2.0, which transmits data at 480 MHz on the physical level. A typical USB PHY chip has an 8- or 16-bit data interface to the chip that does protocol processing. A circuit that checks or generates CRC has to work at that speed.

Another more esoteric application I've encountered has to do with calculating 64-bit CRC on data written and read from a 288-bit-wide memory controller (two 64-bit DDR DIMMs with ECC bits). To achieve higher throughput, the CRC's serial LFSR implementation must be converted into a parallel N-bit-wide circuit, where N is the design datapath width, so that N bits are processed in every clock. This is a parallel CRC implementation, which is the subject of this article. Figure 2 is a simplified block diagram of the parallel CRC.

Even though the CRC was invented almost half a century ago and has gained widespread use, it still sparks a lot of interest in the research community. There is a constant stream of research papers and patents that offer different parallel CRC implementation with speed and logic area improvements. I was searching available literature and web resources about parallel CRC calculation methods for hardware description languages (HDL) and found a handful of papers. (Refer to the Resources section at the end of this article.) However, most were academic and focused on the theoretical aspect of the parallel CRC generation. They were too impractical to implement in software or hardware for a quick HDL code generation of CRC with arbitrary data and polynomial widths.

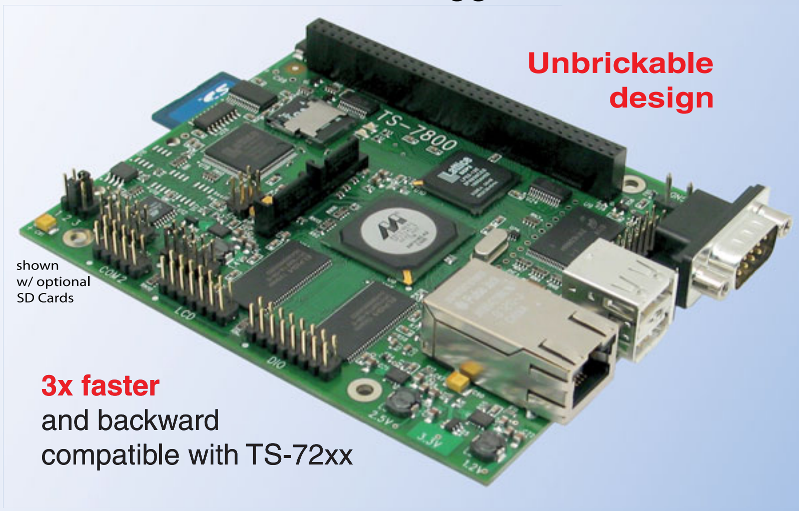
An additional requirement for the method is that the parallel CRC generator must be able to accept any data width (not only power-of-2) to be useful. Going back to the USB 2.0 CRC5

Listing 1—This Verilog module implements parallel USB CRC5 with 4-bit data.

```
//=============================================================================
// Verilog module that implements parallel USB CRC5 with 4-bit data
//=============================================================================
module crc5_parallel(
     input [3:0] data_in,
     output reg[4:0] crc5,
     input rst,
        input clk);

     // LFSR for USB CRC5
     function [4:0] crc5_serial;
           input [4:0] crc;
           input data;

           begin
              crc5_serial[0] = crc[4] ^ data;
              crc5_serial[1] = crc[0];
              crc5_serial[2] = crc[1] ^ crc[4] ^ data;
              crc5_serial[3] = crc[2];
              crc5_serial[4] = crc[3];
           end
     endfunction

     // 4 iterations of USB CRC5 LFSR
     function [4:0] crc_iteration;
           input [4:0] crc;
           input [3:0] data;
           integer i;

           begin
              crc_iteration = crc;

              for(i=0; i<4; i=i+1)
                crc_iteration = crc5_serial(crc_iteration, data[3-i]);
           end
     endfunction


     always @(posedge clk, posedge rst) begin
     I     f(rst) begin
           crc5 <= 5'h1F;
        end
        else begin
           crc5 <= crc_iteration(crc5,data_in);
        end
     end
endmodule
//=============================================================================
```

Embedded Single Board Computers

High-End Performance with Embedded Ruggedness



3x faster and backward compatible with TS-72xx

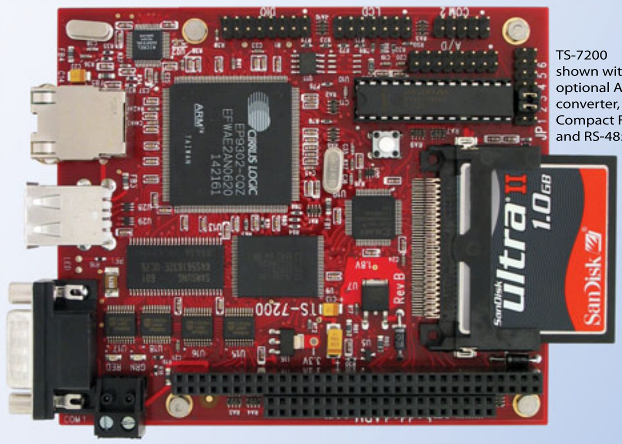
TS-7800
500 MHz ARM9

$229 qty 100

$269 qty 1

- Low power - 4W@5V
- 128MB DDR RAM
- 512MB high-speed (17MB/sec) onboard Flash
- 12K LUT customizable FPGA
- Internal PCI Bus, PC/104 connector
- 2 host USB 2.0 480 Mbps
- Gigabit ethernet
- 2 SD sockets
- 10 serial ports
- 110 GPIO
- 5 ADC (10-bit)
- 2 SATA ports
- Sleep mode uses 200 microamps
- Boots Linux 2.6 in .67 seconds
- Linux 2.6 and Debian by default

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


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

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

200 MHz ARM9 Family
Power as low as 1/4 Watt

as low as **$99** qty 100

$129 qty 1

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

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

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

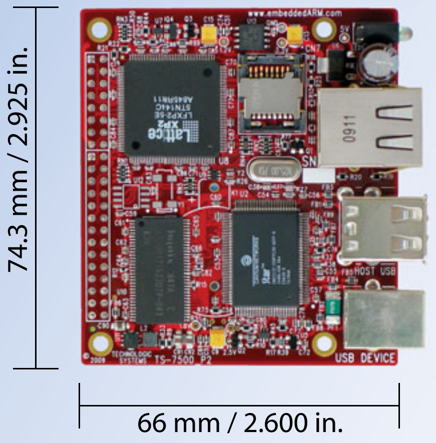
New Products

TS-7500
250 MHz ARM9

NEW!

Our Smallest Computer
at Our Best Price Point

- Low power, fanless, < 2 watts
- 64MB DDR-RAM
- 4MB NOR Flash
- Micro-SD Card slot - SDHC
- USB 2.0 480Mbit/s host (2) slave (1)
- 10/100 Ethernet
- Boots Linux 2.6 in < 3 seconds
- Customizable FPGA - 5K LUT
- Power-over-Ethernet ready
- Optional battery backed RTC
- Watchdog Timer
- 8 TTL UART
- 33 DIO, SPI, I^{2}C



$84 qty 100

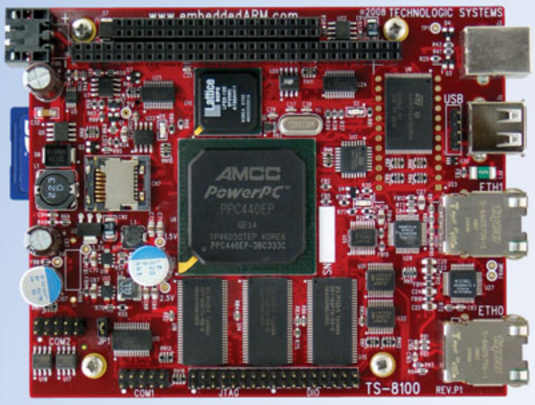
$99 qty 10

TS-8100
Ultra Reliable w/ 128MB ECC RAM

NEW!

400 MHz PowerPC
with Floating Point Unit

- POE ready
- Dual-execution unit, double-precision FPU
- Multifunctional PC/104 connector
- 12K LUT customizable FPGA
- 512MB NAND Flash
- 1 USB Host, 1 USB Device (12 Mb/s)
- Boots Linux 2.6 in < 2 seconds
- Fanless < 4W, sleep mode < 1mW
- Regulated 5-28V power input
- 2 10/100 ethernet
- 4 COM ports
- SPI & DIO
- RS485/RS422
- 2 SDHC sockets
- 5 10 bit ADC
- RTC & WatchDog
- 2 DMX Channels



shown w/ optional SD Card

$229 qty 100

$269 qty 1

Technologic Systems

We use our stuff.

Visit our TS-7800 powered website at

www.embeddedARM.com

example, a convenient data width to use for the parallel CRC of polynomial width 5 is 11 because USB packets using CRC5 are 16 bits. Another example is the 16-lane PCI Express with a 128-bit datapath (16 8-bit symbols). Because the beginning of a packet is a K-code symbol and doesn't participate in the CRC calculation, the parallel CRC data is 120 bits wide.

Before going any further into the topic of parallel CRC, I'll briefly review modulo-2 polynomial arithmetic. A polynomial is a value expressed in the following form:

$$P(x) = \sum_{i=0}^{N} P(i)x^i = p(0) + p(1)x + \dots + p(N)x^N$$

where p(i) = {0,1}.

Polynomial addition and subtraction operations use bitwise XOR. Here is an example:

$$P(x) = x^3 + x^2 + 1$$
$$Q(x) = x^2 + x + 1$$
$$P(x) + Q(x) = x^3 + x^2 + x$$

Polynomial multiplication by two is a left shift, and unsigned division by two is the right shift. Modulo-2 polynomial division is realized the same way as long division over integers. Cyclic left and right shifts are multiplication and division by (2 mod 2^n – 1).

Listing 2—This Verilog function implements the serial USB CRC5.

```
//==================================================================
// Verilog function that implements serial USB CRC5
//==================================================================
     function [4:0] crc5_serial;
         input [4:0] crc;
             input data;

             begin
                crc5_serial[0] = crc[4] ^ data;
                crc5_serial[1] = crc[0];
                crc5_serial[2] = crc[1] ^ crc[4] ^ data;
                crc5_serial[3] = crc[2];
                crc5_serial[4] = crc[3];
             end
     endfunction
//=================================================================
```

Listing 3—This pseudocode is an example of $CRC_{PARALLEL}$.

```
//==================================================================

routine $CRC_{parallel}(N_{in}, M_{in})$
    $M_{out} = M_{in}$
    for(i=0;i<N;i++)
         $M_{out} = CRC_{serial}(N_{in}, M_{out})$
return $M_{out}$

//==================================================================
```

PARALLEL CRC GENERATION

I'll start the discussion with a Verilog module that generates parallel USB CRC5 with 4-bit data (see Listing 1). A synthesis tool will do its magic and produce a circuit depending on the target FPGA or ASIC technology. However, the purpose of this article is to explain how to get a parallel CRC circuit using XOR gates and flip-flops.

Next I'll describe a practical method that I use to generate parallel CRC in a number of projects. It works on any polynomial and data size, independent of the target technology. Later I'll present other methods that have some useful properties.

The step-by-step description is accompanied by an example of parallel CRC generation for the USB CRC5 polynomial G(x) = $x^5 + x^2 + 1$ with 4-

MIN = 0	Mout[4]	Mout[3]	Mout[2]	Mout[1]	Mout[0]
Nin[0]	0	0	1	0	1
Nin[1]	0	1	0	1	0
Nin[2]	1	0	1	0	0
Nin[3]	0	1	1	0	1

Table 1—This is the matrix H1 for USB CRC5 with N = 4.

bit data width. The method—which takes advantage of the theory described in a paper by Guiseppe Campobello et al titled "Parallel CRC Realization," as well as in a paper by G. Albertango and R. Sisto titled "Parallel CRC Generation"—leverages a simple serial CRC generator and the linear properties of the CRC to build a parallel CRC circuit.

In Step 1, denote N = data width and M = CRC polynomial width. For parallel USB CRC5 with a 4-bit datapath, N = 4 and M = 5.

In Step 2, implement a serial CRC generator routine for a given polynomial. It's a straightforward process and can be done using different programming languages or scripts (e.g., C, Java, Verilog, or Perl). You can use the Verilog function `crc5_serial` in Listing 2 for the serial USB CRC5. Denote this routine as CRC_{SERIAL}. You can also build a routine `CRCparallel(Nin, Min)` that simply calls CRC_{SERIAL} N times (the number of data bits) and returns M_{OUT}. The pseudocode in Listing 3 is an example of $CRC_{PARALLEL}$.

In Step 3, parallel CRC implementation is a function of N-bit data input and M-bit current CRC state, as shown in the Figure 2. We're going to build two matrices. Matrix H1 describes M_{OUT} (next CRC state) as a function of N_{IN} (input data) when M_{IN} = 0. Thus, M_{OUT} = $CRC_{PARALLEL}$ (N_{IN}, M_{IN} = 0), and H1 matrix is the size [NxM]. Matrix H2 describes M_{OUT} (next CRC state) as a function of M_{IN} (current CRC state) when N_{IN} = 0. Thus, M_{OUT} = $CRC_{PARALLEL}$ (N_{IN} = 0, M_{IN}), and H2 matrix is the size [MxM].

In Step 4, build the matrix H1. Using the $CRC_{PARALLEL}$ routine from step 2, calculate the CRC for the N values of N_{IN} when M_{IN} = 0. The values are one-hot encoded—that is, each of the N_{IN} values has only one bit set. For N = 4, the values are 0x1, 0x2, 0x4, 0x8 in hex representation. Table 1 shows matrix H1 values for USB CRC5 with N = 4.

In Step 5, build the matrix H2. Using the $CRC_{PARALLEL}$ routine from Step 2, calculate CRC for the M values of M_{IN} when N_{IN} = 0. The values are one-hot encoded. For M = 5, M_{IN} values are 0x1, 0x2, 0x4, 0x8, 0x10 in hex representation. Table 2 shows the matrix H2 values for USB CRC5 with N = 4.

In Step 6, you're ready to construct the parallel CRC equations. Each set bit j in column i of the matrix H1—and that's the critical part of the method—participates in the parallel CRC equation of the bit $M_{OUT}[i]$ as $N_{IN}[j]$. Likewise, each set bit j in column i of the matrix H2 participates in the parallel CRC equation of the bit $M_{OUT}[i]$ as $M_{IN}[j]$.

All participating inputs M_{IN} [j] and N_{IN} [j] that form $M_{OUT}[i]$ are XORed together. For USB CRC5 with N = 4, the parallel CRC equations are as follows:

$$M_{OUT}[0] = M_{IN}[1] \wedge M_{IN}[4] \wedge M_{IN}[0] \wedge M_{IN}[3]$$
$$M_{OUT}[1] = M_{IN}[2] \wedge N_{IN}[1]$$
$$M_{OUT}[2] = M_{IN}[1] \wedge M_{IN}[3] \wedge M_{IN}[4] \wedge N_{IN}[0] \wedge N_{IN}[2] \wedge N_{IN}[3]$$
$$M_{OUT}[3] = M_{IN}[2] \wedge M_{IN}[4] \wedge N_{IN}[1] \wedge N_{IN}[3]$$
$$M_{OUT}[4] = M_{IN}[0] \wedge M_{IN}[3] \wedge N_{IN}[2]$$

M_{OUT} is the parallel CRC implementation. I used Table 1 and Table 2 to derive the equations.

The reason this method works is in the way we constructed matrices H1 and H2, where rows are linearly independent. We also used the fact that CRC is a linear operation:

$$CRC(A + B) = CRC(A) + CRC(B)$$

The resulting Verilog module generates parallel USB CRC5 with 4-bit data (see Listing 4).

OTHER METHODS

There are many other methods for parallel CRC generation. Each method has advantages and drawbacks. Some are more suitable for high-speed designs where logic area is less of an issue. Others offer the most compact designs, but for lower speed. As with almost everything else in engineering, you have to make trade-offs to bring your designs to completion.

Let's review the most notable methods. One method derives a recursive formula for parallel CRC directly from a serial implementation. The idea is to represent an LFSR for serial CRC as a discrete-time linear system:

$$X(i + 1) = FX(i) + U(i)$$

Vector X(i) is the current LFSR output. X(i + 1) is the output in the next clock. Vector U(i) is the ith of the input sequence. F is a matrix chosen according

Nin = 0	Mout[4]	Mout[3]	Mout[2]	Mout[1]	Mout[0]
Min[0]	1	0	0	0	0
Min[1]	0	0	1	0	1
Min[2]	0	1	0	1	0
Min[3]	1	0	1	0	0
Min[4]	0	1	1	0	1

Table 2—This is the matrix H2 for USB CRC5 with N = 4.

$$F = \begin{bmatrix} p(4) & 1 & 0 & 0 & 0 \\ p(3) & 0 & 1 & 0 & 0 \\ p(2) & 0 & 0 & 1 & 0 \\ p(1) & 0 & 0 & 0 & 1 \\ p(0) & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 3—This is matrix F in a formula $X(i + 1) = FX(i) + U(i)$ for recursive parallel CRC method. The values are for USB CRC5 polynomial $G(x) = x^5 + x^2 + 1$.

to the equations of serial LFSR. For example, USB CRC5 $G(x) = x^5 + x^2 + 1$ will produce Figure 3, where p(i) are polynomial coefficients. Addition and multiplication operations are bitwise logic XOR and AND, respectively.

After m clocks, the state is X(i + m), and the solution can be obtained recursively.

$$X(i+m) = F^m X(i) + F^{m-1}U(i) + \ldots + FX(i+m) + U(i+m)$$

m is the desired data width. Each row k of the X(i + m) solution is a parallel CRC equation of bit k. An important result of this method is that it establishes a formal proof of solution existence. It's not immediately obvious that it's possible to derive a parallel CRC circuit from a serial one.

Another method uses two-stage CRC calculation. The idea is that checking and generating CRC is done not with generator polynomial G(x), but with another polynomial $M(x) = G(x) \times P(x)$. M(x) is chosen so that it has fewer terms than G(x) to simplify the complexity of the circuit that realizes the division. The result of the division by M(x), which has a fixed length, is divided again by G(x) to get the CRC.

Calculating the CRC with "byte enable" is another method that is important in many cases. For example, if the data width is 16 bits but a packet ends on an 8-bit boundary, it would require having two separate CRC modules for 8 and 16 bits. The byte enable method allows for the reuse of the 16-bit CRC circuit to calculate an 8-bit CRC.

There is also a DSP unfolding technique to build a parallel CRC. The idea is to model an LFSR as a digital filter and use graph-based unfolding to unroll loops and obtain the parallel processing.

Other methods include using look-up tables (LUTs) with precomputed CRC values.

PERFORMANCE RESULTS

The logic use and timing performance of a parallel CRC circuit largely depends on the underlying target FPGA or ASIC technology, data width, and polynomial width. For instance, Verilog or VHDL code will be synthesized differently for the Xilinx Virtex5 and Virtex4 FPGA families because of the differences in the underlying LUT input sizes. Virtex5 has 6-bit LUTs, whereas the Virtex4 has 4-bit LUTs.

In general, the logic utilization of a parallel CRC circuit will grow linearly with the data width. Using the big-O notation, logic size complexity is O(n), where n is the data width. For example, each of the CRC5's five output bits is a function of four data input bits:

CRCout[i] = Function(CRCin4:0], Data[3:0])

Doubling the data width to 8 bits doubles the number of participating data bits in each CRC5 bit equation. That will make the total CRC circuit size up to 10 times bigger (i.e., 5 × 2). Of course, not all bits will double—that depends on the polynomial. But the point is that the circuit size will grow linearly.

Logic utilization will grow as a second power of the polynomial width, or $O(n^2)$. Doubling the polynomial width in CRC5 from 5 to 10—let's call it CRC10, which has different properties—doubles the size of each CRC10 output bit. The number of CRC outputs is also doubled, so the total size increase is up to 4 times (i.e., 2^2). The circuit's timing performance

Listing 4—This is a Verilog module that implements parallel USB CRC5 with 4-bit data using XOR gates.

```
//=============================================================
// Verilog module that implements parallel USB CRC5 with 4-bit
// data using XOR gates
//=============================================================
module crc5_4bit(
  input [3:0] data_in,
  output [4:0] crc_out,
  input rst,
  input clk);

  reg [4:0] lfsr_q,lfsr_c;
  assign crc_out = lfsr_q;

  always @(*) begin
    lfsr_c[0] = lfsr_q[1] ^ lfsr_q[4] ^ data_in[0] ^ data_in[3];
    lfsr_c[1] = lfsr_q[2] ^ data_in[1];
    lfsr_c[2] = lfsr_q[1] ^ lfsr_q[3] ^ lfsr_q[4] ^ data_in[0] ^
data_in[2] ^ data_in[3];
    lfsr_c[3] = lfsr_q[2] ^ lfsr_q[4] ^ data_in[1] ^ data_in[3];
    lfsr_c[4] = lfsr_q[0] ^ lfsr_q[3] ^ data_in[2];
  end // always

  always @(posedge clk, posedge rst) begin
    if(rst) begin
      lfsr_q <= 5'h1F;
    end
    else begin
      lfsr_q <= lfsr_c;
    end
  end // always

endmodule // crc5_4
//=============================================================
```

a)	
Number of LUTs	5
Number of FFs	5
Number of Slices	2
b)	
Number of LUTs	5
Number of FFs	5
Number of Slices	2
c)	
Number of LUTs	214
Number of FFs	32
Number of Slices	93
d)	
Number of LUTs	161
Number of FFs	32
Number of Slices	71

Table 3a—Logic utilization for USB CRC5,4-bit data using the "for loop" method. b—Logic utilization for USB CRC5, 4-bit data the using "XOR" method. c—Logic utilization for CRC32, 32-bit data using the "for loop" method. d—Logic utilization for CRC32, 32-bit data using the "XOR" method.

decreases because it requires more combinational logic levels to synthesize CRC output logic given the wider data and polynomial inputs.

I used free Xilinx WebPACK tools to simulate and synthesize parallel CRC circuits for USB CRC5 and the popular Ethernet CRC32. You can explore the results in the available Verilog code and project files.

Xilinx's Virtex5 LX30 is the target FPGA. Table 3a shows USB CRC5 with 4-bit data using "for loop" Verilog implementation. Table 3b shows USB CRC5 with 4-bit data using "XOR" Verilog implementation. Table 3c shows CRC32 with 32-bit data using "for loop" Verilog implementation. Table 3d shows CRC32 with 32-bit data using "XOR" Verilog implementation. Note that a single Xilinx Virtex5 Slice contains four FFs and four LUTs.

As expected, the number of FFs is five and 32 for CRC5 and CRC32. For a small CRC5 circuit, there is no difference in the logic utilization. However, for a larger CRC32, the code using the XOR method produces more compact logic than the "for loop" approach.

These synthesis results should be taken with a grain of salt. The results are specific to the targeted technology (ASIC or FPGA family) and synthesis tool settings.

PARALLEL GENERATION

The parallel CRC generation method leverages a simple serial CRC generator and the linear properties of the CRC to build $H1_{NxM}$ and $H2_{MxM}$ matrices. Row [i] of the H1 matrix is the CRC value of N_{IN} with a single bit [i] set, while $M_{IN} = 0$.

Row [i] of the H2 matrix is the CRC value of M_{IN} with a single bit [i] set, while $N_{IN} = 0$. Column [j] of the H1 and H2 matrices contains the polynomial coefficients of the CRC output bit [j].

I've used this method successfully in several communication and test-and-measurement projects. An online parallel CRC generator tool available at OutputLogic.com uses this method to produce Verilog or VHDL code given an arbitrary data and polynomial width. A similar method is also used to generate parallel scramblers. Perhaps I'll cover the topic in a future article. ■

POP QUIZ

If you've read this article carefully, you should be able to solve the following problem.

Problem: Consider the polynomial G(x) = x + 1. What well-known error detection code does this polynomial represent? Derive a parallel equation of this polynomial for 8-bit data input. Hint: Draw a circuit with serial data input and think about how the output depends on the number of "1" bits in the input datastream.

The answer is available on the Circuit Cellar *FTP site.*

Evgeni Stavinov (evgeni@outputlogic.com) is a system design engineer for Xilinx who holds an MSEE from USC and a BSEE from The Technion — Israel Institute of Technology. He has more than 10 years of design experience in the areas of FPGA logic design, embedded software, and networking. Evgeni worked for CATC, LeCroy, and SerialTek designing test and measurement tools for USB, Wireless USB, PCI Express, Bluetooth, SAS, and SATA protocols. He also created OutputLogic.com—a web portal that offers online tools for FPGA and ASIC designers—and serves as its main developer.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/234.

RESOURCES

G. Albertango and R. Sisto, "Parallel CRC Generation," *IEEE Micro*, Vol. 10, No. 5, 1990.

G. Campobello, G. Patane, M. Russo, "Parallel CRC Realization," http://ai.unime.it/~gp/publications/full/tccrc.pdf.

R. J. Glaise, "A Two-Step Computation of Cyclic Redundancy Code CRC-32 for ATM Networks," *IBM Journal of Research and Development*, Vol. 41, Issue 6, 1997.

A. Perez, "Byte-wise CRC Calculations," *IEEE Micro*, Vol. 3, No. 3, 1983.

A. Simionescu, "CRC Tool: Computing CRC in Parallel for Ethernet," Nobug Consulting, http://space.ednchina.com/upload/2008/8/27/5300b83c-43ea-459b-ad5c-4dc377310024.pdf.

FEATURE ARTICLE

by Matt Oppenheim

Multichannel Touch Sensors

Implement Scalable Capacitive Touch Sensing

Tired of switching? Try touching. It's now easy to implement a versatile multichannel touch sensor in various projects. Here you learn how to design a 20-channel device of your own.

One of my many misdemeanors as a child was to scuff across the carpet on a dry day and then touch my sleeping cat on the nose. The resulting static discharge resulted in a few choice words of feline outrage. Several decades later, I have learned to exploit the body's ability to act as a capacitor for more useful purposes.

In this article, I'll explain how easy it is to implement a versatile multichannel touch sensor with Atmel's Qprox QT1103 10-channel capacitive touch sensor IC, which uses QTouch technology. The project progressed through the traditional path. First, I hacked up the manufacturer's demo board to prove the capability of the technology. I then produced custom PCBs. Next, I designed a 20-channel device using two QT1103 chips. I followed that up with a 100-channel device using five of the 20-channel sensor boards. Although I'll focus on the 20-channel device in this article, I'll also mention the main technical issues I dealt with while designing the 100-channel instrument.

HOW IT WORKS

With the new all-in-one "black box" ICs, it is easy to ignore the underlying physics of how the technology works. But an understanding of the principles makes the difference between building a robust device and one with intermittent problems.

By definition, a capacitor is something that stores charge. You are all familiar with the classic two-plate symbol for a capacitor. The charge is stored on the two plates and the space between them may be empty air or filled with material (called dielectric). The bigger the plates, the more charge that the capacitor can hold. A finger also has the ability to hold charge, as my cat found out. So, when you touch one of the plates of a capacitor, you are effectively adding to the amount of charge that it can hold—increasing the total capacitance of the system. Figure 1 depicts how a finger can act as a capacitor interacting with the reference capacitor on the QTouch circuit's input pins.

Before the advent of all-in-one ICs like the QT1103, this capacitance was measured by sending a pulse along the touch sensor line and comparing the charge left after a certain amount of time with a reference level, using a comparator to digitize the output. Sometimes the internal comparator in a microprocessor would be used.

The increased capacitance due to your finger is detected

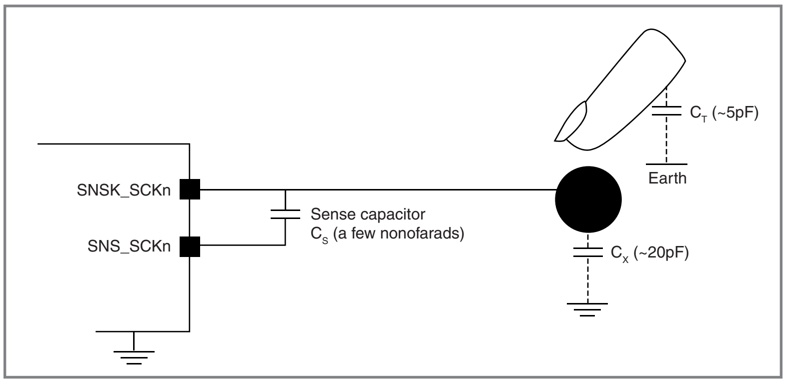


Figure 1—The QTouch measuring circuit. The finger acts as a small capacitor that adds to the capacitance already present between the two sensor pins. (Source: STMicroelectronics QST104 datasheet, which uses the same QProx charge transfer capacitive technology as the QT1103).

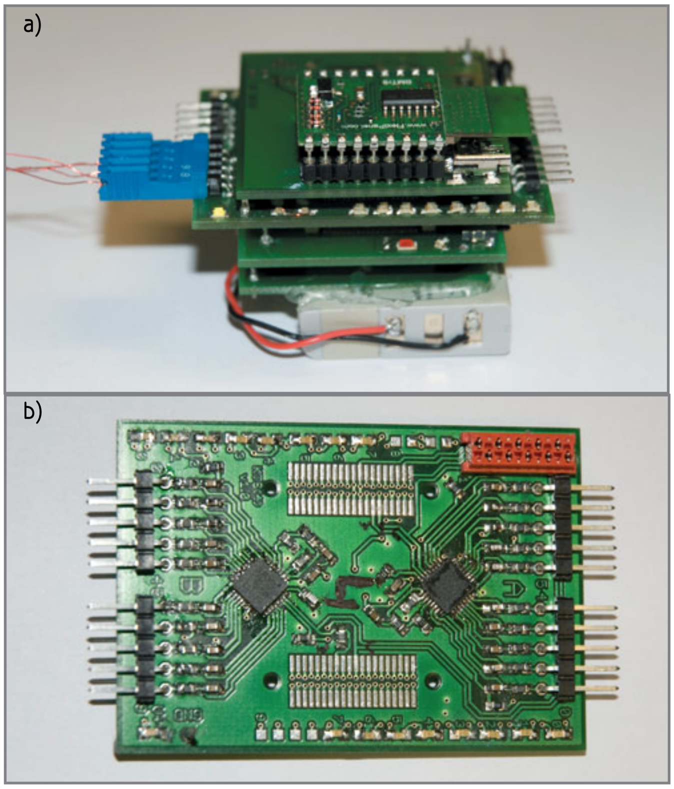


Photo 1a—The Bluetooth DIP module is on the top. The blue connector on the left shows how sensor lines attach to the 0.1″ pins visible on the edges of the sensor board. Underneath the sensor board is the dsPIC33FJ128GP706 controller board. The bottom board houses the battery-management hardware and a lithium battery. b—A close-up of the sensor board. The QT1103 chips are the two black squares.

by a corresponding increase in the charge left after the test interval. This capacitance is over the preset limit, so the comparator indicates that the sensor has been touched. It clearly takes up a lot of PCB real estate and a number of components. The advantage of chips such as the QT1103 is that they offer multiple channels with a minimum of interface lines and they take care of the sensor-triggering processing. The QTouch technology uses a sophisticated arrangement of pulses to detect the change in capacitance so as to reduce cross-sensor interference and power consumption.

DESIGN EVOLUTION

I was tasked to produce devices that would enable the Human Computer Interaction (HCI) group at Lancaster University's Infolab21 to augment surfaces with a touch-sensing capability. The first project involved modifying a QT1103 demo board to enable it to be wired to 10 keys on a standard PC keyboard. The demo board has a single QT1103 with LEDs to display the status of each channel. You can connect your own sensor pads or wires to the board by removing the PCB keypad that comes with it. This exposes 0.1″ headers. It enabled me to connect laminate wires directly to the sensor lines. The result was a classic kludge of a demo board, breadboard, and an FTDI UART-to-USB interface module to get the data to a PC. You can access the QT1103 communication lines on the demo board, but you need to solder in your own 0.1″ pitch connector. I used this to hook up my Microchip Technology dsPIC development board and get to grips with the communication protocol.

Once I was confident in the technology, I designed a PCB for a 20-channel device that used two of the QT1103 chips. This connected to a Microchip Technology dsPIC33FJ128GP706 controller board, a battery board, and a communications board with a Bluetooth module. Photo 1a is the entire device. Photo 1b is a close-up of the sensor board.

I recycled a dsPIC break-out board and battery controller board from an earlier project. Using a 16-bit digital signal processor (DSP) was overkill for this application, but it enabled me to complete the project faster than if I had designed a new board. I knew I could also reuse the C libraries I had written for that DSP. Plus, I knew that if the end application turned out to be a mass-produced commercial device (rather than a custom research instrument), the component cost would be more critical and mandate a cheaper design.

After the successful 20-channel design, I went on to enhance 100 keys of a standard PC keyboard with touch sensitivity using five of the 20 channel boards controlled by a single dsPIC. The touchpad on the back of each key was made by pressing adhesive-backed foil onto a laminated copper wire connecting to the touch sensor. The five sensor channel boards and their interconnections to the controlling dsPIC board appear in Photo 2.

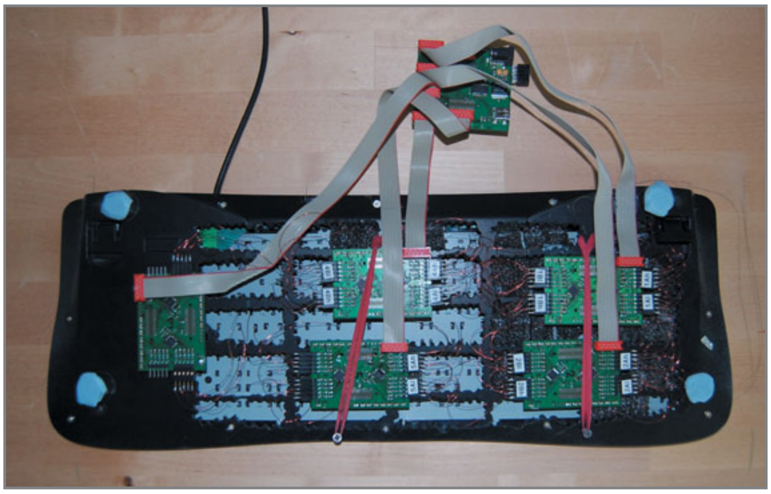


Photo 2—The rear of the 100-channel touch sensor enabled keyboard. Five of the 20-channel sensor boards are connected to the base of the keyboard. The dsPIC controller board is connected with ribbon cables to each of the sensor boards.

QT1103 DESIGN

The QT1103 is advantageous because it requires only a single sensor line for each of the touch sensors. Some other chips require a reference ground line for each channel, which is fine for a static PCB touchpad layout, but would give me a wiring headache because the sensor lines are thin-laminated wire stuck onto the back of a keypad.

Each sensor line has a reference capacitor and resistor. You can alter a line's sensitivity by changing the capacitance. By decreasing the reference capacitor, the channel's sensitivity is increased (so that it will trigger through a sheet of plastic, for example). If you are using large pieces of copper for sensor pads, the value of the reference capacitor may need to be increased to reduce the triggering threshold.

Each chip interfaces with a microcontroller using two or three I/O lines. The *CHANGE line goes low whenever the QT1103 detects a change on any of the sensor lines. The host microcontroller then requests the new status information by sending the character "P" to the QT1103. The chip then sends two 8-bit bytes. These encode the status of the 10 sensor lines (see Figure 2). Then the *CHANGE line returns high. The chip enables you to send and receive data using two separate lines—or to do both on the same line. I used the single line for both transmission and reception. Figure 2 shows the data protocol. The transmission protocol is standard UART. Don't be fooled by the term "one-wire" in the datasheet. It means standard UART, but using only one wire to both transmit and receive.

The transmission lines need to be held high with 100-kΩ resistors. Obviously, I had to change the status of the data line from transmit to receive for each cycle of sensor status updates. If for some reason your microcontroller can't send and receive on the same line, you have to use separate wires for sending and receiving.

The QT1103 datasheet shows a number of options that can be set with resistors (e.g., how quickly the chip enters its low-power mode). The chip can be set into a "simplified" mode using a single resistor—R55 in Figure 3. Sufficient for most applications, this mode is used by the demo board. The only option to consider in the simplified mode is whether to enable the adjacent key suppression feature. This prevents multiple adjacent channels from triggering from a single touch, which can happen with closely spaced keys or with a wet surface. This option is set by another 1-MΩ resistor connected to line SNS0 (R in Figure 3). If this is pulled high, the option is on. If it is pulled low, it is set off. The datasheet includes the full details and the other options.

For my purposes, one of the

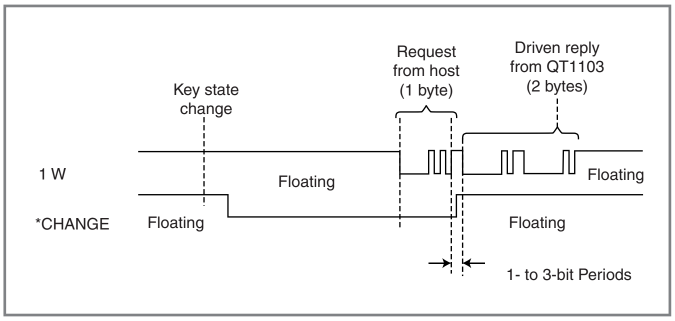


Figure 2—The QT1103 data protocol. The controller sends the character "P" once the *CHANGE line goes low. Then the sensor chip sends the status of the 10 channels in 2 bytes. (Source: Qprox QT1103 datasheet)

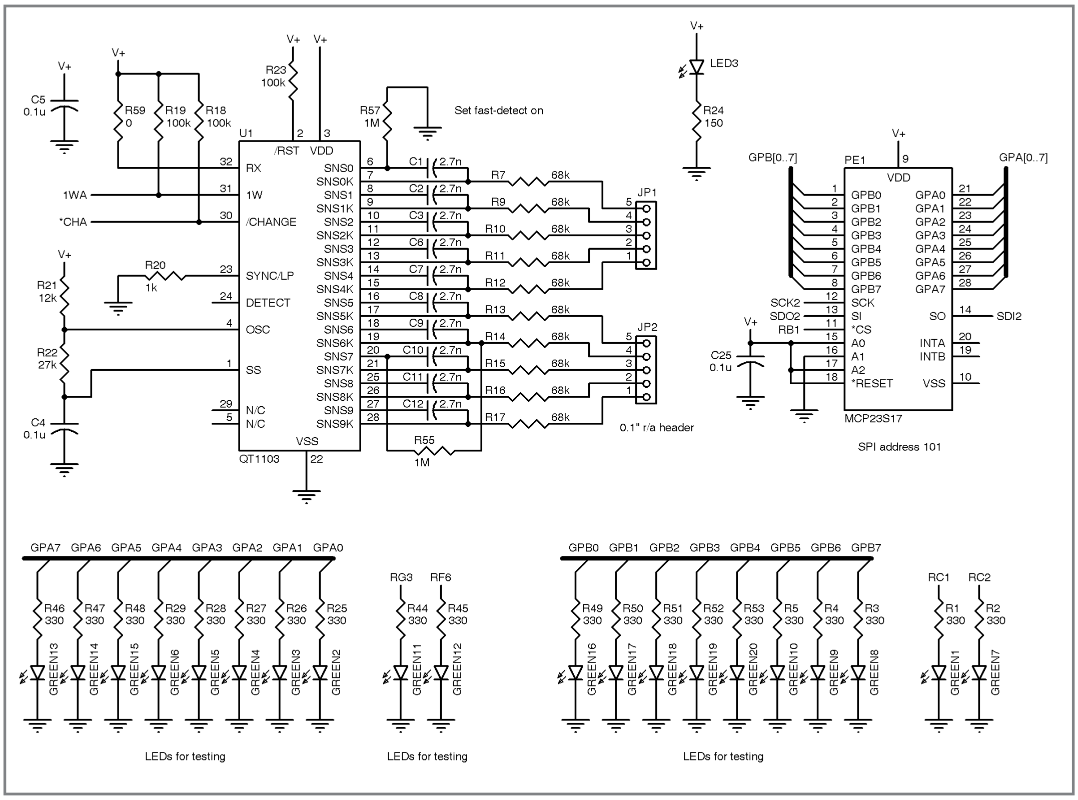


Figure 3—Each sensor channel on the QT1103 requires a resistor and reference capacitor. The MCP2317 can directly power the status LEDs.

Elektor USA Edition
www.elektor.com/usa

"Elektor? Prescribed reading for our R&D staff because that's where we need professional guidance for microcontroller technology."

– Frank Hawkes, 39, development engineer –

Subscribe to Elektor now!

Join the fascinating world of electronics worldwide!

Creative & fresh articles since 1961

11 Issues including the summer double issue for just $39.95*

- Electronics projects to delight and educate in diverse areas of electronics
- Each project is built, tested, and approved by Elektor's laboratory staff
- Access to a wide range of products to support magazine projects in our online shop
- Cheaper than 11 issues from the newsstand: Save 57% of the cover price of $92.45

* Offer available in US and Canada only. Canada add $11 per year.

Take out your subscription now:

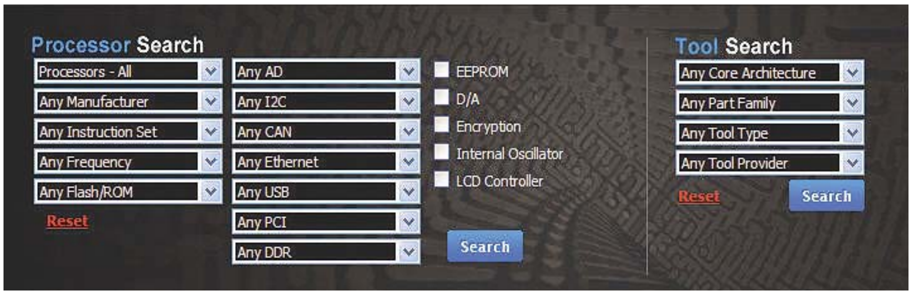
www.elektor.com/usa • Phone 860-875-2199

elektor electronics worldwide

Pick a Chip. Any Chip.

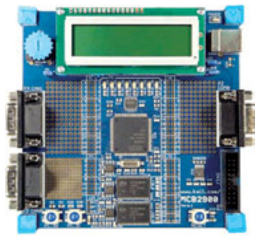
Find a Solution to your next Embedded Challenge.
Do the Research you should, but never had time for.

Embedded Developer's intuitive research engine helps you speed your chip evaluation time. You don't have to know the manufacturer, chip family or part number--just select the features you want and let us do the rest.



Part Number	AT91SAM7X	MCF5208	LPC2923
Manufacturer	ATMEL	freescale semiconductor	NXP
Core Variant	ARM7TDMI	ColdFire V2	ARM968E-S
Flash	262144	0	262144
RAM	65536	16384	16384
Max. Freq.	55	166	125
Dhrystone MIPS	50	159	156
Timer Bits	16	32	32

We help you research your best option. Nowhere else can you compare your best options side-by-side from different manufacturers. Click on the device you want, and a product page lets you select Distributor Buy / Quote options, send RFQs, download datasheets, and more. Plus--Hearst stock check gives you up-to-date inventory on every device.



Once you have the chip that meets your needs, review and compare the hardware and software development tools that support it from multiple manufacturers, and buy them on-line through our shopping cart.

Shave days off your schedule with Embedded Developer, the only site in the world where you're only clicks away from finding the chips and tools to get you up and running, quickly. Try EmbeddedDeveloper.com, or EmbeddedDeveloper.cn in Chinese.

EMBEDDEDDEVELOPER.COM
FIND. COMPARE. BUY.

EMBEDDEDDEVELOPER.CN
查询． 选型． 采购

The Sites for Engineers with a Job to Do.

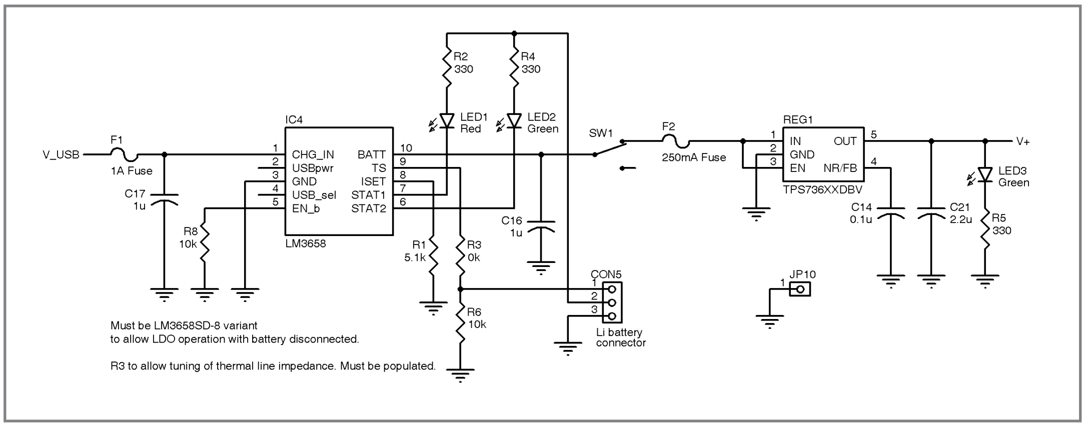


Figure 4—The LM3658B battery-management IC allows for the concurrent recharging of a Lithium cell from a powered USB hub while the board is in normal operation. The chip also works as an LDO without a battery attached.

QT1103's most useful features is that the sensor recalibrates each time it powers on. So, if you are using trailing wires for the sensor channels, which inevitably become tangled and start causing misfires, all that you have to do is cycle the power. You don't even have to untangle the wires.

HARDWARE CONSTRUCTION

I completed the schematic capture and a two-layer PCB layout with EAGLE v5.2. I packed the boards as densely as possible because I was charged per square inch for PCB manufacturing (see Figure 3).

I put ground planes under all of the sensor line tracks and kept them as short as possible because they carry an analog signal. The main construction-related challenge was soldering on the QT1103 chips because they are only available in QFN32 packages—no legs! After a little practice, I cracked this by pre-tinning all of the PCB pads (but not the pads on the IC), applying some liquid flux, carefully positioning the IC (a head torch helped me see), and then tacking down a few of the pads by heating the exposed PCB pads.

When your chip is in place, run the tip of your soldering iron down the chip's edge. The solder resist should stop bridges forming. You may need to build up the joints in some places. Then wipe down the edge again with your iron tip. If you get any bridges, keep cleaning your iron tip and wiping down the pads to remove the excess solder and leave a neat joint. Otherwise, reach for your solder wick!

I placed right-angle 0.1″ headers connectors to access the sensor lines. The 20-channel device needed to be a stand-alone unit, so I placed 40-way 0.8-mm Tyco connectors on the boards to enable them to connect into a stack with the dsPIC, battery, and communications boards, which enabled a modular yet compact device. I can reuse individual boards from the stack for future projects.

On the battery board, I used a National Semiconductor LM3658B battery-management IC circuit with a rechargeable lithium battery. Figure 4 shows the battery-management circuit. The LM3658B was another of the leadless QFN packages, but I was getting the hang of soldering them. I was tempted to put a via in the PCB's base so I could hand-solder the ground pad on the chip's base. But in the end, after soldering using a fine bit, I used a Weller Pyropen with a narrow hot air nozzle to make sure it was all stuck down properly. Any problems that I had with this circuit were solved by applying a little more hot air to the QFN package with the Pyropen.

The output voltage from the management IC is fused and regulated. I always put fuses onto my circuits. I recharge the battery using a mini-USB connector on a board that clips onto the battery board. The "B" version of the LM3658 family allows you to use the battery management IC powered from the USB line without a battery attached, which is useful for testing purposes. I placed a ground solder pin on each board to ground an oscilloscope probe or multimeter in case debugging is required.

BLINKING LEDs

Light goes on. Light goes off. Electronics engineer happy.

Having an LED illuminate each time a channel was activated enabled me to verify that the sensor wires were correctly connected to their keypads and responding to touch. Plus, I can never have too many blinking LEDs! This would be an expensive luxury for a mass-produced commercial device, but these boards were designed to be as flexible as possible for research projects where the goal posts often move.

I incorporated a useful Microchip Technology MCP23S17 port expander IC to enable the maximum number of flashing lights for the minimum amount of interface. I sprung for an extra data line because I used the SPI incarnation of the MCP23S17 as opposed to the I²C version. Figure 3 shows how I connected the MCP23S17.

The chip has 16 channels of I/O divided into two banks (A and B) of eight channels. Each channel can source or sink up to 25 mA—plenty enough to light up an LED. There are three SPI address lines on the chip (A0, A1, and A2), but by default they aren't used. They are enabled by enabling the relevant register bit. I was using all of the ports as outputs to drive LEDs, but you can also configure any of them to be an input and read the status via a SPI.

To use the MCP23S17 to light up LEDs, several internal registers need to be set up. The ports are configured as outputs and then the relevant bits of the output port must be set high. In the chip's default state, the register address toggles automatically between the two eight-port status banks, reducing the amount of commands needed. Refer to the code on the *Circuit Cellar* FTP site to see how I implemented

“I hacked up the manufacturer’s demo board to prove the capability of the technology. I then produced custom PCBs. Next, I designed a 20-channel device using two QT1103 chips. I followed that up with a 100-channel device using five of the 20-channel sensor boards.”

this. I plan to use the versatile chip for future projects.

Obviously, with 20 channels and a single 16-port expander I was still a few channels short! For the 20-channel device, I used some spare I/O lines from the dsPIC to enable each sensor line to have a dedicated status LED. Once I got onto the 100-channel device, I didn’t have enough spare lines to display each channel with a separate LED. So, some of the LEDs were used for two channels—which worked fine for testing that all of lines were working.

COMMUNICATIONS

The 20-channel design required Bluetooth communications. I used a Linkmatik 2.0 DIP module, which interfaces to the dsPIC using a standard UART.

The 100-channel multi-board design required a USB interface to a PC. I used an FTDI FT232RL UART-to-USB IC and a miniature USB connector. The same UART communication lines from the dsPIC and C code worked virtually unchanged for both communication devices as they are both seen as a standard UART interface by the controller.

I placed a 10-pin Micromatch connector on each of the 20 channel sensor boards so they can connect to a connector/power/USB board. The connectors are keyed and locking, which makes for a more reliable system than the traditional 0.1″ header connectors so beloved of research instrumentation. The dsPIC controller board connects to the back of this board using 40-way Tyco 0.8-mm pitch connectors. To enable all of the boards to work together, the SPI addressing on the MCP23S17 boards had to be enabled. Each of the sensor boards used different data lines for communications. No one was more surprised than me when it all worked!

CODING

I use the Mikro C_dsPIC compiler. A free version is available from Mikroelektronika, the same folks who built my dsPIC development board. It comes with a good IDE and a range of useful libraries. I had to implement a software UART interface for each of the QT1103 chips. The software UART library allowed me to define the same I/O line to both transmit and receive without throwing up errors. The IDE interfaces directly with the Mikroelektronika ICD programmer, which I used, but you can import the hex file to MPLAB and use the Microchip ICD2 without any problems. If you make your own programming cable, you can

Listing 1—When the *CHANGE line goes low while the 1-W line is still held high, a service routine is called to obtain the status for all of the 10 sensor channels for the chip. The status is displayed on the board’s LEDs and transmitted via UART to the user.

```
// services touch sensor when a new sensor status is detected
  void service_QT1103(unsigned int *port, short uart_port, short board)   {
       unsigned short bytea=0x00, byteb=0x00;
       int reca, recb;
       soft_uart_init(port, uart_port, uart_port, UART_BAUD, 0); // set up uart on 1W line
       soft_uart_write(0x50);                                    // write 'P' on 1W line
       bytea = soft_uart_read(&reca);                            // read sensor status byte a
       byteb = soft_uart_read(&recb);                            // read sensor status byte b
       Display_bytes(bytea, byteb, board, port);                 // display to LEDs
       code_sensor_status(bytea, byteb, port, board);            // transmit status
} // end service_QT1103

// main while loop
while (1) {
// Board 1
   if (!bit_read(PORTD,CHA1)&& bit_read(PORTB,ONEWA1)) {          // QPROX sensor A /CH goes low
       service_QT1103(&PORTB, int_onewa1, board1);
       }
   if (!bit_read(PORTD,CHB1)&& bit_read(PORTD,ONEWB1)) {          // QPROX sensor B /CH goes low
       service_QT1103(&PORTD, int_onewb1, board1);
       }

}  // end while
```

use the Microchip ICD2 directly on the same header pins as the Mikroelektronika ICD uses.

The main servicing loop is shown in Listing 1. Initially, I used a button debounce algorithm on the *CHANGE line, which triggered when that line went low. I found it more reliable and faster to look for that line to go low at the same time as the 1-W data line was still high—especially on the 100-channel design. Using this method of checking two lines allowed individual boards to be removed from the 100-channel device and for the remainder to continue to operate correctly.

The `service_QT1103` function sets up a soft UART interface on the 1-W line for the chip and writes a character "p" on the line to trigger the QT1103, which sends the sensor status in 2 bytes. These bytes are passed to two further subroutines. `Display_bytes` displays the activated channels on the board's LEDs. `code_sensor_status` adds board and sensor identification bits to the status bytes and then sends them to be transmitted to the user.

A lot happens each time a channel is triggered, but the sensor lines are only being triggered at the speed that a human can activate them, which is a glacial speed for today's microcontrollers. My main problem was that none of the hardware interfaces would work! I eventually found that the compiler had some errors in the definition file for my dsPIC, at least this was detailed on the manufacturer's forum. During the interim, I got all of the hardware working by using the software interface libraries. I knew the hardware interfaces worked because I had them all running using a different compiler during a previous project.

SWITCH TO TOUCH

The first piece of design advice I received was to minimize the number of mechanical switches in a design because they would inevitably become a mode of failure. Now I can do this cheaply and easily.

I am not going to predict the end of the mechanical switch, especially for power applications, but I do predict that we are going to see a lot more use of capacitive touch sensors. I can see many potential uses for this technology. One of the more imaginative ideas I have been asked about is the viability of implementing a touch sensor on the entrance to a bird-feeder to help monitor its use. I am not sure if my long-deceased cat would approve of this application, but he suffered for a noble cause by teaching me the fundamentals of how the body acts as a capacitor.

Matt Oppenheim (matt.oppenheim@gmail.com) holds an MSc in mechatronic systems engineering from Lancaster University. After working for 12 years in the marine and land seismic survey industry, he is now designing and prototyping instrumentation for research groups at InfoLab21, Lancaster University. Matt's technical interests include embedded design, but his first love is analog. Matt no longer owns a cat.

PROJECT FILES

To download code and additional schematics, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/234.

RESOURCES

F. Block, H. Gellerson, M. Oppenheim, and N. Villar, "Touch-Display Keyboards and their Integration with Graphical User Interfaces," Demonstration, UIST, 2009.

Microchip Technology, "MCP23017/MCP23S17 16-bit I/O Expander with Serial Interface," DS21952B, 2007, ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf.

National Semiconductor, "LM3658B Dual Source USB/AC Li Chemistry Charger IC for Portable Applications," LM3658, 2006, www.national.com/ds/LM/LM3658.pdf.

Quantum Research Group, "QT1103 QTouch 10-Key Sensor IC," QT1103_3R0.03_0607, 2007, www.qprox.com/assets/Downloadablefile/qt1103_3r0.03(1)-15736.pdf.

SOURCES

Qprox QT1103 Capacitive touch sensor IC
Atmel Corp. | www.atmel.com

Linkmatik 2.0 DIP
FlexiPanel | www.flexipanel.com

FT232R USB UART
FTDI | www.ftdichip.com

dsPIC33FJ128GP706 Controller board and MCP23S17 IC
Microchip Technology, Inc. | www.microchip.com

MikroC for dsPIC30/33 and PIC24
MikroElektronika | www.mikroe.com/en/compilers/mikroc/dspic/

LM3658B Battery management IC
National Semiconductor | www.national.com

Toothpick 2.0 Bluetooth Transceiver
RF Solutions | www.rfsolutions.co.uk

40-way 0.8-mm Pitch connectors
Tyco Electronics | www.tycoelectronics.com

FEATURE ARTICLE

by Miguel Sanchez

Three-Axis Stepper Controller

Stepper motor controller projects are posted all over the Internet, but many are platform-dependent. Now you can build an Arduino-based, platform-independent driver board that enables you to break free from platform constraints.

Some of you will remember the vertical plotter design I described in my March 2008 article titled "Vertical Plotter System" (*Circuit Cellar* 212). I used an off-the-shelf USB stepper motor controller for that project. One of the motors was a bipolar stepper, which gave me a little trouble when I tried to get it working. While a unipolar stepper motor can be driven by a bipolar driver, the opposite is not true. Therefore, if you plan to build a stepper motor controller, a bipolar one is a safer bet because you can use it with both unipolar and bipolar stepper motors. But keep in mind that a unipolar driver is simpler and probably cheaper.

Many stepper motor controller projects are available on the Internet. The one I bought was Windows-only, so I decided to build my own platform-independent driver board. A new USB-based driver board was on its way.

ARDUINO PLATFORM

I've been playing with some Arduino boards for a while now. Arduino is open-source hardware built around an Atmel ATmega168 processor (see Photo 1). Schematics, bootloader code, printed circuit board CAD files, the software development environment, and more are available as free downloads.

Shops around the world sell Arduino boards in kit form. Finished boards are also available if you don't want to use your soldering iron. The board has 14 digital pins, six of which can use hardware-based PWM, and six pins that can be used as analog inputs (10-bit ADC). It also has an asynchronous serial port. The original Arduino used a serial port to upload programs from the PC running the development environment. Newer Arduinos use a USB connection instead. But to keep

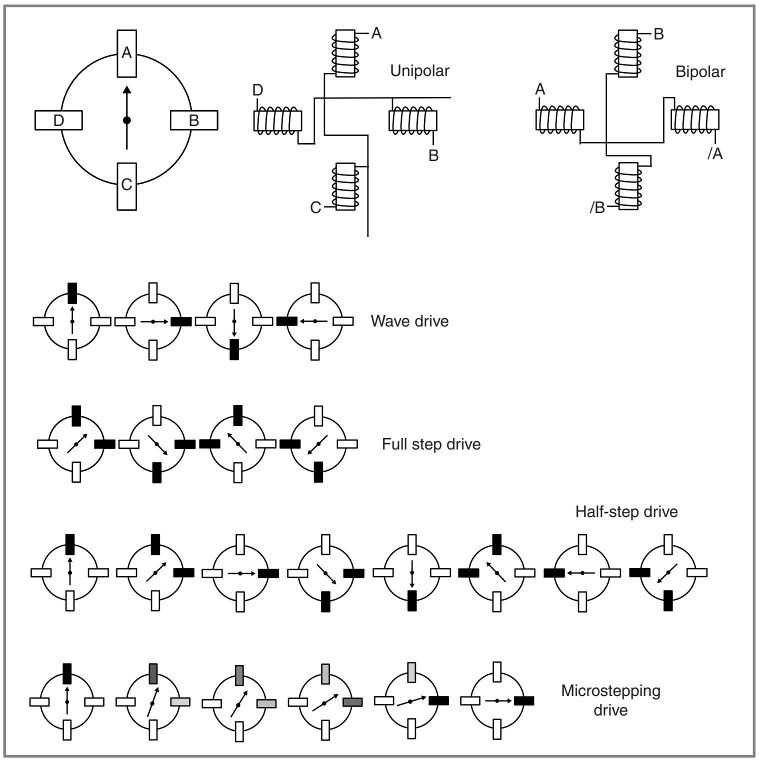


Figure 1—This is a wiring diagram of a four-pole stepper motor. Different drive modes are shown.

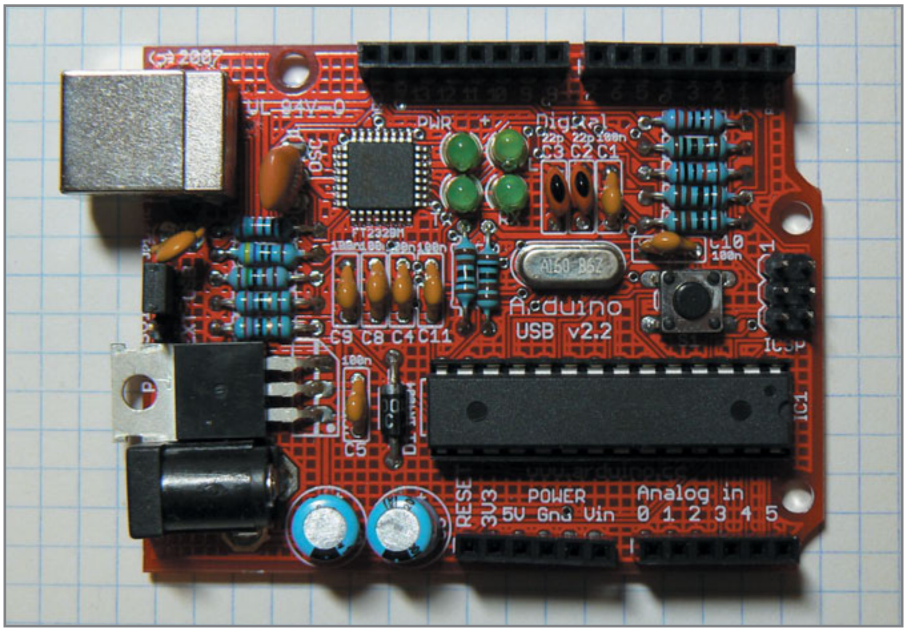


Photo 1—The Arduino board features an Atmel ATmega168.

things simple, they use a Future Technology Devices International FT232 chip that acts as a USB-to-serial converter, so bootloader code is not changed. The good thing about the FTDI chip is that it is already supported on many operating systems, so no additional drivers need to be installed. An additional advantage associated with using a USB port over serial is that you can borrow power from it, so an external power supply is not needed unless the project requires more than 500 mA at 5 V.

Another key element of the Arduino board is that all the available I/O and power pins have attached headers so you can plug in other boards on top. Several of these boards—called "shields"—are available to add extra functionality to a project like a motor controller, an Ethernet interface, or a ZigBee-based wireless interface. In some cases, you can stack more than one shield on the Arduino board.

Go to www.arduino.cc for more information about Arduino and add-ons. The list of contributors and developers is a long one. As you'll see, several pieces have been put together to create an easy-to-use platform for embedded development than can be used by designers and even artists! For instance, many people are interested in physical computing, which is a form of art that enables you to interact with artistic designs (e.g., human-to-light/audio interaction). Such systems require sensors, actuators, and software for an embedded controller.

The Arduino development environment is written in Java and runs on many different operating systems. It's based on another open-source tool from MIT called Processing. Programs are created in C or C++ because the compiler in the background is the GNU AVR compiler for Atmel 8-bit microcontrollers. Built-in peripherals (e.g., timers, PWM, ADCs, and so on) are abstracted by simple, easy-to-use constructs like `analogRead` or `digitalWrite`. Most of the hardware configuration is automatic. Serial communication is also well supported. The development environment handles the process of compilation, linking, and uploading the code to the microcontroller with just one click.

Arduino system hardware is simple, but it is also worth mentioning that a Reset button and an in-circuit serial programming six-pin port are included in the board too. The former may be useful for restarting the system and the latter for programming the boot-loader code on a brand new microcontroller. The rest of the programming will be done through the development environment and serial communications. There is a jumper to select whether external power supply or USB power is used.

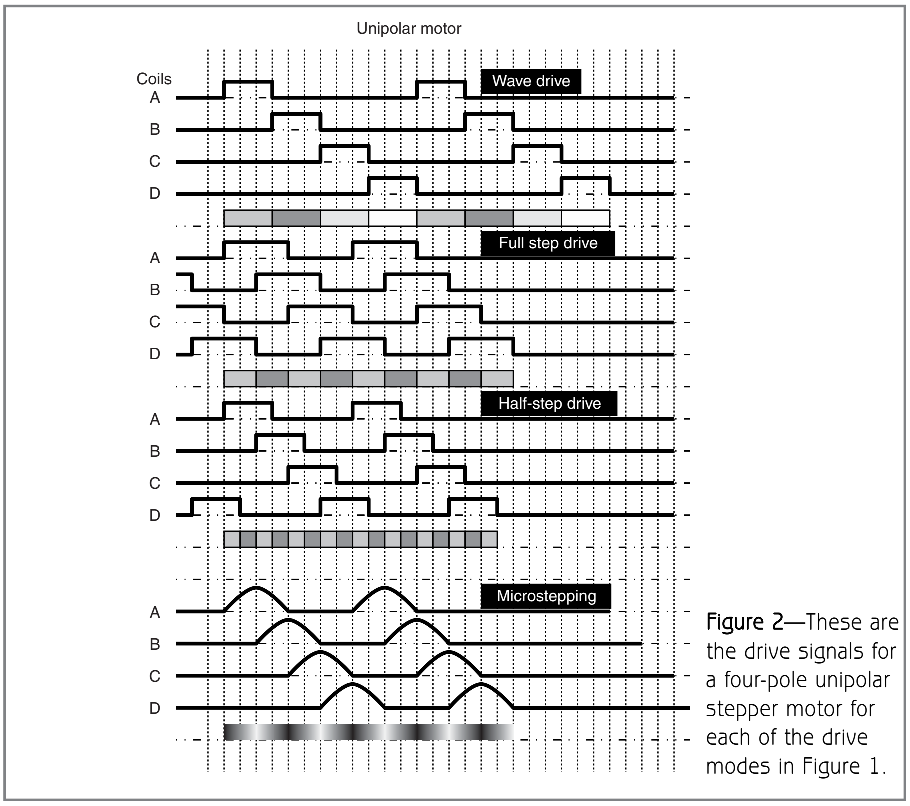
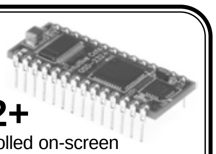


Figure 2—These are the drive signals for a four-pole unipolar stepper motor for each of the drive modes in Figure 1.

DO IT YOURSELF

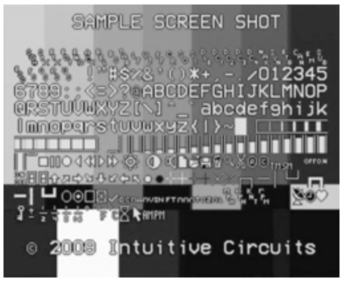
The kit was affordable at about $20, so I used an Arduino board to handle the USB interface and the protocol

New!



OSD-232+

RS-232/TTL controlled on-screen composite video character and graphic overlay in a small 28 pin dip package.



Intuitive Circuits
www.icircuits.com
(248) 588-4400

Add USB to Your Designs

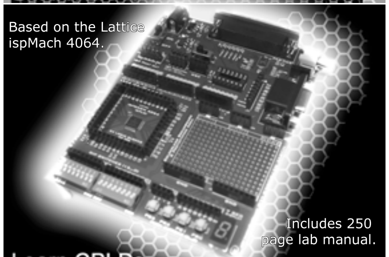
Chips, code, protocols, embedded hosts, wireless options, debugging, **USB 3.0** and **SuperSpeed** too!

USB Complete
The Developer's Guide
Fourth Edition
Jan Axelson

ISBN 978-1-931448-08-6 $54.95
Lakeview Research LLC **www.Lvr.com**
By the author of *Serial Port Complete*

mach64
PROGRAMMABLE LOGIC STARTER KIT
NURVE NETWORKS LLC

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

Command	Comment	Response
H X	Goes to home location on X axis.	OK/ERR
S X NNN	Set motor X speed to NNN rpm	NNN/ERR
? X	Query the number of steps of motor X	NNNNNN/ERR
F X NNNNN	Go forward NNNNN steps on motor X	NNNNNN/ERR
B X NNNNN	Go backward NNNNN steps on motor X	NNNNNN/ERR

Table 1—This is a list of available commands for the motor controller. Interpreter code is shown in Listing 2.

with the PC. I created a new shield with three bipolar stepper motor drivers to complete the system. I found shields that could power one or even two stepper motors, but a driver for three motors was not available.

A stepper motor is a special type of motor in which the rotating action is not always the desired outcome (see Figure 1). Typical DC or AC motors are designed to rotate as soon as they are powered. They rotate because the motor's design creates a rotating magnetic field the rotor motor follows. Stepper motors are designed to rotate if such a rotating magnetic field is provided, but they can also remain still at a fixed location if the magnetic field is active but holds the same position. This makes them ideal for mechanisms that require a fine position control. There are two ways of winding a stepper motor: unipolar and bipolar. In the former, the windings are designed for current to always follow the same direction in the motor coils. The latter needs the motor windings to be energized with current flowing in one or another direction depending of the case. The good news about unipolar design is that the drive circuit can be a simple transistor per winding (typically four windings). Bipolar motors require an H-bridge for each motor coil (typically there are two coils).

Regardless of the motor type of winding, energizing a coil only causes the rotor to align with the magnetic field created. To achieve a rotating action, the different coils have to be activated in a certain sequence. Depending on the order of coil activations, different effects can be achieved, and this gives way to several methods of driving stepper motors.

STEPPER DRIVE MODES

Whether the stepper motor is bipolar or unipolar, there are different energizing patterns (see Figure 2). Depending on the pattern, a motor will exhibit higher torque and more or less steps per revolution. The complexity of the driver circuit may change too. Current sensing is needed only for microstepping, but current-limited circuits are welcome.

The test source code in Listing 1 is for operating a stepper motor using each of the drive methods: wave drive, full step drive, half step drive, and microstepping. Let's review each one.

Wave drive mode is the simplest mode. It involves energizing one winding at a time. A typical unipolar stepper motor3r has four coils. Let's call them A, B, C, D. An energizing sequence A-B-C-D will make the motor turn clockwise. The opposite sequence A-D-C-B will

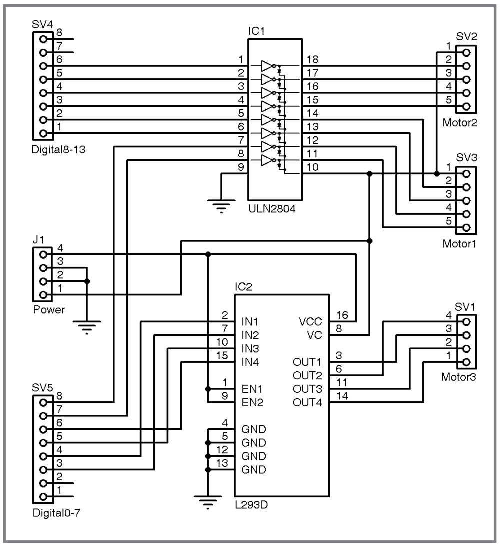


Figure 3—This is the controller circuit schematic to interface Arduino I/O headers with stepper motors.

DesignCon® 2010

The **BRIGHTEST** Minds in Electronic Design

Conference: February 1–4, 2010
Exhibition: February 2–3, 2010
Santa Clara Convention Center
Santa Clara, California

Future milestones start at DesignCon

Be a Chiphead

Kick-off the New Year with DesignCon, the premier annual event for electronic design experts, addressing common issues in signal integrity, power management, interconnection, and design verification.

DesignCon 2010 Keynote Speakers

John Daane
Chairman, President, and Chief Executive Officer
Altera

Alex Shubat
President and Chief Executive Officer
Virage Logic

Jure Sola
Chairman and Chief Executive Officer
Sanmina-SCI

Event highlights:

New! PCB Summit • IP Summit • Tutorials and Technical paper sessions • Technical panels
Business Forum Sessions • 125+ exhibitors • DesignVision Awards
Unmatched networking opportunities • And more!

Visit www.designcon.com for full event details and to register for this must-attend industry event

Enter promotion code Chiphead7

Featured Events IP SUMMIT at DesignCon PCB SUMMIT at DesignCon

ICbank
Gateway to Electronics

MP3P DIY KIT, Do it yourself

(Include Firmware Full source Code, Schematic)

• myPIC

Only
$160 qty 100 $220 qty 1

Powerful feature

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

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

• myWave (MP3 DIY KIT SD card Interface)

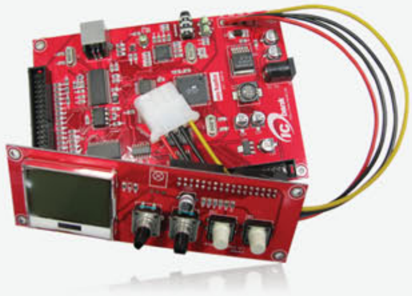
Only
$150 qty 100
$200 qty 1

Powerful feature

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

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

• myAudio (MP3 DIY KIT IDE)



Only
$180 qty 100
$220 qty 1

Powerful feature

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

ICbank
Gateway to Electronics

www.icbank.com

create a counterclockwise motion. For a stepper motor with N steps, N winding activations must occur for the motor to complete a full revolution.

The same can be achieved by energizing two coils at the same time. Full step drive mode improves the motor's holding torque. This is definitely advantageous when you need to get the most torque out of a motor.

The half-step drive mode energizes two windings at a time. When two consecutive coils are energized, the rotor is forced into an intermediate position. For example, when A and B coils are energized at the same time, the rotor is set midway between the A and B locations. (There are N/4 A or B

Listing 1—This is sample C code showing how to program each drive mode.

```
// test code for several drive modes for unipolar stepper motors
// Miguel Sanchez 2008
// -------------------
// output pins 6,9,10 & 11 have been chosen so PWM can be used.
// this code is just a proof of concept for Circuit Cellar article
#define MICROSTEPS  16
int i,j;
byte r[MICROSTEPS];

void setup()
{
  for(i=0;i<MICROSTEPS;i++) {r[i]=sin(i*3.1416/MICROSTEPS)*255.0; Serial.println(r[i],DEC); }
  pinMode( 6,OUTPUT);
  pinMode( 9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  // wave drive (one coil at a time)
  for(j=1;j<100;j++)  {
     digitalWrite(6,HIGH); digitalWrite(9,LOW);  digitalWrite(10,LOW);  digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,HIGH); digitalWrite(10,LOW);  digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,LOW);  digitalWrite(10,HIGH); digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,LOW);  digitalWrite(10,LOW);  digitalWrite(11,HIGH); delay(10);
  }
  delay(1000);

  // full drive (two coils at a time)
  for(j=1;j<100;j++) {
     digitalWrite(6,HIGH); digitalWrite(9,HIGH); digitalWrite(10,LOW);  digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,HIGH); digitalWrite(10,HIGH); digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,LOW);  digitalWrite(10,HIGH); digitalWrite(11,HIGH); delay(10);
     digitalWrite(6,HIGH); digitalWrite(9,LOW);  digitalWrite(10,LOW);  digitalWrite(11,HIGH); delay(10);
  }
  delay(1000);

  // half-step drive (combines the two methods above)
  for(j=1;j<50;j++) {
     digitalWrite(6,HIGH); digitalWrite(9,LOW);  digitalWrite(10,LOW);  digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,HIGH); digitalWrite(9,HIGH); digitalWrite(10,LOW);  digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,HIGH); digitalWrite(10,LOW);  digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,HIGH); digitalWrite(10,HIGH); digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,LOW);  digitalWrite(10,HIGH); digitalWrite(11,LOW);  delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,LOW);  digitalWrite(10,HIGH); digitalWrite(11,HIGH); delay(10);
     digitalWrite(6,LOW);  digitalWrite(9,LOW);  digitalWrite(10,LOW);  digitalWrite(11,HIGH); delay(10);
     digitalWrite(6,HIGH); digitalWrite(9,LOW);  digitalWrite(10,LOW);  digitalWrite(11,HIGH); delay(10);
  }
  delay(1000);

  // microstepping
  // we need PWM here !!!

  for(j=1;j<100;j++) {
     for(i=0;i<MICROSTEPS;i++) { analogWrite(6,r[MICROSTEPS-i-1]);  analogWrite( 9,r[i]); delayMicroseconds(500); }
     for(i=0;i<MICROSTEPS;i++) { analogWrite(9,r[MICROSTEPS-i-1]);  analogWrite(10,r[i]); delayMicroseconds(500); }
     for(i=0;i<MICROSTEPS;i++) { analogWrite(10,r[MICROSTEPS-i-1]); analogWrite(11,r[i]); delayMicroseconds(500); }
     for(i=0;i<MICROSTEPS;i++) { analogWrite(11,r[MICROSTEPS-i-1]); analogWrite( 6,r[i]); delayMicroseconds(500); }
  }
  delay(2000);
}
```

locations in a full revolution.) With this drive mode, the motor turns using half-steps; therefore, a full revolution requires 2N half steps.

The fourth drive possibility is microstepping. While energizing two coils at the same time, different current may be applied to each coil. Imagine two people pulling the two ends of a rope. The rotor position is signaled by the marker in the middle of the rope. By carefully adjusting the current on each coil, you can create a large number of microsteps. With this drive mode, you get much smoother control over the rotor's the rotating angle. The drawback is that it is a more complex technique because it requires software to set coil drive current. Fortunately, there are custom chips in the market that can employ this drive mode by using PWM on the coil drive.

It's common to use eight or 16 microsteps when you're microstepping. This means a full revolution requires 8N or 16N microsteps. The current pattern that works best for microstepping is a combination of sine and cosine functions on adjacent coils.

HOW IT WORKS

Different manufacturers sell stepper motor driver chips. Some of them split the power and logic sections in two chips, while others pack them together as a single chip. I was focused on a smaller design, so I searched for single-chip solutions. Modern chips pack a lot a smart features on the controller, but they also need a bit more PCB real state, they aren't always available in DIP packaging, and they can be more expensive.

I finally settled on a driver chip designed for driving DC motors, but it can be used for driving a bipolar stepper motor too. The L293D is a quad half-H that can be wired as a dual H-bridge so it can power the two coils of a bipolar stepper motor. It can handle up to 600 mA per coil. (The L293 goes up to 1 A.) The good thing about the L293D is that the output clamp diodes are internal, although the current limit is slightly lower than L293. The circuit also includes a thermal shutdown feature for protection in case there is a problem with the load.

Listing 2—This is Arduino code for interpreting Table 1's motor control commands.

```
#include <Stepper.h>

// feel free to adapt the steps/revolution and I/O pins of your motors

Stepper stepper1(200,6,7,8,9);
Stepper stepper2(200,10,11,12,13);
Stepper stepper3(48,2,3,4,5);

  int in,i,motor;
  long n,m1,m2,m3;

int nextchar() {
while(Serial.available()==0) {} // wait till next char is available
return Serial.read();
}
void setup() {
  Serial.begin(9600);
}

void loop() {
  //Serial.print("OK\n");
  in=nextchar(); // first letter of command
  if(in!='H' && in!='S' && in!='?' && in!='F' && in!='B') // bad command
    while(nextchar()!=13); // purge till CR
  else // valid command
  {
  motor = nextchar(); n=0;
  switch((char)in) {
    case 'H':
  if(motor=='1') { stepper1.step(1000); m1=0; }
  if(motor=='2') { stepper2.step(1000); m2=0; }
  if(motor=='3') { stepper3.step(1000); m3=0; }
case 'S':
  for(i=0;i<3;i++) n=(nextchar()-48)+n*10;
  if(motor=='1') stepper1.setSpeed(n);
  if(motor=='2') stepper2.setSpeed(n);
  if(motor=='3') stepper3.setSpeed(n);
  break;
case '?':
  if(motor=='1') n=m1;
  if(motor=='2') n=m2;
  if(motor=='3') n=m3;
  break;
case 'F':
  for(i=0;i<3;i++) n=(nextchar()-48)+n*10;
  if(motor=='1') { m1+=n; stepper1.step(n); }
  if(motor=='2') { m2+=n; stepper2.step(n); }
  if(motor=='3') { m3+=n; stepper3.step(n); }
  break;
case 'B':
  for(i=0;i<3;i++) n=(nextchar()-48)+n*10;
  if(motor=='1') { m1=n; stepper1.step(n); }
  if(motor=='2') { m2=n; stepper2.step(n); }
  if(motor=='3') { m3=n; stepper3.step(n); }
  break;
  }
  if(nextchar()!=13) Serial.println("ERR");
  else Serial.println(n);
  }
}
```

Which drive is the best? It depends. If you need maximum torque, the second mode is the best choice. But the simplest choice is the first drive mode. Listing 1 is test code for the drive modes of a unipolar motor.

SOFTWARE DESIGN

I had to design several things, such as microcontroller and PC code, as well as the communication protocol used for communication. I started by defining the service the controller would provide the PC.

Commands are sent from the PC via the USB port that emulates a serial port. Each command is a single line of text. A line ends with a carriage return code (ASCII 13). Each command is answered with a single line by the controller when it's done. The available commands are listed in Table 1.

Some applications may need to set the steps of two or more motors so they can all move at once. But I've lived without this feature given the peculiar mechanics of my system. A command interpreter runs on the Arduino's microcontroller, so it performs the requested command and replies after the command is completed. If there is an error in the command string, it answers accordingly too. The Arduino programming environment is a simple but effective tool.

Functions have been added to the standard C language, so basic input/output and delay generation don't require additional libraries. Note that the C source code does not have a `main()` function. Instead the program must include two mandatory functions: `setup()` and `loop()`. The former contains all the required initialization code. The latter is constantly called once setup is finished. Thus, there is an implicit endless loop. Syntax is the same as C. More details about the functions are available at www.arduino.cc.

The programming environment has also been enriched with some useful libraries to deal with I^{2}C communication, stepper motors, or EEPROM access. Most of the hardware configuration details are hidden, and many sample programs are included too. However, an advanced user can directly access the processor registers to do it his or her own way. Timers or serial port configuration details are things you do not need to learn to use serial communications in a project.

I knew I'd have the benefit of the stepper motor library, so I didn't have to write my own. Besides the command interpreter for the microcontroller, there's PC-side software that interprets graphic files and it turns them into a sequence of commands for the motor controller. You can write this program in any language that can interface with a serial port. I used Perl because it's easier to use when parsing data files. The PC software also includes the geometry of the physical system, and it performs most of the necessary math. I haven't included a graphical representation of the data yet.

HARDWARE DESIGN

The Arduino board includes plenty of I/O pins, but they aren't suitable for driving parts other than LEDs and piezo buzzers directly. Many projects may need signal adaptation for either the inputs or outputs. A clever idea is to put the extra hardware (i.e., output drivers, input conditioners, etc.) on another PCB (i.e., shield) that you can plug into the Arduino board. This creates a compact design that can easily communicate with a host computer (serial), and it may be powered by the USB bus.

My project requires Arduino to drive three stepper motors: two unipolar and one bipolar. The easier parts are the unipolar motors because each coil can be driven by a single Darlington output. The bipolar motor requires one H-bridge per coil. To keep the cost low, I used a single ULN2804 to drive the unipolar motors while I used an L293 to drive the bipolar motor (see Figure 3). Four digital outputs are used for each stepper motor driver. A total of 12 output pins—digital outputs 2 to 13—are used for driving the three motors.

Some of the available inputs can be use to get the signals from the home detectors on each axis. But I chose a low-tech approach. I turned the motors enough so they reached the home position, and then some steps were skipped once the mechanical limit was reached.

PUTTING IT TOGETHER

Once the hardware was designed, I needed to make it all work together nicely. This meant I had to develop some software. The easiest way to do this was to use the Arduino IDE. Communication was easily handled with Arduino code because the ATmega168's hardware serial port is wrapped with the serial class. For the stepper motor controller, I chose a built-in library to implement the aforementioned full drive mode.

Listing 2 is microcontroller code. It reads from the serial port, which is connected to the USB-to-serial chip, and it interprets the commands detailed in Table 1. An error message is returned if the command syntax isn't correct; otherwise, a numeric value of the command is returned. The system accounts for the location of each motor. The value can be queried with the ? command.

MORE DRIVING

If you need to drive three bipolar stepper motors, you can use the design I described here, but with L293 drivers. If the bipolar drive is always energizing two coils, you can use a couple of inverter gates so only two output pins would be needed per stepper motor. If you want to drive three bipolar motors, you can remove IC1 and repeat the circuit of IC2 three times.

I had a lot of fun playing with different models of Arduino boards. I think I will be using it in the future. It is available (and cheap), easy to interface, and powerful enough for many tasks. My only complaint about Arduino design is that not all headers are located on the same grid. So any standard 0.1" prototype board has some trouble fitting one of the headers. ■

Miguel Sanchez (m1gs4n@gmail.com) holds a B.S., M.S., and PhD in Computer Science. He has been teaching computer networking courses at the Technical University of Valencia (UPV), Spain, since 1989. Miguel's interest in electronics and microprocessors sparked his career in computer science, but his solder is always at hand. His research is currently focused on vehicle energy efficiency. Miguel also works as a consultant.

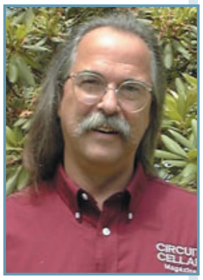
PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/234.

SOURCE

Arduino Decimila
Arduino | www.arduino.cc

by Jeff Bachiochi



Good Vibrations

Wave Shaping and Theremin Design with an MCU

Do you want to build your own version of the Theremin? This project enables you to produce a varying sine tone without physical contact. You can use a microcontroller's output bits to create a DAC with an R-2R ladder network of resistors. Each write to the port produces an immediate output voltage.

So to you I shall put an end. And you'll never hear surf music again. —The Jimi Hendrix Experience, "Third Stone From the Sun," *Are You Experienced?*, Track Records, 1967.

If you're a Jimi Hendrix fan, you're familiar with the lyrics to the song "Third Stone From the Sun." Even Hendrix acknowledged the music of the 1960s as belonging to an American rock band whose popularity for its close vocal harmonies and lyrics reflected a Southern California youth culture of cars and surfing. The Beach Boys are to America what the Beatles are to England. Although the Beatles may have won the hearts of many Americans,

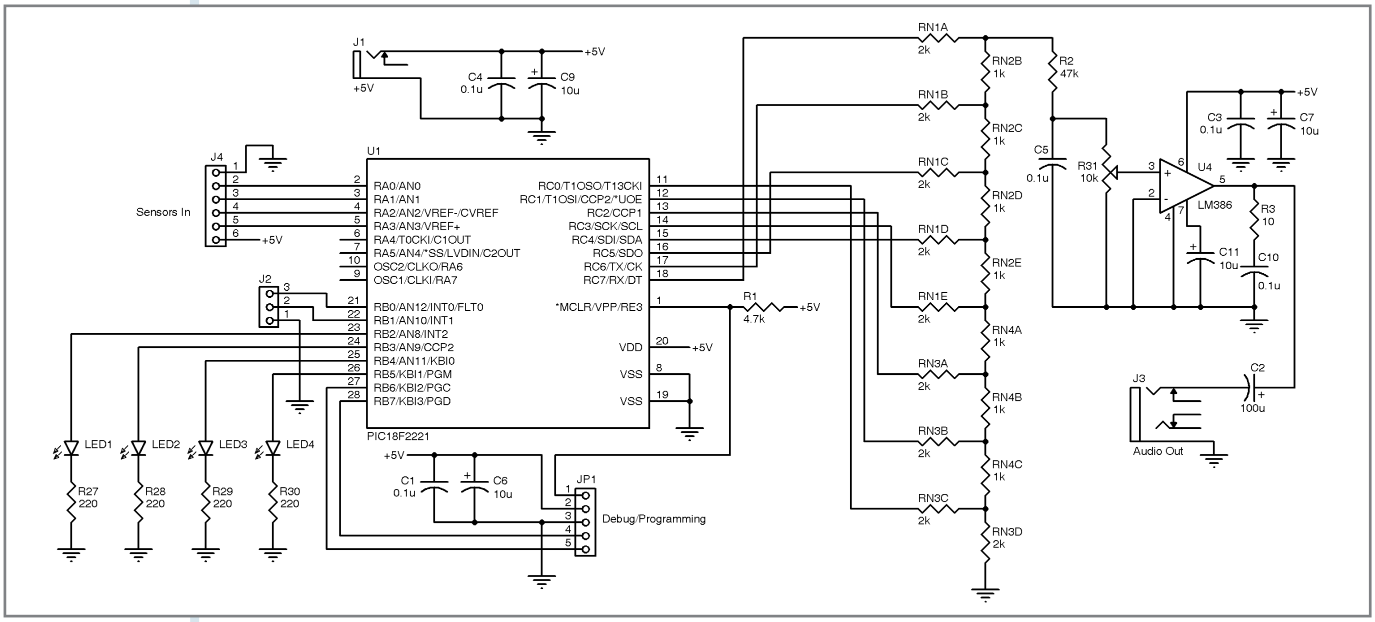


Figure 1—The schematic for this project introduces the use of an R-2R ladder DAC. This allows for very fast changes in analog values, which is a must for producing waveforms in the audio range. The pots originally used as analog inputs at J4 were replaced by infrared proximity sensors.

the Beach Boys remained the creative equals to the Fab Four. Their album *Pet Sounds* broke new ground in album production. In fact, Paul McCartney has stated that it was the inspiration behind the album *Sgt. Pepper's Lonely Hearts Club Band*.

Some critics say the number 1 Beach Boys hit "Good Vibrations" is the best rock single of all time. As it turns out, the "good" vibrations in the single were produced by an electronic musical instrument called an Electro-Theremin. Trombonist Paul Tanner and amateur inventor Bob Whitsell developed it in the late 1950s to mimic the sound of a Theremin—an electronic musical instrument a musician controls without touching it. It's named after its Russian inventor Leon Theremin (1896–1993). Confused? Well, hold on.

One of my favorite movies is *The Day the Earth Stood Still* (20th Century Fox, 1951). Even the 2008 remake—which I saw in a theater—was good. Unlike most remakes, it lived up to the message of the original. On a recent Sunday evening, I ran across the DVD, which I had received as a birthday gift. With nothing else to do, I watched the black-and-white classic, as well as the special features on the DVD. As it turns out, the movie's composer Bernard Herrmann used two Theremin electronic instruments to create the eerie music from beyond our world. (Unfortunately, this sound has become synonymous with B movies.) Right then, I knew I had to dedicate a "From the Bench" (FTB) article to the process of building my own version of the Theremin. I call it the "FTB-Theremin."

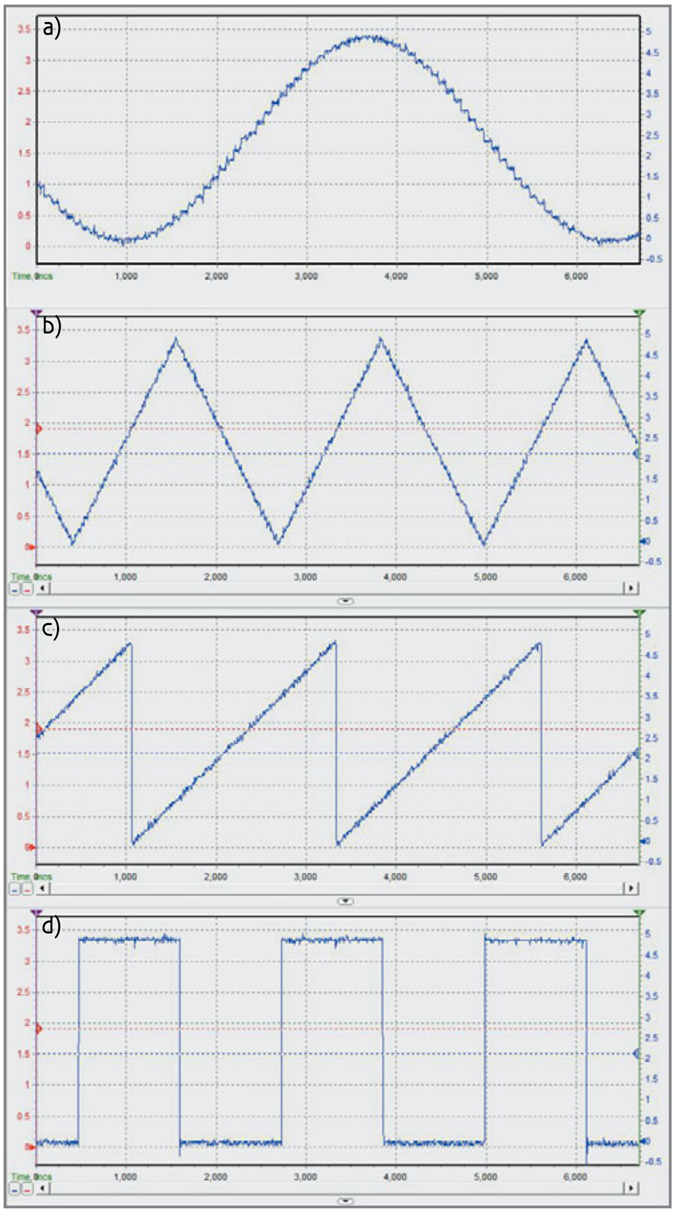


Photo 1—The microcontroller's 256-byte internal EEPROM has room for four 64-byte data tables. Each table holds the output values necessary for producing a particular wave shape at the output of the R-2R DAC. Here are the DAC outputs for: (a) a sine wave, (b) a triangle wave, (c) a saw wave, and (d) a square wave.

THEREMIN TECHNOLOGY

The Theremin is the only musical instrument I know of that's played without physical contact. You stand in front of the instrument and move your hands in the proximity of two metal antennas. The distance from one antenna determines frequency (pitch), and the distance from the other controls amplitude (volume).

The instrument's pitch circuitry includes two radio-frequency oscillators. One oscillator operates at a fixed frequency. Your distance from the pitch control antenna dictates the other's frequency. Your hand acts as the grounded plate (your body being the connection to ground) of a variable capacitor in an inductance-capacitance (LC) circuit. The difference between the frequencies of the two oscillators creates a difference tone in the audio frequency range.

To control the instrument's volume, your other hand acts as the grounded plate of another variable capacitor. In this case, the capacitor affects the amplifier circuit. The distance between your hand and the volume control antenna determines the capacitor's value and the Theremin's volume. You can connect the instrument's audio output to an external amplifier and speaker setup.

ELECTRO-THEREMIN

In the early 1950s, Robert Moog began manufacturing and selling vacuum tube Theremin kits. By 1961, he had marketed his first transistorized Theremin kit. His interest in the design and construction of complex electronic music systems enabled him to establish a small synthesizer company named R. A. Moog Co., which later became Moog Music. In the early days of electronic music, there weren't piano-type keyboards. Experimentation consisted of using knob-controlled oscillator modules with filter banks and envelope generators all wired together with patch cords.

The Electro-Theremin used the new transistorized audio oscillators in place of RF oscillators, which were used in the original Theremin. Whitsell used a mechanical means to vary the audio oscillator directly. The instrument featured a tone and portamento (slide between pitches) similar to that of the original, but with

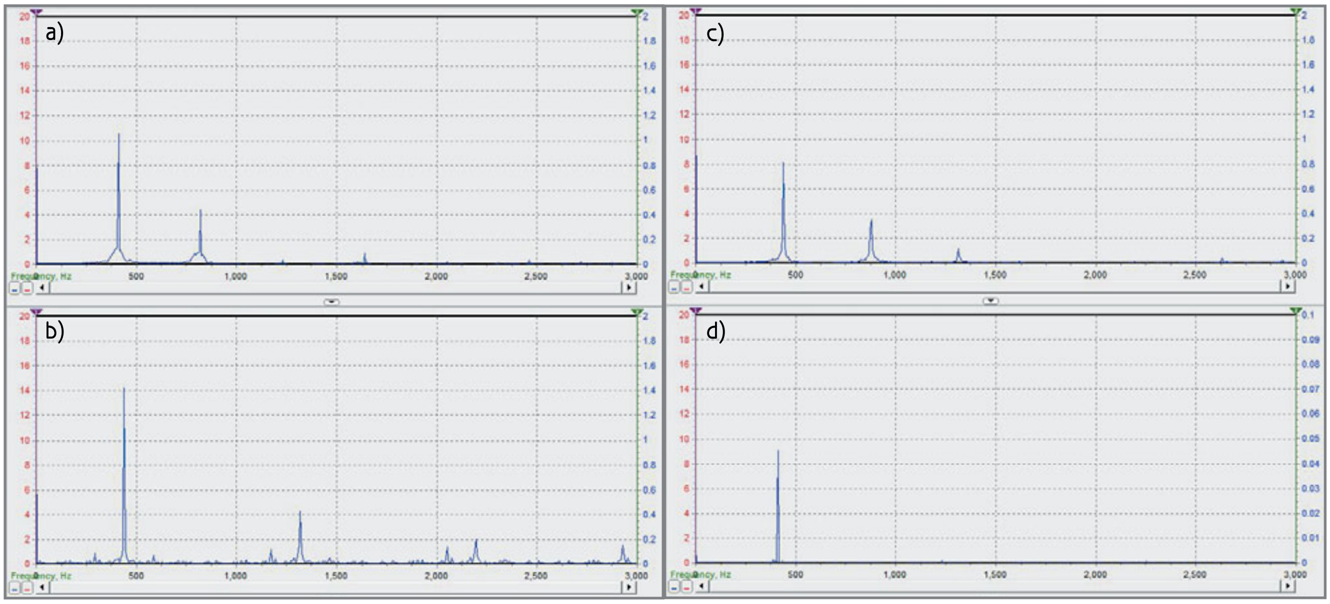


Photo 2—While a true sine wave should have only the fundamental frequency, a triangle wave is made up of the fundamental and all odd harmonics (i.e., the third, fifth, etc.). The sawtooth (ramp) wave is a combination of the fundamental and all harmonics (i.e., second, third, fourth, etc.). This figure shows the frequency components of (a) the sine wave output, (b) the triangle wave output, and (c) the ramp wave output. The last graph (d) shows the frequency components of the sine wave after a low-pass filter.

a hands-on control. This was analogous to the slide of the trombone, which was Tanner's instrument of choice. A knob was used to adjust the volume, which contrasted with the hand movements in space that were the original Theremin's signature feature.

FTB-THEREMIN

I'm not a musician, although I certainly appreciate the

Ultra Small Panel PC

PPC-E4

- Fanless ARM9 200MHz CPU
- 3 Serial Ports & SPI
- Open Frame Design
- 2 USB 2.0 Host Ports
- 10/100 BaseT Ethernet
- Audio Beeper
- Micro SD Flash Card Interface
- Battery Backed Real Time Clock
- 64 MB Flash & 64 MB RAM
- Linux with Eclipse IDE or WinCE 6.0
- JTAG for Debuging with Real-Time Trace
- WQVGA (480 x 272) Resolution TFT LCD with Touch Screen
- Four 12-Bit A/Ds, Two 16-Bit & One 32-Bit Timer/Counters

Microsoft Windows CE

2.6 KERNEL

The PPC-E4, an ultra compact Panel PC with a 4.3 inch WQVGA (480 x 272) TFT color LCD and a resistive touch screen. The dimensions of the PPC-E4 are 4.8" by 3.0", about the same dimensions as that of popular touch cell phones. The PPC-E4 is small enough to fit in a 2U rack enclosure. **Price is $345 at quantity 1.**

For more info visit: www.emacinc.com/panel_pc/ppc_e4.htm

Since 1985
OVER
24
YEARS OF
SINGLE BOARD
SOLUTIONS

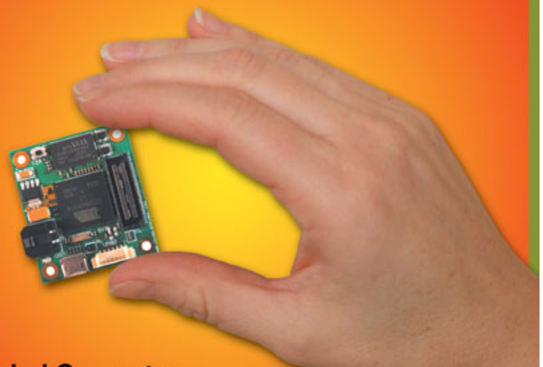
EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

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

CALAO SYSTEMS

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

endless hours of practice a musician devotes to create the music that so many of us enjoy today. I played the saxophone in the school band, but I never practiced much. I also played—I use the term *played* loosely—guitar in a garage band. However, other interests always took precedence. Fortunately, I can now use my engineering skills to approach music-making from a new angle.

To keep in harmony with the original Theremin's features, my project produces a varying sine tone without physical contact. The design uses a microcontroller's output bits to create an output DAC using an R-2R ladder network of resistors. With the R-2R ladder, each write to the port produces an immediate output voltage. The faster this voltage can change, the higher the output frequency. If an output bit is simply toggled, the square wave frequency can be quite high. In fact, it easily can be out of range of your ears. However, to produce a sine wave, one cycle needs to consist of many output levels. An infinite number of output bit values is necessary to approach a true sine wave. Without an infinite number of available output bits, I need to give up something to approach my goal. (I will come back to this in a bit.)

DAC

Producing a square wave involves alternating an output bit between 0 and 5 V. Producing a sine wave requires an output of 0 and 5 V and all the levels in between. To do this, I can use multiple digital outputs and a voltage divider. Two resistor values—R and 2R—make up the voltage divider. Refer to Figure 1. All the outputs supply either 0 or 5 V to a particular leg of the R-2R voltage divider. The divider output is a sum of all voltages produced at each node. For D7 (MSB, N = 1), this node contributes $1/2^N$ ($1/2^1$), or 0.5 of 5 V. For D6 (N = 2), it is $1/2^2$, or 0.25 of 5 V—and so on. With eight outputs, D0, the LSB (N = 8) can add $1/2^8$, 1/256 of 5 V. With all the outputs at 0 V, the sum of all the nodes is 0 V. With all the outputs at 5 V, the sum of all nodes is 4.98046875 V (i.e., 2.5 + 1.25 + 0.625 + 0.3125 + 0.15625 + 0.078125 + 0.0390625 + 0.01953125).

The number of bits for the DAC and the number of divisions for a single cycle have a common-sense relationship. For instance, in this example, the 8-bit DAC has the ability to divide the amplitude of the waveform into 256 different levels. With an optimum change of 1 bit, you might shoot for 510 divisions per cycle—255 steps increasing and 255 steps decreasing. This would give a triangle-shaped waveform. While this is optimum for a triangle wave, 510 divisions is overkill for a square wave (which needs only two) and under-kill for a sine wave (which requires an infinite number).

I began thinking about external EEPROM for data, but many micros have internal EEPROMs that can eliminate the requirement to spend valuable cycles clocking data into the microcontroller. I chose a Microchip Technology PIC18F2221 for the project because it has an

Circuit Cellar magazine Digital Plus subscribers are among the first to receive notification about limited-supply product samples as part of our many design contest and product sampling programs. Thousands of these subscribers have received an impressive array of evaluation kits as a result.

By purchasing the Digital Plus edition, not only will you have access to your monthly copy of *Circuit Cellar* magazine before print subscribers, but with the same speed-of-light transmission, you'll also receive special-access offers delivered via e-mail only to Digital Plus subscribers.

Ensure your "A" list status for these special offers. Sign up for your Digital Plus subscription today!

internal oscillator that can run at 32 MHz, it has ADC inputs, and it has 256 bytes of EEPROM. Plus, it costs less than $4.

The first part of this design required me to create a table for the values that would create a sine waveform. I used my PC and a basic program to calculate the 8-bit integer values to put into a 256-byte table in EEPROM.

```
10 for x = 0 to 256
20 degree = $\frac{x}{256} \times 360$
30 y = int$\left\{\left[\sin\left(\frac{\text{deg ree}}{57.29577951}\right) \times 128\right] + 128\right\}$
40 print x,y
50 next x
```

The PIC18F2221 has a two-level interrupt system that enables the waveform code to have the highest priority and interrupt low-priority routines. For now I just want to see how fast I can output these 256-byte values in a loop. With the PLL enabled, the oscillator runs at 32 MHz. Therefore, the execution speed is 125 ns per instruction ($F_{OSC}/4$). If I can keep the loop to about 20 instructions, I can output 256 values, one cycle's worth of data, in 640 µs (i.e., 20 × 256 × 125 ns). Thus, the highest frequency is approximately 1,500 Hz (i.e., 1/640 µs).

THE A440 STANDARD

You have to start somewhere. A440 is the standard for musical pitch. As you know, A is one of the notes in the scale A, B, C, D, E, F, G and back to A. Or maybe you're more familiar with this scale: Do-Re-Me-Fa-Sol-La-Ti-Do? In 1936 the American Standards Association recommended that A, above middle C, should be tuned to 440 Hz. Today this is known as ISO 16. And you thought the electronics industry had a lock on standards!

Almost all scales use the octave as a doubling of frequency, as in A440 and A880. If the same note—A, for instance—is an octave higher (or lower), it is twice (or half) the frequency of the original. The Western scale breaks the octave down into 12 semitones. This includes A–G (the white keys on a piano) along with sharps and flats (the black keys), which are also real notes. While the A440 and A880 seem straightforward, the actual frequencies of the notes in between are not so "integery" (if you will). The actual frequencies can be slightly different depending on whether *equal temperament* or *just intonation* is used to determine these values. I don't want to stray too far from the path here so you can do a search on these terms to learn more.

For this project, I was hoping for something like plus and minus an octave around A440. My initial tests showed that the highest frequency I might produce could be almost two octaves above A440, which I think is fine. If the loop used to output a waveform's values takes 100% of the execution cycles, no other application work can be done. So a timer must be used to allow some number of non-high-priority interrupt routine cycles to execute before going back to the waveform loop. The initial test used about 10 execution cycles within the waveform loop, I used a count of 10 for the time. This is where the 20 cycles comes from in the initial calculation. The timer reload value (10), along with the number of execution cycles for the waveform routine (10), sets the maximum frequency of the waveform. To change the frequency, I need to increase the timer reload value. This is the point where the thought process moves to a user interface.

CONTROL INPUTS

The original Theremin had two metal antennas that were used to vary the pitch and volume of the sound. I wanted to emulate this feature. I needed to choose the actual sensor technology, and I knew it would have to interface with the microcontroller.

One of the most prolific peripherals in many microcontrollers is the ADC. I decided to design in the use of four analog channels. Each analog input should cover a range of 0 to 5 V. This voltage will be converted to a 10-bit number between 0 and 1,023.

The ADC is initialized to operate in the background (converting an input voltage to a 10-bit value) and signaling when done using (in this case) a low-priority interrupt. The low-priority conversion-complete routine will be responsible for storing the conversion value into one of four variables. Each variable holds the latest conversion of each input channel. After storing the value properly, the channel is incremented, another conversion is initiated, and the interrupt exits. It is possible that the high-priority interrupt could grab control before both bytes of the 10-bit conversion are stored. If this should happen, the value for this particular input would not be valid. To prevent this, all interrupts are disabled during the two instructions, where the conversion value is actually transferred to its associated variable.

For testing purposes, I used four potentiometers to provide analog voltages to the four input channels. Channel 1 is used to change the pitch of the waveform. Increasing the timer's count increases the time between output values placed on the R-2R ladder from the EEPROM, and thus lowers the frequency. So, I wanted a high voltage in to convert to a low number and a low voltage in to convert to a high number. This is opposite of the way the ADC works, so I complement the 10-bit ADC result to get this. I only need to add a few instructions in the high-priority routine to reload the 10-bit conversion value into the timer to change the frequency. If a (minimum) timer value of 10 + 0 (A/D conversion value) gives 1,500 Hz, a timer value of 10 + 1,023 (A/D conversion value) will give approximately 33 ms (1,043 × 256 × 125 ns). Therefore, the lowest frequency would be 1/33 ms, or approximately 30 Hz.

The timer resolution (125 ns based on the 32-MHz clock) is the key element in reproducing Theremin quality. A Theremin has an infinite amount of pitch adjustment, so it slides (glissando) between tones. If the timer resolution is too coarse, there will be a perceptible jump in frequency change instead of a slide. At the upper frequency, a 1-bit

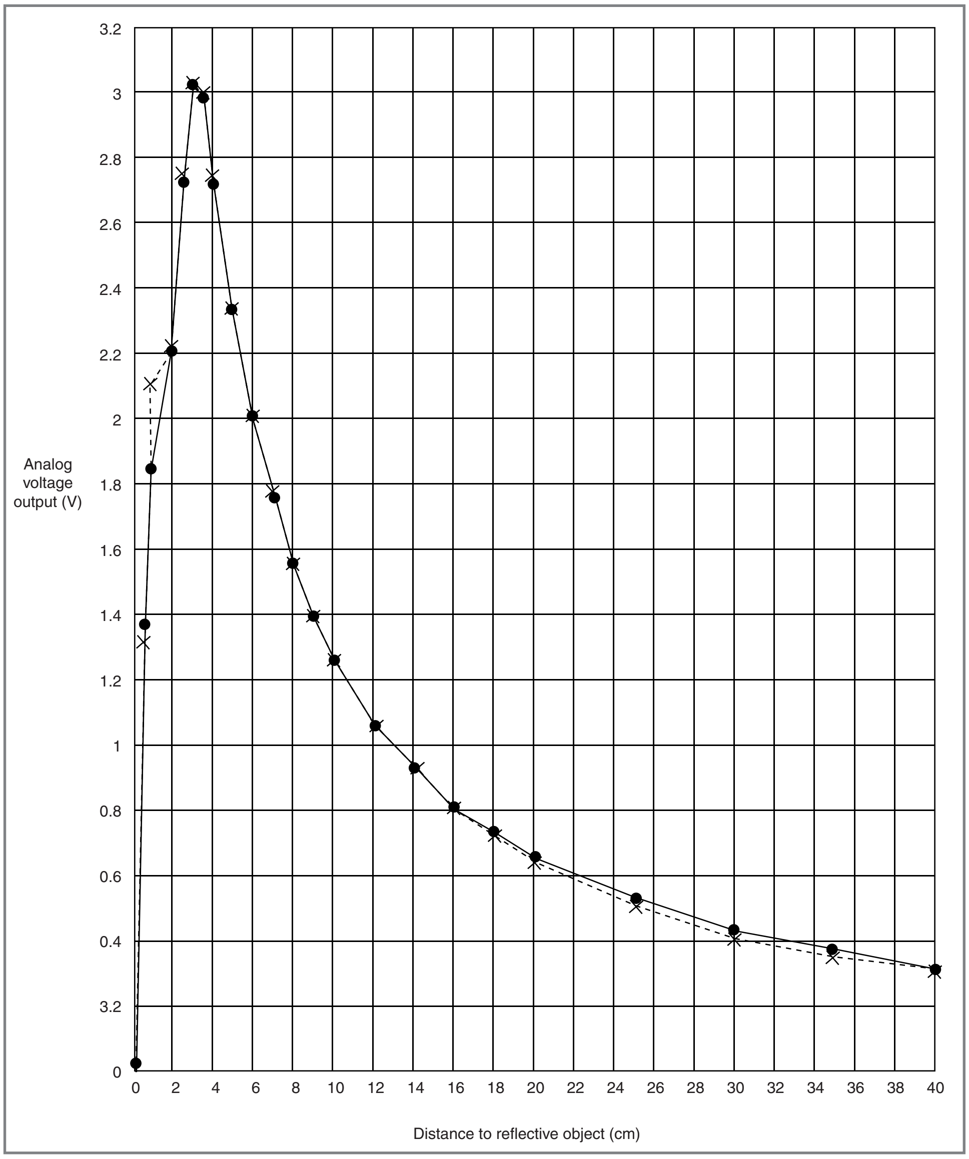


Figure 2—This Sharp IR distance sensor outputs an analog voltage based on its reception of reflected IR. Its nonlinear output is cause for some frustration.

change in the reload count will lower the frequency from 1,500 to 1,488 Hz (approximately 12 Hz). At 440 Hz, a 1-bit change in the reload count will lower the frequency from 440.14 to 434.02 Hz (approximately 6 Hz). While this change is imperceptible at the low end, it can be heard at the upper end. One might consider this a defect, whereas marketing would surely put a positive spin on this, ahem, *feature*.

VOLUME

This function turns out to be one of the easiest to implement. The A/D routine responsible for storing the input value for this function uses only the most significant bits. The new value is stored into a single byte. This is done to allow the hardware multiply routine (byte times byte) to apply this (volume) value to the present EEPROM (waveform) value. The product's high byte is used as the output value to the R-2R ladder. The multiply scales the waveform data. Large (volume) values scale down the waveform values slightly, while small (volume) values reduce it a great deal. The wave shape remains the same (relationship between values), while the amplitude gets reduced in proportion to the volume control input value.

WAVE SHAPE

The Theremin is noted for its pure sine wave. This project mimics the sine wave shape by retrieving a predetermined set of values in a periodic fashion. By altering this set of values, you can produce just about any wave shape you want. To better demonstrate this, I decided to reduce the sample size of a cycle from 256 bytes down to 64 bytes. This further reduces the resolution of the wave, but I can now store four different waveforms in the EEPROM.

I chose to predefine four different sets of data: sine, triangle, saw, and square wave shapes. This required only minor changes to the high-priority interrupt routine. Instead of cycling through 256 values, the routine used the least significant 6 bits of the EEPROM address for the waveform. The most significant 2 bits would select which waveform data set was being retrieved. These would be set by the most significant 2 bits of a 10-bit analog input. This essentially allows an input voltage to be divided into four areas, like using a 2-bit ADC.

You can see scope shots of these wave outputs in Photo 1. In each wave, you can see the individual R-2R ladder outputs. As the number of samples for each cycle is reduced, you not only reduce the smoothness of the waveform, but also begin to introduce aliasing that can become an artifact within the audio range. Even though at 64 times the fundamental frequency this puts aliasing out of my perceptible audio range, I can hear it when synthesizing the lower frequencies. This can be eliminated by adding a low-pass filter after the R-2R ladder.

Check out Photo 2. This set of figures shows the makeup of the waveforms as seen by the spectrum analyzer portion of my DiSco oscilloscope. A pure sine wave should have a single frequency component, while triangle and square waves contain odd harmonics, and a sawtooth wave includes all harmonics. Note the difference in frequency content of the sine wave outputs between Photo 2a (the raw R-2R output) and Photo 2d (after a low-pass filter).

SENSORS

From a design point of view, all controls must follow 0- to 5-V input criteria. By rights this would require all input devices to meet the design requirements. In the case of rotary potentiometers, there no problems meeting this requirement—and these are inexpensive to boot! However, this doesn't solve the no-contact requirement that the original Theremin had going for it. The most obvious approaches to sensing proximity (of the hand) involve using ultrasonic or infrared technologies. Both have appeal, but I had infrared sensors leftover from a previous column.

The Sharp Electronics 2D120X is a 5-V infrared distance sensor with a 4- to 40-cm range and an analog output. Figure 2 shows the output profile of distance versus voltage. With my hardware hat on I thought, "Design a variable gain and offset amplifier to follow this giving a 0- to 5-V output." But, with my programmer hat on I thought, "I can do an easy shift in data to get it into the appropriate range." While I might regret this "do

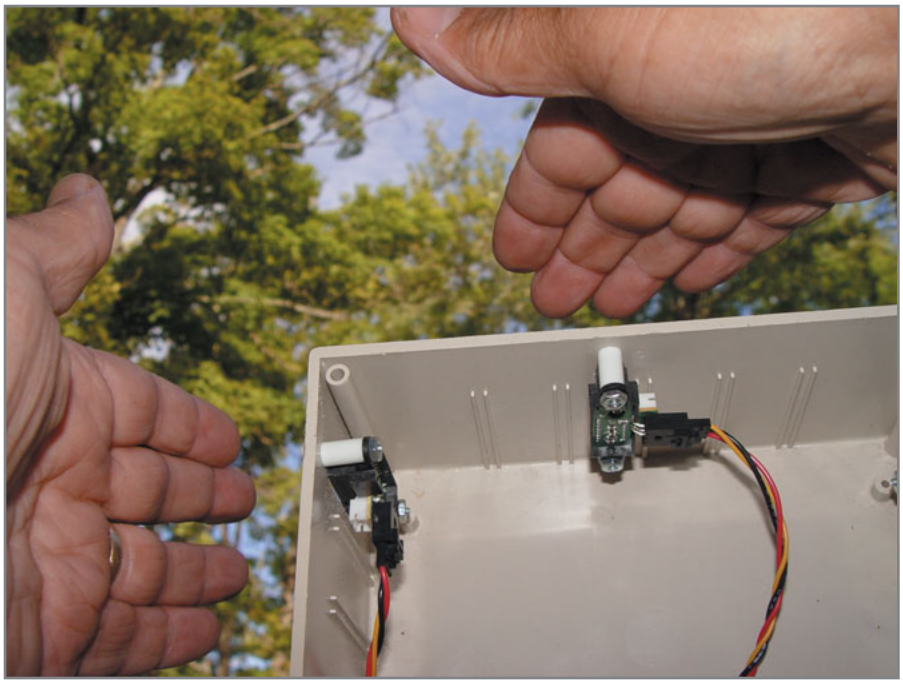


Photo 3—Here I am conversing with some birds on my front deck. With the enclosure's cover removed, you can see the IR sensors mounted to the inside. I'm sure this sounded more like "blah-blah-blah" to the birds. Where's that mockingbird when you need it?

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

Programmers
JTAG / SBW or BSL
USB
PC
One PC and 8 programmers
USB-FPA-1
USB-FPA-2
USB-FPA-3
USB-FPA-8
MSP430Fxx
MSP430Fxx
MSP430Fxx
MSP430Fxx

Elprotronic Incorporated
www.elprotronic.com

THE ORIGINAL SINCE 1994
PCB-POOL®
Beta LAYOUT

Servicing your complete PCB prototype needs :

* **Low Cost - High Quality** PCB Prototypes
* **Easy online Ordering**
* **Full DRC included**
* **Lead-times from 24 hrs** NEW !
* **Optional Chemical Tin finish** no extra cost NEW !

Watch "ur" PCB®
Follow the production of your PCB in **REALTIME**

email : sales@pcb-pool.com
Toll Free USA : 1 877 390 8541
www.pcb-pool.com

Beta LAYOUT

it in software decision" later, I chose to take the lower cost route.

You might notice in Figure 2 that the voltage output doesn't seem to fall below approximately 0.3 V. I'll adjust the converted value from this sensor by subtracting a constant equal to this 0.3-V value. Next, since the output will never reach 5 V, I'll shift the value left one bit, thereby multiplying it by two. Now to put this all in a box!

SAVE THE TREES?

Wouldn't it be nice to make a real piece of wooden furniture? It seems the tradition is to put the guts of a Theremin in a highly polished wooden case. I don't have room for a nice woodworking shop in my basement, so plastic had to do.

Although I now concede that the waveform selection should be a physical pot, I initially mounted three IR distance sensors to three sides of an enclosure. Using these I can control the wave shape, the frequency, and the volume with just hand gestures. In Photo 3, I'm interacting with the frequency and volume sensors mounted inside the enclosure. If only I had three hands! Guess I'll add the pot to set the wave shape manually.

You may have noticed the logarithmic response in Figure 2. This means that the sensor does not produce a linear voltage with respect to distance. You can get good response using the lower half of the range; but once you get in close, the pitch rises pretty rapidly along with an audible step in frequency shift. I'd like to go back and experiment with ways of converting the logarithmic response to a linear one. Another thought is to use an external VREF, which might be 2 or 3 V for the ADC, thereby avoiding a rotate (a times two). An artifact of rotating left reduces the 10 bits value to a shifted 9 bits (the LSB is stuck at zero). This effectively doubles the timer increments creating a larger jump between adjacent frequencies.

I might take a second look at other sensors. There might be something available that would have a better linear performance. However, these will most likely cost more to implement.

I bet some of you may already have ideas about this project. As always, feel free to send me your comments and suggestions. I look forward to your input. ■

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.*

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/234.

RESOURCES

R. George, "Super Mario" Theme Played on a Theremin, www.youtube.com/watch?v=xcTPRjiCs6s.

P. Tanner, Electro-Theremin sample, http://upload.wikimedia.org/wikipedia/en/d/d3/Beach_Boys_-_Good_Vibrations.ogg.

SOURCES

PIC18F2221 Microcontroller
Microchip Technology, Inc. | www.microchip.com

GP2D120 Optoelectronic distance-measuring sensor
Sharp Electronics Corp. | www.sharpsma.com

Touch Panel PC

CUPC-P80 $699 / Qty. 1

Windows XP based Touch Controller
- Touch panel & Touch controller
- AMD Geode LX800 (500MHz)
- 512M DDR RAM
- 4G FLASH HDD
- USB, ETHERNET, RS232, RS485 port
- SD card support
- Windows XP embedded, Windows 2000 / XP, Linux support

From $299 / Qty. 1

CUWIN3100
- ARM9 32bit 266MHz
- WinCE 5.0 Core
- RTC
- SD CARD support
- RS232 x 2 ports
- USB port
- Speaker & Stereo Jack

COMFILE TECHNOLOGY
Toll-Free: 1.888.928.2562
http://www.cubloc.com

by George Martin

Debugging Techniques

During the course of your engineering career, you'll likely work on projects of all sorts: big systems, little systems, real-time systems, and more. As you'll see, there's not one sure-fire debugging technique that you can apply to every project. Your best bet is to have a good grounding in a few techniques and then approach each project on a case-by-case basis.

In my article titled "Passing Parameters" (*Circuit Cellar* 232, 2009), I covered the topic of partitioning a design and presented different techniques to control the amount of resources (typically CPU time) you're using in any one routine. Well, the idea of partitioning a design leads us right into the next interesting topic—debugging.

Why debugging? Or, perhaps stated more elegantly, why do you have to debug? In a perfect world, you don't. Your design and coding is done correctly and your program just runs. Your customer never changes its mind and regulatory authorities never ask you to document and control your design process. But that's so far from the real world I hate to waste the ink even stating the obvious.

A related topic is testing, and the same reasoning applies. For our purposes, I'm going to combine testing and debugging into one category. Check out what Wikipedia says about unit testing. If you think of debugging and testing as the same task, you will become more familiar with the intimate details of your design, code, compiler, and the hardware. When you are designing, you should think about how you are going to test your code. That planning will create a much more solid and stable design. As you design, think about partitioning that design into modules and routines. Your modules, routines, and code will all start to become clear and well defined.

Circuit Cellar readers are faced with designing large systems, small systems, real-time systems, and systems for work and home use. That range of systems makes it unrealistic to come up with a set or rules or a simple checklist to solve every situation. But I can certainly describe where to start and give you some examples for the various types of systems.

TOOLS

A critical component (perhaps the most critical component) of your design and testing is an integrated development environment (IDE). A very powerful feature of the IDE and compiler is the ability to select different types of warnings for the compiler. This is usually available via a pull-down list.

Consider the statements `if (a=b)` and `if (a == b)`. Both are correct, but they are vastly different operations. The former copies the value of variable `b` into variable `a`, and then if that value is nonzero (TRUE) it executes the next statement. The latter compares variable `a` to variable `b`. If they are equal, then the next statement is executed. It is a common mistake to leave out the double equal sign.

You can set the compiler to check for this type of error and issue a warning. I strongly suggest that you turn on all the warnings available on your compiler. Sometimes that slows down the compiler. And if that's a drag, turn all warnings on once a week and before every release or buy a faster computer and leave them on. Next, when a warning is issued, take time to understand the warning and *fix it*! Don't ignore warnings.

With a good IDE you can code, compile, load, set breakpoints, and single-step through your code. All this is functioning under one program. Even the smallest CPU (a Texas Instruments TMS430 in my case) lets you set break points. In these smaller devices, there is a limit on the number of available breakpoints, but it's far better to have just a few than none at all. Also, with a modern IDE, you can inspect C variables (including arrays and data structures). Also note that memory can be inspected and changed. In the more highly integrated CPUs (e.g., a Renesas Technology MC16C/28), you can inspect internal device registers and I/O ports. Again, this is critical to getting a design up and running.

A C compiler with an IDE for modern devices ranges from free (as in free beer) to low cost ($100 to $500) to expensive ($2,500 and up). This is a cost that you need to include in your project's budget. If the expensive IDE is too much, don't try to test and debug with a lesser tool. It's just not worth it. Perhaps you might give it a try for a home project, but certainly not when your job is on the line. Demand the best tools and use them to their fullest.

For real-time embedded systems, a good digital storage scope is also invaluable in the testing and debugging phase of a project. With it, you can capture events that occur infrequently. If you're doing embedded work on both large and small projects, you need good tools. Voltmeters, power supplies, and the like are affordable enough that they are probably not an issue to purchase. But when the topic of a digital scope comes up, right away cost becomes an issue. Look into renting and auctions. I have done both and added the cost of the equipment (or at least a portions of the cost) into the cost of the project. Once you have one, you'll wonder how you ever lived without it.

UNIT TESTING

Unit testing is very much like the testing a hardware designer performs on a schematic. Conditions are set up and signals applied. Internal portions of the design are probed to be sure nothing internally is tending to a limit. Well, when it comes to software, the very same situation applies. What you are trying to do is to confirm that the smallest testable piece of code is functioning as expected. The other issue to keep in mind for software is that all paths through the code need to be tested.

Unit testing is the starting point for the debugging operation. With your IDE and without production hardware, you can start testing your code. I do this all the time and have written about running the C code on a PC. Sure you don't have the real hardware, and it's a different compiler, but significant testing can be completed.

Consider the routines that you've just designed and coded. Routines like a square root or testing for printable characters or conversion from ASCII hex to binary can all be tested on a PC. The simpler routines can be tested by sending all possible values as inputs. So, for the routine that converts from ASCII hex to binary, I would first send all the hex values 0…9, A…F, and check the results. Next I would send a…f and see what happens. And finally, I would send all the ASCII characters, even the nonprintable ones like carriage return and make sure the code handles these properly. For a square root routine, I might generate random numbers, have my routine take the square root, and then square the result and make sure it's close to the original number. Of course, you will need to worry about rounding. As you design your routines, think about how you will test them.

Routines that are not easily tested by sending all values (as the ASCII hex to binary), you will need to get creative with you test plan. On a PC, I have used the keyboard to input characters as if they came from the keyboard on the actual hardware. I also simulated the real display by using the PC display. I could also make up text files with a series of input values and feed that file to a PC running the C code. Next time, I'll show how that's done as we write our first program.

Another key point to keep in mind is that when a system problem is detected or major changes are made to a design, you should go back to your unit testing for the routines involved and confirm that they are still functioning as designed. Think of unit testing as a way to minimize the debug effort.

LITTLE SYSTEMS

With little systems, your resources are limited just by the nature of the design. The microcontroller costs less that $1 and you won't have many (if any) spare pins free to help with debugging. Also, any test or debugging code that you insert will just change the timing of your routines and that typically changes the problem you are addressing. But even one pin can shed light on the problem. Again, when you're deciding on a design approach, include a test approach as part of your plan. In the planning phase of your project, reserve unused system resources for test (CPU pins). That spare pin might help debug.

On all my designs I now look for four pins to be used as test points. These are software controlled and can be set high, low, or toggled by software. Why four? Probably because I have a four-channel digital storage scope. If the hardware design doesn't have four unused pins, then settle for what you can get but get some.

On the smaller systems you'll probably have limited resources. You can use spare pins in several ways. In a real-time interrupt, set the pin high at the start and low at the exit. On the scope, you can see how frequently your interrupt runs and how long the code takes to execute. Is the frequency of execution what you designed? Is executing time the same each pass? Does it ever take much—shorter or longer—to get through the code? Do you know why? These unexpected observations become valuable clues to how your design is functioning. You can also time any routine in this manner. That timing can be used as an overall report to the "real-time" operation of your design.

Perhaps one routine reads data and passes that data to another routine for processing. Well, the first routine could set a test point and the second routine could reset that same test point. Is the test point high for the same amount of time each pass? Again, valuable information in helping you track down a problem. Remember last time I showed you a design for a zero-crossing detector for a full wave rectified input signal

(*Circuit Cellar* 232)? Well, I used a test point to measure how the executing of the design matched the input signal. I could set a test point at a point the code and look on the scope to see if the code was following the input as designed. I'm now getting into that design and prototype hardware is available. I'm using the test point to measure how long the code is spending in each part of the routine. I put lines of code that toggle the test point as they are encountered. This gives me an idea of how the code is progressing as it is tracking the full wave input signal.

REAL-TIME SYSTEMS

I was designing some digital filters on a MSP430. The output was a digital value reporting which filters were active and thus which frequencies were present. I was able to change CPUs to one with a DAC. It was understood this would not be the CPU for production. I used the DAC to output internal variables in the digital filter. These variables reported on how the filter was functioning and how much of the frequency of interest was present. I could then sweep the filter with input signals of different magnitude and frequency and see what the filters were doing. This saved weeks as we needed to tune the filter constants to meet our specific requirements. Felt a lot like unit testing!

In another example, we were having a problem in a motion system. As I write this I forget the details of the problem. But it was impossible to put in a break point to see what was going on. We would destroy the drive components if we stopped the software. We created a structure to hold copies of all the key variables and then an array of that structure to save successive results. As we passed through the code and suspected a problem condition, we took a snapshot of the key variables. We saved copies of those variables in RAM. It was very brute force to begin with, but as we learned more about the problem we could place the snap shot function in more and more informative places in the code. After the system was safely at a stop, we could then inspect the variables. We sent them out a serial port to a PC for printing. It was sort of like a poor man's trace unit.

Today I'm working on two different projects that require very controlled operation of high power output devices. One drives an incandescent lamp and the other drives an LED. On both projects I put in a jumper on the controlling output pin. With the jumper off, I can look at the driving signal on a scope and see if it's what the design calls for. I can also put in break points without worry of damaging the output devices. With the jumper in the hardware is controlled. Again, a good digital scope lets me look at the exact waveforms and builds confidence that the code is correct.

LARGER SYSTEMS

Larger systems have more resources but also have larger more complicated designs and thus larger more complicated problems. All of the techniques previously mentioned still apply. As a reader of *Circuit Cellar*, your systems are probably not of the inventory or payroll nature. But design techniques found in those type systems are creeping into our large projects.

Once I was working on a motion system using an Intel 80C196 microcontroller (just a simple 16-bit device). The IAR C compiler had floating-point and trigonometric functions. The project was pack writing disk drives on the factory floor. We needed to convert from track number on a hard disk to a step size for the controller. This conversion was one big trigonometric function. For the testing, I ran the same C code on a PC with an Intel 8086 CPU (Turbo-C compiler and probably Math Chip 8087 support). The results for many "track number to controller count" conversions were printed. I then ran the same C code routines and test function on the 80C196 and sent the results out a serial port to a PC screen. After comparing the results of both computers/compilers, we began to have confidence that the trig routines in the 196-IAR combination matched the PC/Turbo-C results. That process became our unit testing for that module.

We had a good environment for this test and everyone could look at the results. No one ever questioned if the math was correct in the unit. The smallest resolution of the step size was 0.6 millionths of an inch. It's really difficult to see that amount of movement with the naked eye. (I'm wasting ink again.) A point to remember: a good testing plan with good tests and clear results will keep fingers from pointing your way when problems start to appear. Also, with a good test plan, you can put more eyes on finding the problems.

In addition, when testing and debugging large systems, allocate hardware test points along with the software test points. The software test points can be set, cleared, and toggled as you pass through your code. The hardware test points are great if you have an FPGA in your design. You can bring out internal FPGA signals to track the operation of the hardware combined with the software. These hardware test points can be places to connect the scope, but they could also be just simple LED indicators. On one system, we connected an LED to come on whenever we were writing to EEPROM memory. We left this in and it's a good tool to be sure production units complete the updating of EEPORM before power is turned off. We can also monitor the LED and see which operator actions cause an EEPROM write and then check that against the design. ■

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 is currently working on a mobile communications system that announces highway info. He is also a nationally ranked revolver shooter.

RESOURCES

Unit testing discussions, Microsoft Developer Network, http://msdn.microsoft.com/en-us/library/aa292197%28VS.71%29.aspx.

Unit testing, Wikipedia, http://en.wikipedia.org/wiki/Unit_testing.

ICbank
Gateway to Electronics

3 PORT INTERFACE RS-485 to Ethernet Converter

Only $170

RS-485 to Ethernet Converter

Powerful feature

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

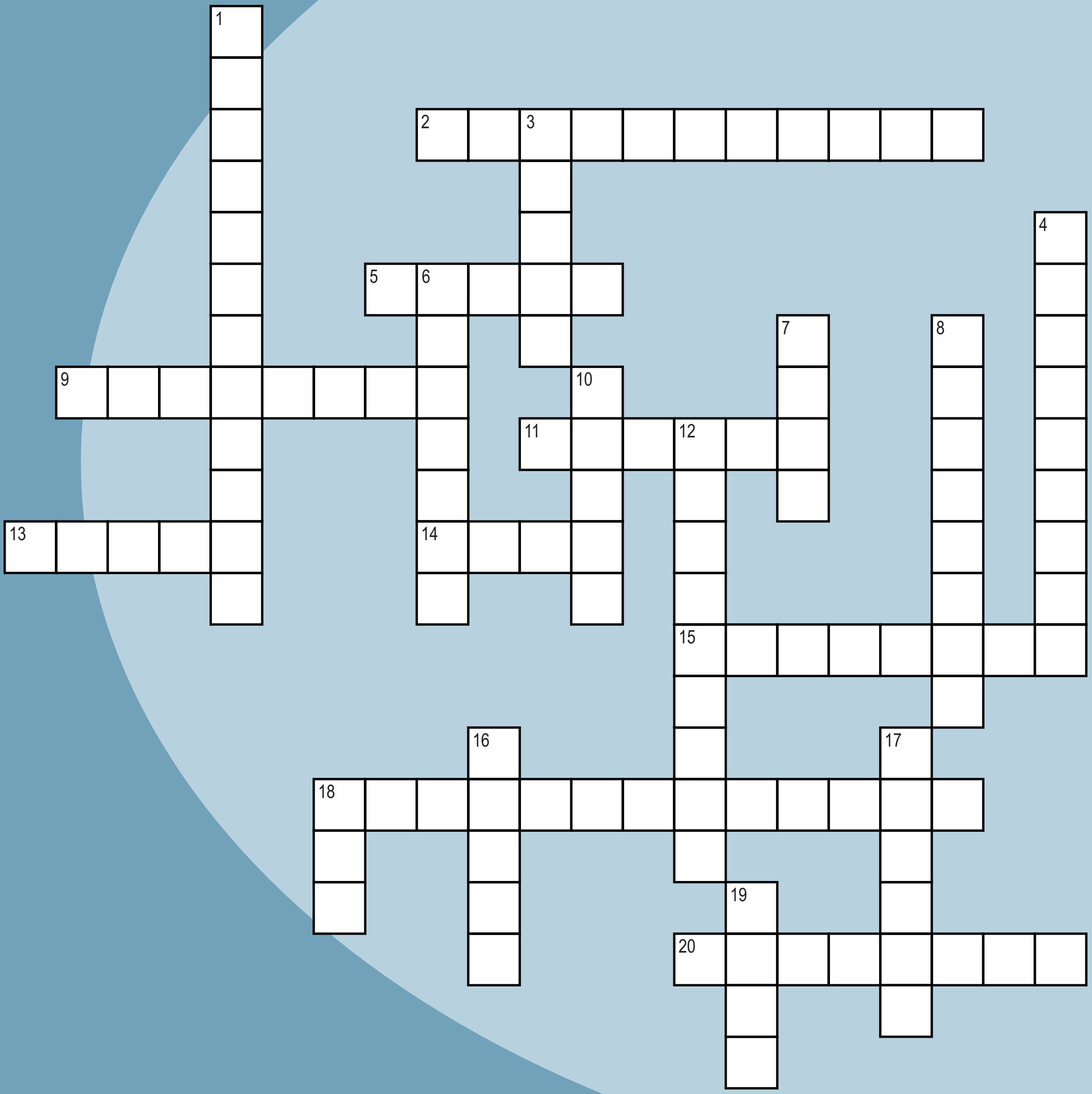
Specification

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

ICbank
Gateway to Electronics

www.icbank.com

CROSSWORD



Down

1. NAK means Not ______
3. A440 Standard
4. Condsenser
6. Location in memory
7. 10^{-18}
8. Jagged, sharp waveform
10. Wb
12. 1°/60
16. 0.016
17. Greek: *haptesthai*
18. One of these is 251.9 calories
19. 12 points

Across

2. <sup> </sup>
5. A "bay" for cable connections
9. Within a system
11. Hypotenuse/adjacent
13. App status
14. Opposite/hypotenuse
15. The "L" in an LC circuit
18. Two-way data transmission
20. Oil-drop demonstration

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.

I2C SPI 1 Wire

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

www.i2cchip.com

Weather Instruments for PCs

AAG electrónica

www.aagelectronica.com

bob
basic overlay board

BOB-4
Low-cost video data overlay module

Features:
Automatic text scroll and crawl
'TTL-232' and SPI control ports
Any size for custom fonts
Vector and bitmap graphics

www.decadenet.com

DECADE ENGINEERING
503-743-3194 Turner, OR, USA

AVOCET SYSTEMS, INC.
www.avocetsystems.com

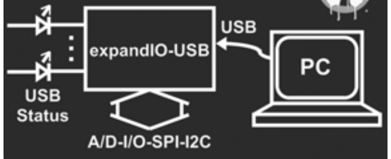
Microcontroller Development Tools

68000 68010 683xx
68HC11 68HC12
68HC16 Z80/EZ80
6502 65816
PIC16

Macro Assemblers
C Compilers
Software Simulators

(207) 596-7766
sales@avocetsystems.com

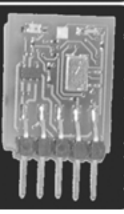
Revolutionary new expandIO-USB chip



- Ideal for adding USB to sensors & peripherals
- No drivers needed for Windows, Mac, Linux
- No microcontroller programming required
- Also check out our USB-232 USB to UART

www.hexwax.com - Buy from Mouser & Farnell

Amazing
PIC programmer
Most devices supported
ICSP, SQTP, & copy limits
$32 at Digikey & Mouser
www.flexipanel.com



Adapt9S12
Modular Prototyping System
For education & development:
*** Assembler, BASIC, C, or Forth**
*** Supports 9S12A,B,C,D,E,N,X**
*** Robotics, Mechatronics, & Automotive Apps**

freescale
Alliance Member

Evaluate * Educate * Embed

www.TechnologicalArts.com

www.can232.com

Only $108 €89

The original (Controller Area Network) CAN to RS232 serial converter.

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 independant
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

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.

Google Checkout
Fast checkout through Google

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

ATTENTION

Print Magazine Readers - BONUS CONTENT NOW AVAILABLE

The following Circuit Cellar bonus content is now available for you to r ead online or in a downloadable PDF. Just visit Circuit Cellar's home page and click on the link to All Bonus Content.

Issue #228: NimbleSig III A New and Improved DDS RF Generator
Thomas Alldread

Sound Synthesis Made Simple (Full article plus video example)
A Multi-MIPS Music Box
Peter McCollum

Issue #229: USB I/O Expansion
Brian Millier

Issue #230: Verification and Simulation of FPGA Designs
Sharad Sinha

Issue #231: Arduino-Based Temperature Display
Mahesh Venkitachalam

Buddy Memory Manager
Sitti Amarittapark

Issue #232: Measuring Propagation Delay with a Universal Counter
Neil Foricer

Issue #233: The Evolution of Rabbits Five Generations of Rabbit Microprocessors
Monte Dalrymple

Are you interested in writing for Circuit Cellar? Consider a submission to Circuit Cellar's bonus section in the Digital Plus venue. As you see from this statement of availability, the bonus section of Digital Plus is available to all Circuit Cellar readers. Authors are choosing to be published in our bonus section for a variety of reasons. These reasons include but are not limited to:

- Articles of various lengths can be published in the digital venue
- Follow-up articles are published in the bonus section without concern for the impact on the current issue's theme
- Articles may include audio or video enhancements
- Speed to publication. Space restrictions in the print magazine can delay publication. There are fewer restrictions on the digital side.

Whether you want to submit an article for print publication or for publication in the bonus section of Digital Plus, please write to editor@circuitcellar.com to present your ideas.

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

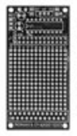
Serial LCDs
2-line x 16 $39.95
4-line x 20 $49.95
Quantity Discounts Available!

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

PICPROTO™ Prototyping Boards



Double-Sided with Plate-Thru Holes
Circuitry for Power Supply and Clock
Large Prototype Area
Boards Available for Most PIC® MCUs
Documentation and Schematic

Pricing from $8.95 to $19.95

BASIC Compilers for 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

See our full range of products, including books, accessories, and components at:
www.melabs.com

Add a color touch interface to your embedded product!

- High level RS232 interface
- Easy to program
- Low cost
- 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

Prolific USA

World Class Solutions

www.ProlificUSA.com

USB 3.0

USB 2.0

GPS

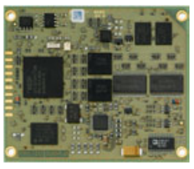
SoC

PHYTEC

Inside great products. Behind great ideas.

phyCORE® System on Modules:

- shorten time-to-market
- reduce development costs and avoid substantial design issues and risks
- Windows® Embedded CE and Linux BSPs (processor-dependent)
- $129/unit benchmark price at 1K for ARM9-based SOM
- Design Services available to assist with deployment into target applications



phyCORE-LPC3250

ARM11: i.MX35, i.MX31
ARM9: i.MX27, LPC3250, LPC3180
Cortex M3: STM32F103
ARM7: LPC2294
XScale: PXA270
x86: Z510, Z520, Z530 (Atom®)
Blackfin: ADSP-BF537
Coldfire: MCF5485
PowerPC: MPC5554, MPC5567, MPC5200B, MPC565, MPC555

phyCORE® Rapid Development Kits include SOM, Carrier Board, LCD (kit specific), schematics, software, free BSP for applicable kits and a start-up guarantee. The Carrier Board serves as a target reference design, allowing the SOM to easily port to the user's target hardware.

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

---

iMAGEcraft C

C development tools for AVR • PSoC Pro • CPU12 ARM • MSP430 Propeller

NEW: V8 IDE!

Dinkumware libraries

- Power and Simplicity -

Plus: Hardware CANUSB, CAN232 ELMICRO kits, etc.

Full-featured 45-day demos on our website!

www.imagecraft.com

info@imagecraft.com

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

KETEREX®

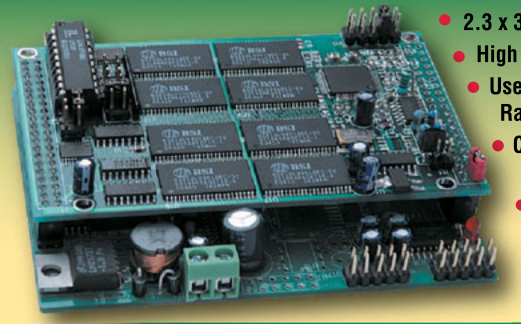
Turn your PC into a SPI, I2C, SMBus, or GPIO Controller

Affordable Test and Measurement Tools

www.keterex.com

GR16™ *Low Cost, High Speed, Analog Signal Grabber — Up to 10MHz SPS, 16-bit ADC, and 8 MB FIFO*

Start at $149



- 2.3 x 3.6 x 0.5 inches
- High speed 16-bit ADC (ADS1610)
- User Programmable Sample Rate, up to 10M SPS
- Configurable FIFO data size, up to 8 MB.
- Stackable multi units sharing Sync, and Trigger
- Driven by 16-bit TERN controllers

100+ 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.

TERN INC. 1950 5th Street, Davis, CA 95616 USA

Tel: 530-758-0180 • Fax: 530-758-0181 www.tern.com • sales@tern.com

---

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

*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

CROSSWARE®

Tools for Embedded Development

ARM7
ARM9
CORTEX™-M3

A quality tool suite for ARM® microcontrollers

- ❑ *C/C++**
- ❑ *Code Wizards*
- ❑ *Debugging***
- ❑ *Simulation*
- ❑ *Support*

* Embedded C++
** Including Jaguar interface

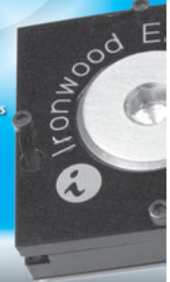
Advanced software tools since 1984

www.crossware.com
info@crossware.com

GHz BGA/QFN Sockets 0.3mm to 1.27mm

Industry's Smallest Footprint

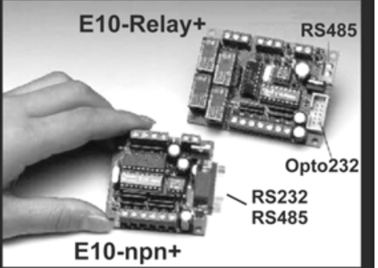
- Up to 500,000 insertions
- Bandwidth to 40 GHz
- 2.5mm per side larger than IC
- Ball Count over 3500, Body Size 2 - 100mm
- Four different contactor options
- Optional heatsinking to 100W
- Four different Lid Options
- <25 mΩ Contact Resistance throughout life



Ironwood ELECTRONICS 1-800-404-0204
www.ironwoodelectronics.com

The $69 PLC

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



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

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

Triangle Research

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

Flashlite 186

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

$69 QTY 1

$99 Development System

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

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

JK microsystems

Low Cost CAD Software for Windows XP, NT and Vista

- Circuit design package with schematic entry circuit-board layout with autorouting and simulation for only $499!
- Buy modules starting at $119 (SuperCAD, SuperPCB, mentalSPICE & SuperSIM)
- Order and download instantly
- Full up package allows up to 16 layers plus 4 power planes
- Manufacture circuit boards at any board house

Mental Automation, Inc.
253-858-8104
www.mentala.com

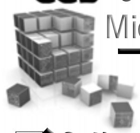
ALL ELECTRONICS
CORPORATION

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

Checkout V4.1xx

CCS• C Compiler for all Microchip PIC® MCU



New in the compiler:

✓ **Optimzed String Handling**
- New Built-in Functions

✓ **New Syntax Extensions**
- Use standard operations directly on strings

✓ **IDE Enhancements**

FREE Download for customers on Active Maintenance
www.ccsinfo.com/V4.1xx
sales@ccsinfo.com · 262.522.6500 x35

It writes your Full Speed 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!
www.TraceSystemsInc.com
301-262-0300

Trace SYSTEMS, Inc.

INDEX OF ADVERTISERS

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

Page		Page		Page		Page	
75	AAG Electronica, LLC	77	Earth Computer Technologies	32, 34	JKmicrosystems, Inc.	68	PCB-Pool
32	AP Circuits	7, 49	Elektor	78	JKmicrosystems, Inc.	C4	Parallax, Inc.
78	All Electronics Corp.	68	Elprotronic	21	Jameco	77	Phytec America LLC
76	Apex Embedded Systems	12	Elsevier	28	Jeffrey Kerr, LLC	25	PoLabs
75	Avocet Systems, Inc.	50	Embedded Developer	77	Keterex, Inc.	27	Pololu Corp.
33	CWAV	19	ExpressPCB	56	Lakeview Research	77	ProlificUSA
64	Calao Systems	75	FlexiPanel Ltd.	76	Lawicel AB	77	Reach Technology, Inc.
10	Cleverscope	11	Front Panel Express LLC	11	Lemos International Co. Inc.	5	Spark Fun Electronics
69	Comfile Technology, Inc.	13	Grid Connect, Inc.	77	MCC (Micro Computer Control)	40, 41	Technologic Systems
78	Crossware Products, Inc.	28	HobbyLab, LLC	78	Mental Automation, Inc.	75	Technological Arts
78	Custom Computer Services, Inc.	75	I2CChip	17	Microchip Technology, Inc.	77	Tern, Inc.
75	Decade Engineering	58, 73	ICbank, Inc.	76	microEngineering Labs, Inc.	78	Trace Systems, Inc.
34, 57	DesignCon	77	Imagecraft Creations, Inc.	29	Mouser Electronics	78	Triangle Research Int'l, Inc.
28	DesignNotes	1	Imagineering, Inc.	C2	NetBurner	2, 3	WIZnet
C3	Digi International	56	Intuitive Circuits LLC	56	Nurve Networks LLC		
64	EMAC, Inc.	78	Ironwood Electronics	13	PCBCore		

PREVIEW of February Issue 235

Theme: **Wireless Communications**

RFID-Based Liquid Control (Part 1): Working with Off-the-Shelf Components

Floating Point for DSP

Advanced Encryption Standard: Understanding AES Without Math

FPGA Embedded Microcontroller Environment

THE DARKER SIDE **Living with Errors:** An Introduction to Forward Error Correction

ABOVE THE GROUND PLANE **Totally Featureless Clock (Part 1):** WWVB Simulator

FROM THE BENCH **Sun Tracker (Part 1):** Create a Directional Light Sensor

SILICON UPDATE **A Winning Hand:** Betting on the ARM Cortex-M3

ATTENTION ADVERTISERS

March Issue, 236 Deadlines

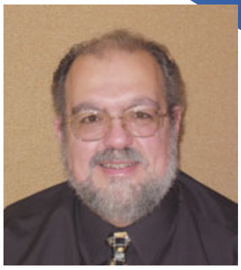
Space Close: Jan. 14
Material Close: Jan. 21

Theme
Robotics

Bonus Distribution
IPC Printed Circuits Expo;
APEX & The Designers Summit;
The Trinity Robotics Contest

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

A Handshake and a Future

I got an interesting letter from a young engineer a couple months ago. He was a long-time *Circuit Cellar* reader who wanted to know how I got my start and if it was still possible to emulate my career path in today's business climate. Apparently, he was at his third job out of college and increasingly disappointed that no one shared his entrepreneurial fervor and design creativity. Basically, he wanted to start his own company and was asking for advice.

While I'm sure he'd prefer I simply send him a detailed game plan, my answer to such letters is usually as follows. In my experience, making a successful business isn't just about entrepreneurial drive and a good product idea. It's about blending a good product idea with the proper resources and a bit of humility. *Circuit Cellar* is a prime example. It is a successful magazine because it fills a specific professional-level publishing niche better than anyone else, but it wouldn't have happened without about a million dollars (at least I didn't go to Las Vegas instead) and some real talent. Yeah, my name is all over the place, but I'll be the first one to tell you that *Circuit Cellar*'s talent resource is the vast support of its readers, authors, and contributors like Ed Nisley, Dave Tweed, Robert Lacoste, George Martin, Jeff Bachiochi, and Tom Cantrell. They've all been making me look good for years. ;-)

The formula for starting and continuing a successful business is straightforward: design a product, sell a bunch, keep doing it. Continuing this process forever is the difficult part. Unlike Microsoft or Cisco who can buy a government or two to perpetuate its existence, small business usually has to rely on inheritances or mergers. The problem with this idea is that too often entrepreneurs fail to pass the torch at the right time and end up taking their creations to the grave with them. And, while Ed Nisley was kind enough to remind me that this concept seemed to work well for the Egyptians, I was determined that this would not happen to *Circuit Cellar*.

This is not the first time *Circuit Cellar*'s future has been considered. About eight or nine years ago three of the large American mega-trade-magazine publishers independently approached me about purchasing *Circuit Cellar*. With all of them, the first concern was *Circuit Cellar*'s profitability. The second thing out of their mouths was: "Of course, you'll have to work for us for the next five years and achieve certain revenue goals we set as part of the earn-out." Let's face it, Guys, after 33 years of this, I don't do well taking orders from pencil-pushing VPs in 100-magazine conglomerates. I believe they have their place in the world, and I highly respect some of them, but as far as *Circuit Cellar* goes, I'll keep my marbles, thank you.

About a year ago, I was approached at the Embedded Systems Conference by Elektor International Media, a subsidiary of Zhomer Media Business BV in the Netherlands. They were introducing their new magazine for the American market, *Elektor USA*. It was refreshing that *Elektor*'s first comment to me wasn't, "We will bury you." But instead, it was to tell me how all the Elektor engineers read *Circuit Cellar* and thought it was a great magazine with impressive editorial. More importantly, their second subject was about how they would love to expand *Circuit Cellar* and increase its presence around the world. Certainly, if they were trying to schmooze me, it worked, because I was still listening. ;-)

To make a long story short, during the last year, I've come to understand *Elektor*'s resources, talents, and ambitions. I believe it's the proper match. I've been looking for a merger partner with global ambitions along with the resources and staff to truly expand *Circuit Cellar*'s distribution around the world. Similarly, *Elektor* realizes that if they want a significant publishing presence in the U.S., it is with *Circuit Cellar*.

The most important factor in my decision is that *Elektor* and *Circuit Cellar* magazines are complementary rather than competitive. It is a good combination where each can draw resources from the other while remaining completely independent. And, even though this is an American magazine, the culture of Dutch publishing is to keep a firm division between editorial and advertising. (I was surprised to learn that this division is actually Dutch law.) *Elektor* has publically stated its intention to retain *Circuit Cellar*'s editorial assets and keep the editorial level on the high-integrity path it has always traveled. Our uncompromised editorial is a prime reason that all of you have supported and contributed to *Circuit Cellar* for so many years. It is my belief and understanding that *Elektor* has the same future in mind for *Circuit Cellar*.

Finally, I'm not jumping ship either. I don't have any quotas, earnout thresholds, or anything like that. I just have a desire to see that what I started 22 years ago keeps going. Unless they get tired of my experience and consulting (meddling), I'll be around for years.

So, now you know the game plan for *Circuit Cellar*. Some say the best marriages are made in heaven. I don't necessarily disagree, but first I'd put my wager and trust on a marriage made across the lanai table on a warm afternoon at Steve and Jeannette's winter cottage.

steve.ciarcia@circuitcellar.com

Steve

Winning Ideas Start Small and Grow Big

Digi's Wireless Green Design Contest

$20,000 in Cash Prizes

Check out our video at **www.digi.com/gogreen** to see how easy it is.

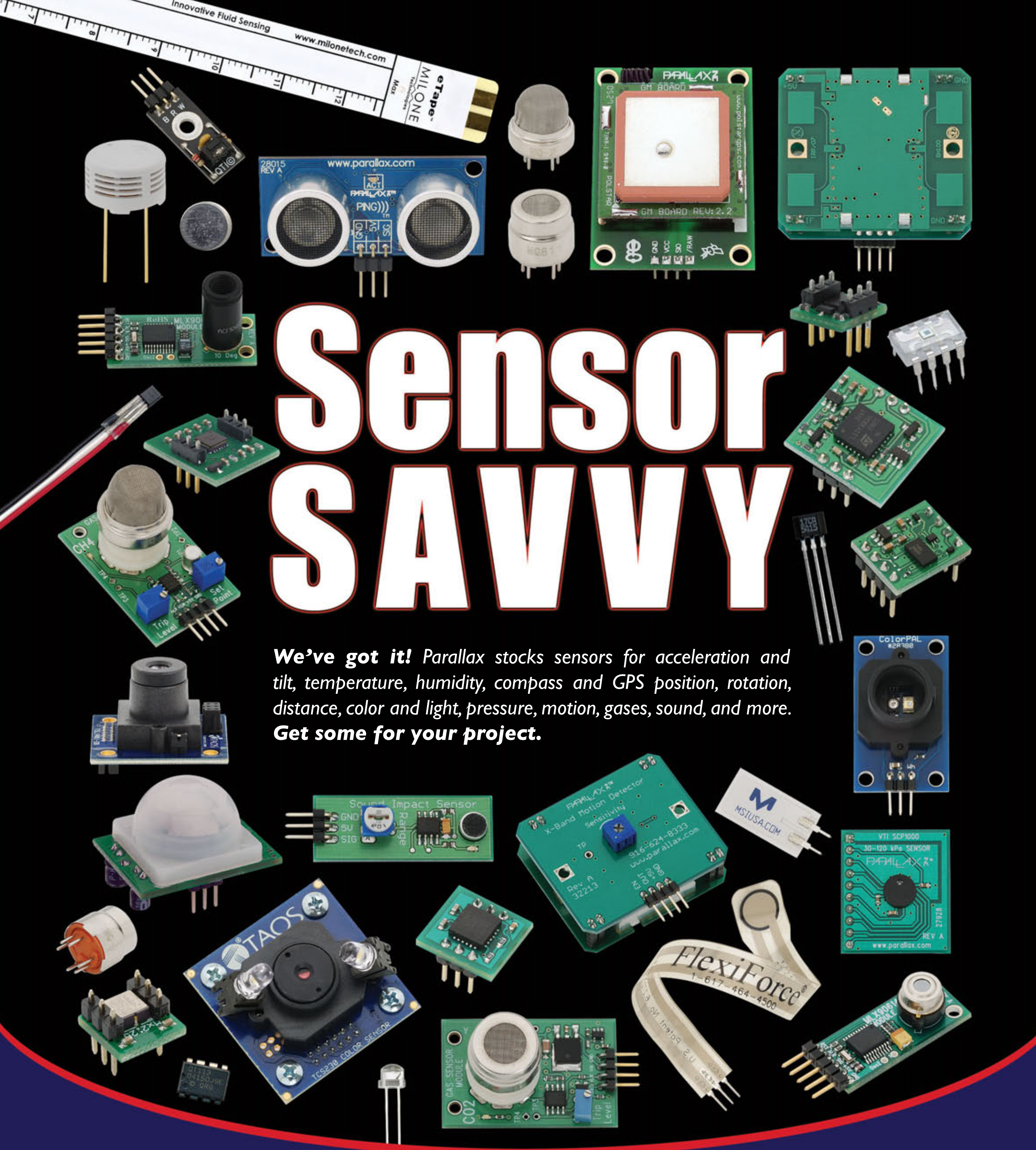
Digi International is challenging anyone with a good idea, no matter how big or how small, to participate in our Wireless Green Design Contest. iDigi Starter Kits provide the hardware, software and services to grow a good idea into a great green solution. Buy a $149 kit, submit your design and you could win!

iDigi™

Go green. Go Digi.

Digi®

www.digi.com/gogreen



Sensor SAVVY

We've got it!** Parallax stocks sensors for acceleration and tilt, temperature, humidity, compass and GPS position, rotation, distance, color and light, pressure, motion, gases, sound, and more.* ***Get some for your project.

Discover **Parallax Sensors** at www.parallax.com
Order online orcall our Sales Department toll-free:
888-512-1024 (Monday - Friday, 7 a.m. - 5 p.m., PT).

Prices subject to change without notice. Parallax and the Parallax logo are trademarks of Parallax Inc.

PARALLAX INC

www.**parallax**.com

Friendly microcontrollers, legendary resources.™

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

PROJECT CORNER

by Andrew Lindsay

Arduino Internet Clock

The Adruino Internet Clock is a timepiece that acquires the current date and time information from Network Time Protocol (NTP) servers on the Internet. It then displays the information on an LCD for easy reading.

The Adruino Internet Clock obtains current date and time information from widely available Network Time Protocol (NTP) servers on the Internet. It is based on the Arduino open-source electronic prototyping platform. This is an easy-to-use microcontroller board based on Atmel ATmega chips.

In this article, I'll describe the design process. I'll wrap up with some ideas about possible future design upgrades.

HARDWARE

The hardware comprises a main board with the ability to receive stackable add-on boards called shields. The shields then provide additional functionality to the basic Arduino digital and analog I/O. In my design, I used two shields. The first is an Ethernet shield. There are a number of Ethernet shields available. The official Arduino shield is based on the WIZnet W5100 Ethernet chip, and there are a number of versions based on the Microchip Technology ENC28J60 Ethernet chip. The libraries for the two versions are not compatible with each other.

The second shield I used was a prototyping shield. It includes a matrix of holes that allows small circuits to be built up to create your own custom hardware. I have a small breadboard installed here so that small circuits can be built up without the need for soldering or flying leads to an external breadboard. The prototyping shield holds the interface circuit to connect the LCD to the Arduino.

With the clock fully assembled and the software loaded, it needs to be connected to a local LAN where it is able to obtain the time and display it. There is a simple web interface where a number of settings including IP address and display size can be adjusted. The settings are then stored in the onboard EEPROM to be reloaded when it is next powered up. There is the option to reset the configuration to its default values by grounding digital I/O pin 0 while resetting the Arduino.

STARTING WITH ARDUINO

I worked with embedded systems for a number of years in the early part of my career, but lost interest when the cost of development systems began to rise. Fortunately, the recent arrival of the Arduino boards—with their simplified IDE and built-in program upload—rekindled my interest in microcontrollers.

To begin, the first step is almost always to get it to flash an LED. After that, it's a case of working through the

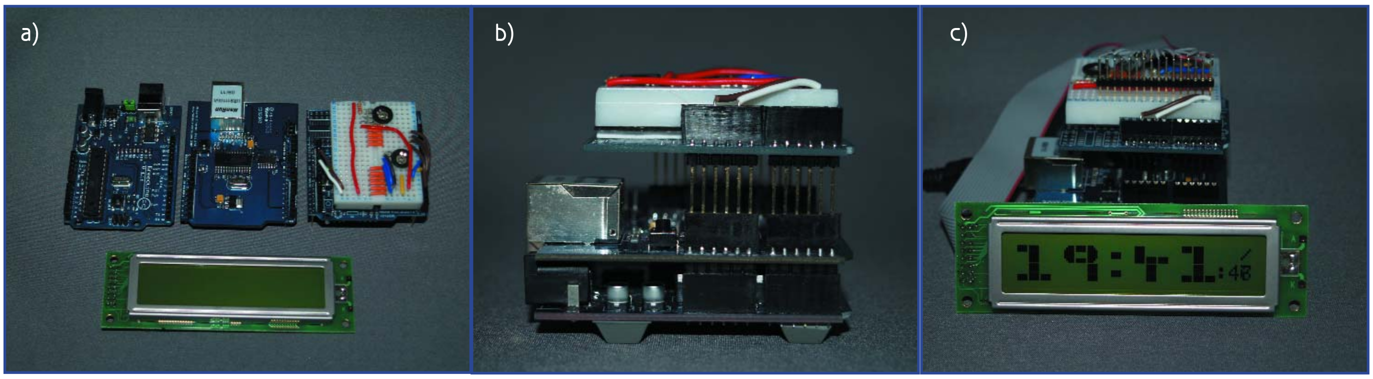


Photo 1a—Here you see the Arduino, shields, and LCD prior to assembly. b—The Arduino and stacked shields. c—Take a look at the Internet clock in operation.

LCD Pin	Digital pin
Rs	8
Enable	7
D4	3
D5	4
D6	5
D7	6

Table 1—These are Arduino-to-LCD connections.

example programs to read and write digital I/O, read analog inputs, and drive servo motors. The next stage involves adding a simple LCD. The software includes a library for the common Hitachi HD44780-based displays. The displays are readily available in a number of sizes including single one-, two-, and four-row displays. With the right software on the Arduino and the host system, it is then possible to feed the display with system status information.

More advanced applications of the Arduino come with the use of the Ethernet shields, which allows devices to be created that are able to communicate with remote systems by means of the TCP/IP and UDP protocols over Ethernet. After trying the client and server examples included with the library, I was looking for a larger project that would combine a number of elements to produce something that would be more challenging and to build something that could be useful. I had seen a design for an Internet clock based on custom AVR hardware and using the same Ethernet chip as I was using. It was then that I decided to port this code on the Arduino platform and improve upon its features.

CLOCK DESIGN

My clock design consists of three main components. The design is built around an Arduino Duemilanove featuring an Atmel ATmega328. I chose this over the ATmega168 variants due to the amount of available flash memory and RAM.

The second component is the ENC28J60-based Ethernet shield. I used it purely for cost reasons. It's less than half the price of the official Arduino Ethernet shield.

The third main component is a ProtoShield with an LCD. There are ready-made LCD shields; but unfortunately, they are not flexible in the pins they use and the pins may conflict with the Ethernet shield's SPI. Using the ProtoShield approach gives me the flexibility to use LCDs of different sizes.

The LCD is wired up in 4-bit mode using the Arduino digital I/O pins 3 to 8, with RW tied to ground (see Table 1). The LCD is controlled using the LiquidCrystal library included in the Arduino IDE. There are some additional functions to create large digits spanning either two or four rows of the display .

Ethernet shield connections are mainly the SPI with the addition of an Enable pin and an Interrupt pin. Even though an interrupt is specified and connected, it doesn't appear to be used within the library, and without cutting the track it can cause problems when using digital I/O pin 2 for other purposes (see Table 2). Arduino digital I/O pins 0 and 1 are normally used by the serial port, leaving only pin 9 in this application as a spare digital I/O pin for future use.

When using the Arduino with the ENC28J60, it is not currently possible to send or receive full-sized Ethernet packets because the Arduino has limited available memory. As a compromise, a buffer of 950 bytes was allocated. This is used to receive packets and to build up packets that are being sent. In order to reduce the size of the created web pages, I've used a feature whereby most web browsers are able to display basic HTML tags without having to include the full HTML/HEAD/BODY tags.

Ethernet	Digital pin
Int	2
Enable	10
MOSI	11
MISO	12
SCK	13
D7	6

Table 2—These are Ethernet shield connections

SOFTWARE OPERATION

The basic operation of the software is as follows: initialize the LCD and show a welcome message; initialize the ENC28J60 and wait for a link up; send an ARP to get MAC address of the router; wait for an ARP response and set the router MAC address when received; send the NTP request; wait in loop ready to receive packets; if the packet is an NTP response, then update display with date and time; if the packet is a ping request, then send a response; if the packet is HTTP request, answer with a static web page or configuration form; if the packet is HTTP Post data and password is correct, update the stored configuration; and if there are any packets received, keep the display updated, check if link is still active, check for the next hour, and request an NTP update.

A simple interrupt service routine is executed once per second. This increments the time, requests the LCD to be updated, and toggles the flashing dots on the time display because digital clocks need flashing dots!

FUTURE DEVELOPMENTS

After completing this project, I can see there is still room for development, either from me or someone else who takes the design and code (this is an open platform). I can try the official Arduino Ethernet shield instead of the ENC28J60 shield. I can add an alarm or timed event facility so that a digital I/O pin is set when a certain date, time, or interval has occurred. I can replace the web-based configuration with a number of buttons and on-screen menus. It is possible to use the analog inputs for this. And finally, I am looking to update the ENC28J60 Ethernet library to address the current shortcomings, such as only using small packet sizes and it being very easy to exceed the allocated buffer causing the Arduino to behave erratically. ■

Author's note: The code for this project is available at my web site, blog.thiseldo.co.uk/?p=336.

Andrew Lindsay (andrew@thiseldo.co.uk) has used and worked with computers since he was 12. In addition to working as a software developer for the last 22 years, he spent a number of years working with embedded systems. Andrew is now an application developer for Telstra International based in London. His technical interests include Linux, Arduino, and tinkering with various electronic gadgets.

Resources

Big number code, www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1245352653.

Tuxgraphics.org, "Introduction to the Tuxgraphics TCP/IP Stack (Third Generation)," www.tuxgraphics.org/electronics/200905/embedded-tcp-ip-stack.shtml.

———, "The Tuxgraphics AVR NTP Clock," www.tuxgraphics.org/electronics/200710/avr-ntp-clock.shtml.

Source

Arduino Duemilanove board
Arduino | www.arduino.cc