Q&A: The Enterprising Roboticist
LOCATION: New Zealand
PAGE: 16

HOW TO: Plan for Component Obsolescence
LOCATION: Canada
PAGE: 50

PROJECT: Parallel Port Interface Replication
LOCATION: United States
PAGE: 60

CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

MARCH 2012
ISSUE 260

ROBOTICS

Robot Navigation with Acoustic Delay Triangulation

Get Started with a Mechatronics Robot

"NakedCPU" Experiments

An MCU-Based Color Identifier

IR Thermal Detection

PLUS

DIY GAMING PLATFORM

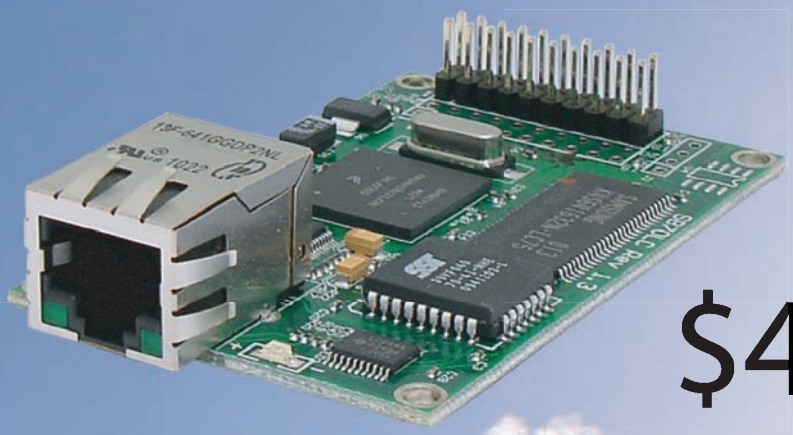
Develop a System from Start to Finish

// Hack a Joystick & Add a New Circuit Board
// Emulate Classic Arcade Hardware
// Sound Interface Device Implementation
// And More



7 25274 75349 9 03>
$7.50 U.S. ($8.50 Canada)
www.circuitcellar.com

The NetBurner SB70LC



$49.00
Qty. 100
SB70LC Board

The complete hardware and software solution

- Low-cost
- 10/100 Ethernet
- Two Serial Ports
- 32-bit Performance
- Digital I/O
- SSH/SSL Option
- Flash Card Ready
- I²C • SPI Interface
- Example applications



$99.00
SB70LC Development Kit

The SB70LC Development Kit

The NetBurner SB70 LC Development Kit is available to customize any aspect of operation including web pages, data filtering, or custom network applications. The kit includes platform hardware, ANSI C/C++ compiler, TCP/IP stack, web server, e-mail protocols, RTOS, flash file system, Eclipse IDE, debugger, and cables. The kit enables you to communicate with peripherals that use SD/MMC Flash Card (including SDHC), SPI, I²C, or the general purpose digital I/O interface. The NetBurner security suite option includes SSH v1, v2 and SSL support.

NetBurner
Networking in One Day!

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

freescale™
Alliance Member



Coming Soon!

The RL78 Green Energy Challenge

$20,000 in cash prizes

Competition starts March 26, 2012

With an incredible ecosystem of hardware, software and third-party vendors, Renesas' family of RL78 MCUs are optimized for efficient power consumption and deliver up to 41 DMIPS at 32MHz. These versatile MCUs offer a true low-power platform for the most demanding 8- and 16-bit embedded applications.

Renesas, along with *Circuit Cellar* and *Elektor*, invites you to experience true low power by developing a green energy application using the RL78 MCU and IAR tool chain. **Succeed and win a share of $20,000 in cash prizes!**

RENESAS RL78

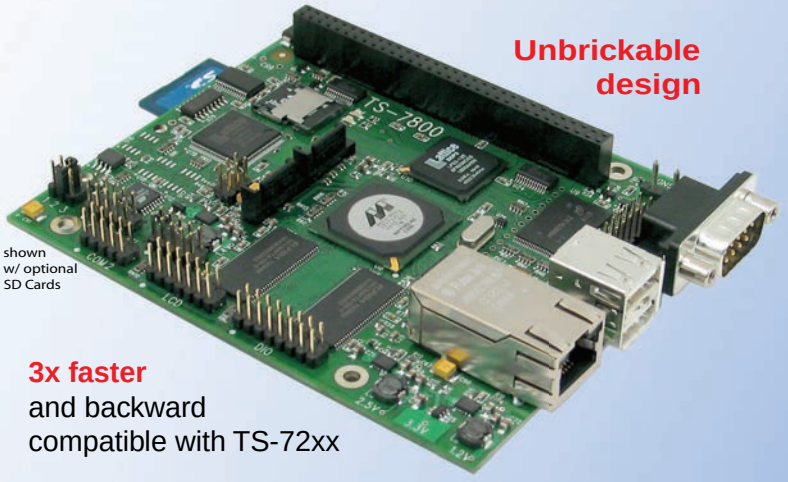
In association with *Elektor* and *Circuit Cellar*

Be the green you see in the world and get ready for the RL78 Green Energy Challenge.

www.circuitcellar.com/RenesasRL78Challenge

Embedded Systems

High-End Performance with Embedded Ruggedness



shown w/ optional SD Cards

Unbrickable design

3x faster and backward compatible with TS-72xx

TS-7800
500 MHz ARM9

- Low power - 4W@5V — $229 qty 100
- 128MB DDR RAM
- 512MB high-speed (17MB/sec) onboard Flash — $269 qty 1
- 12K LUT 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 0.7 seconds
- Linux 2.6 and Debian by default

TS-TPC-7390
7" Color Touch Panel Computer

- Low Power, Industrial Quality Design
- Mountable aluminum frame
- 200 MHz ARM9
- 64MB SDRAM (128MB opt)
- 512MB Flash w/ Debian Linux
- 800 x 480 video core
- Dedicated framebuffer- 8MB RAM
- Audio codec with speaker
- Boots Linux 2.6 in less than 2 seconds
- Unbrickable, boots from SD or NAND
- Runs X Windows GUI applications
- Runs Eclipse IDE out-of-the-box



$449 qty 1

More Touch Panel Computers on our website

- Over 25 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-WIFIBOX-2
A Complete Solution for 802.11g WiFi Applications

NEW!

$185 qty 1

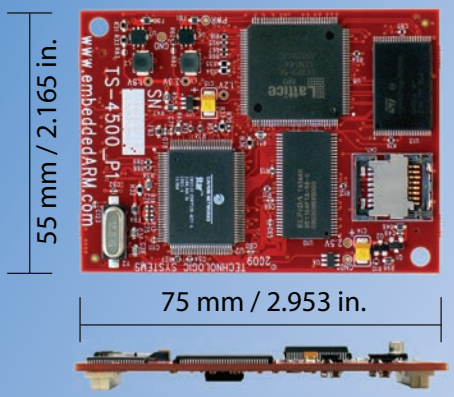


- Low power (3.2 watts), fanless
- Power via 5-12 VDC, USB, PoE (opt.)
- 64MB DDR-RAM
- 256MB ultra-reliable XNAND drive
- Micro-SD Card slot
- RS-232, RS-485, CAN, RTC
- Header with SPI and 11 DIO
- 480Mbit/s USB, Ethernet, PoE option
- Boots Linux 2.6.24 in < 3 seconds
- Un-brickable, boots from SD or flash
- Customizable FPGA - 5K LUT
- Optional DIN mountable enclosure

Ideal for gateway or firewall, protocol converter, web server, WiFi audio, and unattended remote applications

NEW!

series starts at $92 qty 100



- Dual 100-pin connectors
- Secure connection w/ mounting holes
- Common pin-out interface
- Low profile w/ 6mm spacing

TS-SOCKET Macrocontrollers
Jump Start Your Embedded System Design

TS-SOCKET Macrocontrollers are CPU core modules that securely connect to a baseboard using the TS-SOCKET connector standard. COTS baseboards are available or design a baseboard for a custom solution with drastically reduced design time and complexity. Start your embedded system around a TS-SOCKET Macrocontroller to reduce your overall project risk and accelerate time to market.

Current projects include:

- TS-4200: Atmel ARM9 with super low power
- TS-4300: Cavium ARM11 with dual 600 MHz and FPU
- TS-4500: Cavium ARM9 at very low cost
- TS-4700: Marvell PXA168 with video and 1.2 GHz CPU
- TS-4800: Freescale iMX515 with video and 800 MHz CPU
- Several COTS baseboards for evaluation & development

Technologic Systems

We use our stuff.

Visit our TS-7800 powered website at

www.embeddedARM.com

TASK MANAGER

Creativity in Design

The seed for the interview with Hanno Sander on page 16 was sown at the 2008 Embedded Systems Conference in San Jose, CA. Hanno was at the Parallax booth demonstrating his "Dancebot," which is a Propeller-based, two-wheeled balancing robot he wrote about in *Circuit Cellar* 224 (March 2009). Since Circuit Cellar is scheduled to publish Hanno's book *Advanced Control Robotics* later this year, it made sense to interview him for our annual Robotics issue. As you'll learn, Hanno is an enthusiastic designer whose intellect and passion for engineering have enabled him to travel the world and make electronics innovation his life's work. His story should be an inspiration to everyone who reads this magazine.

After you finish the interview, check out the two robotics-related articles featured in this issue. First, on page 20, Larry Foltzer tackles the topic of robot navigation with a fascinating article about position determination and acoustic delay triangulation. Next, turn to the back of the issue for Jeff Bachiochi's article, "Wheel-Free Mobile Robots" (p. 64). Jeff takes you inside a Freescale FSLBOT mechatronics robot. Study the concepts he covers to prepare yourself for your next mobile robot design.

The rest of the issue includes a variety of articles on creative designs and essential engineering topics. I'm sure you'll agree that the following articles are just as inspirational as they are informative.

Engineer and video game enthusiast Chris Cantrell explains how he built a Propeller-based TV gaming platform (p. 28). He describes how he hacked a joystick and got the classic Space Invaders video game up and running.

On page 36, Charles Edmondson presents his Rainbow Color Reader. The compact design can identify and announce colors for visually impaired users.

In the February issue, Alexander Pozhitkov introduced the NakedCPU project. This month he wraps up the article series by describing actual experiments and sets the foundation for future research (p. 42).

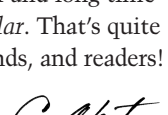
On page 50, George Novacek helps you prepare for the inevitable: microelectronic component obsolescence. You'll find his tips invaluable as you move on to new projects.

Interested in the topic of thermal detection? Want to know how IR thermal sensing works? Turn to Richard Wotiz's article on page 54.

On page 60, columnist George Martin provides his next engineering lesson learned from a real-world project. This month he covers the topic of working with—or without?—printer port connections.

I'd like to wrap up with a note to our staff and long-time readers. This is the 260th issue of *Circuit Cellar*. That's quite an achievement! Thank you, colleagues, friends, and readers!

cj@circuitcellar.com



CIRCUIT CELLAR®

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
C. J. Abate

ASSOCIATE EDITOR
Nan Price

CONTRIBUTING EDITORS
Jeff Bachiochi
Bob Japenga
Robert Lacoste
George Martin
Ed Nisley
George Novacek
Richard Wotiz

PROJECT EDITORS
Ken Davidson
David Tweed

PUBLISHER
Hugo Van haecke

ASSOCIATE PUBLISHER
Shannon Barraclough

ART DIRECTOR
KC Prescott

CONTROLLER
Jeff Yanco

CUSTOMER SERVICE
Debbie Lavoie

ADMINISTRATIVE COORDINATOR
Valerie Luster

MARKETING REPRESENTATIVE
Amanda Anderson

GRAPHIC DESIGNER
Grace Chen

Circuit Cellar's mission is to collect, select, and disseminate need-to-know information around the world in the fields of embedded hardware, embedded software, and computer applications. Circuit Cellar uses an assortment of print and electronic content-delivery platforms to reach a diverse international readership of professionals, academics, and electronics specialists who work with embedded, MCU-related technologies on a regular basis. Our aim is to help each reader become a well-rounded, multidisciplinary practitioner who can confidently bring innovative, cutting-edge engineering ideas to bear on any number of relevant tasks, problems, and technologies.

ADVERTISING
800.454.3741 • 978.281.7708 • www.circuitcellar.com/advertise

ADVERTISING REPRESENTATIVE
Peter Wostrel
Strategic Media Marketing, Inc.
1187 Washington St., Gloucester, MA 01930 USA
800.454.3741 • 978.281.7708
peter@smmarketing.us • www.smmarketing.us
Fax: 978.281.7706

ADVERTISING COORDINATOR
Kim Hopkins
E-mail: khopkins@circuitcellar.com

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

PRINTED IN THE UNITED STATES

CONTACTS

SUBSCRIPTIONS
Information: www.circuitcellar.com, e-mail: circuitcellar@pcspublink.com
Subscribe: 800.269.6301, www.circuitcellar.com, Circuit Cellar, P.O. Box 462256, Escondido, CA 92046
Address Changes/Problems: E-mail: circuitcellar@pcspublink.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® (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 $50, Canada $65, Foreign/ROW $75.** 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 462256, Escondido, CA 92046 or call 800.269.6301.**

Postmaster: Send address changes to Circuit Cellar, Circulation Dept., P.O. Box 462256, Escondido, CA 92046.

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

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

Entire contents copyright © 2012 by Circuit Cellar, Incorporated. All rights reserved. Circuit Cellar is a registered trademark of Circuit Cellar, Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.



▶ CUWIN6300

▶ CUWIN5500

▶ CUWIN5200

Windows CE 6.0 Touch Controller

CUWIN

The CUWIN is a series of Windows CE touch controllers that are more cost-effective than a PC, but with more features than an HMI touch screen.
Create sophisticated applications with C++ or any .Net language.

533MHz ARM CPU
128MB SDRAM & Flash
SD Card Support
Ethernet, RS-232/485
USB, Audio Out
Windows Embedded CE 6.0



C-Programmable Modular Industrial Controller

MOACON

The MOACON is a modular, C programmable, ARM-based automation controller designed for industrial environments.

Choose from a diverse, feature-rich selection of modules including:
Digital I/O
Analog I/O
RS-232/485
Motor Control
Ethernet
High-speed Counter & PWM

BASIC with LADDER LOGIC CONTROLLER



New

Integrated CUBLOC Controller and I/O Board

CB210

The CB210 is an inexpensive, integrated CUBLOC controller and I/O board programmable in both BASIC and Ladder Logic.

I/O Ports x20	10-bit A/D x6
PWM x3	RS-232 x1
80KB Program Memory	3K Data Memory

The CUPC is a series of industrial touch panels with all the features of a modern PC for the most feature-rich user experience.

ATOM N270 1.6GHz CPU
2GB RAM
320GB HDD
SD Card Support
Color Display (1024 x 768)
RS-232, Ethernet
USB, Audio Out
Windows XPe/XP Pro

CUPC

Industrial Touch Panel PC with ATOM Processor



▶ CUPC-P200

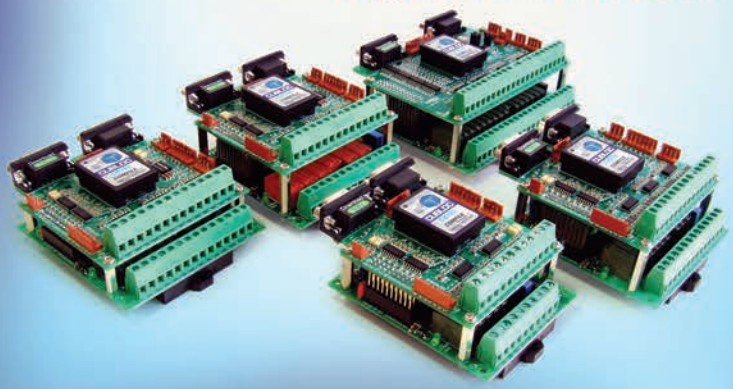
▶ CUPC-P220

The CUSB is a series of compact, CUBLOC-integrated, industrial I/O boards programmable in both BASIC and Ladder Logic.

CUBLOC CB280 Core Module
80KB Program Memory
3KB Data Memory
DC 24V Inputs x9~16
A/D Inputs x2~6
Relay Outputs x6~16
PWM x2~6
High-speed Counters x0~2
RS-232/485 x1~2

CUSB

Integrated Industrial Controllers



COMFILE TECHNOLOGY
1175 Chess Dr., Suite F, FOSTER CITY, CA 94404
call : 1-888-9CUBLOC (toll free)
1-888-928-2562
email : sales@comfiletech.com
www.comfiletech.com

INSIDE ISSUE 260

March 2012 • Robotics

20 **Robot Navigation**
Position Determination from Acoustic Delay Triangulation
Larry Foltzer

28 **All-in-One Joystick**
Build a Propeller-Based TV Gaming Platform
Chris Cantrell

36 **The Rainbow Color Reader (Part 1)**
A Talking Color Identifier for the Visually Impaired
Charles Edmondson

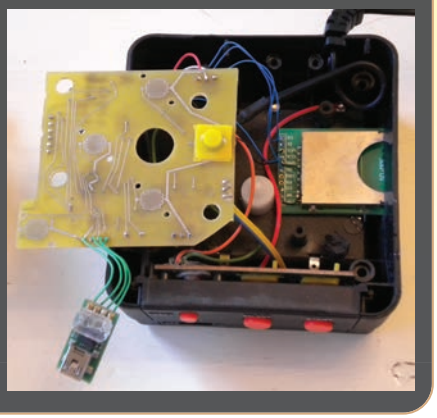
42 **The NakedCPU (Part 2)**
Experimentation and Communication
Alexander Pozhitkov

Robot Design with Hanno Sander p. 16

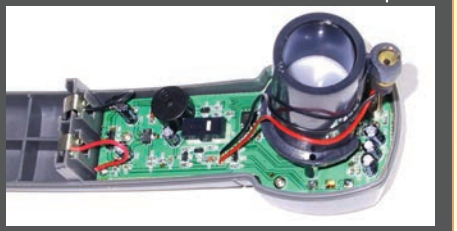


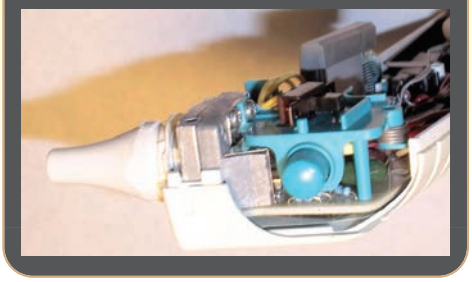
p. 28 DIY Gaming Platform





Thermal Sensing Tech Unveiled p. 54





50 **THE CONSUMMATE ENGINEER**
Microelectronic Component Obsolescence
How to Prepare for an Inevitable Scarcity of Parts
George Novacek

54 **EMBEDDED UNVEILED**
Infrared Thermal Detectors
Richard Wotiz

60 **LESSONS FROM THE TRENCHES**
Printer Port Connections
George Martin

64 **FROM THE BENCH**
Wheel-Free Mobile Robots
Jeff Bachiochi

TASK MANAGER **4**
Creativity in Design
C. J. Abate

NEW PRODUCT NEWS **8**

TEST YOUR EQ **15**

QUESTIONS & ANSWERS **16**
Advanced Robot Technologies
An Interview with Hanno Sander

CROSSWORD **74**

INDEX OF ADVERTISERS **79**
April Preview

PRIORITY INTERRUPT **80**
A Rose by Any Other Name...
Steve Ciarcia



mouser.com

Find It Here. Faster.™

Fastest Part Search.

Best Selection.

Immediate Delivery.

The Newest Products for Your Newest Designs®

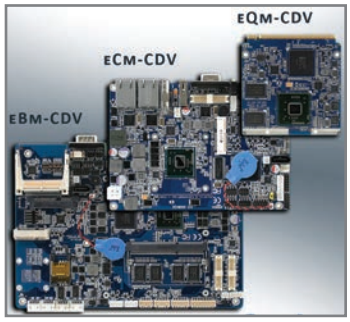
MOUSER ELECTRONICS®

a tti company

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

ENHANCED EMBEDDED BOARDS

The **ECM-CDV**, **EBM-CDV**, and **EQM-CDV** are powered by the next-generation Intel Atom processor N2600, N2800, and D2700 and Intel NM10 Express chipset featuring low-power consumption and high performance. The newest Intel Atom processor series, formerly codenamed "Cedar Trail," is based on Intel's 32-nm process technology, and features a two-chip platform integrating the Northbridge chip with CPU. This two-chip solution will provide system builders with supports for slimmer housing designs. This platform features enhanced performance, energy efficiency, manageability, security functions, and smoother visual experiences.



The 3.5" embedded board, ECM-CDV, supports one 204-pin DDR3 SODIMM system memory with up to 4 GB capacity. ECM-CDV supports dual-channel 18/24-bit (Note: 24-bit is enabled on D2700 only) low-voltage differential signaling (LVDS) with eDP, and multiple display configurations (CRT+LVDS, HDMI+LVDS, or CRT+HDMI). The rich I/O deployment includes one SATA, two COM, seven USB 2.0, and 16-bit GPIOs, making various requirements of embedded applications possible. One audio jack supports Realtek ALC892, 7.1-CH audio. Two Intel 82574L Gigabit LANs and one expansion slot of mini PCIe card (support mSATA) and one CF card slot.

The 5.25" embedded board, EBM-CDV, supports one rank of onboard DDR3 system memory with up to 2-GB capacity and one 204-pin DDR3 SODIMM system memory with up to 2-GB capacity. EBM-CDV supports dual-channel 18/24-bit (Note: 24-bit is enabled on D2700 only) LVDS (with eDP) and multiple display configurations (CRT+LVDS, HDMI+LVDS, or CRT+HDMI). The rich I/O deployment includes one SATA, six COM, eight USB 2.0, and 16-bit GPIOs, making various requirements of embedded applications possible. One audio jack supports Realtek ALC892, 5.1-CH audio. Two Intel 82574L GB LANs and one expansion slot of mini PCIe card (supports mSATA).

The Q7 Module board, EQM-CDV, supports onboard DDR3 system memory with up to 2-GB capacity. EQM-CDV supports single-channel 18-bit LVDS and HDMI + LVDS display configurations to baseboard. For I/O, it includes two SATA ports to baseboard, eight USB 2.0 ports to baseboard, one Intel 82574L GB LANs, and three PCIex1.

Avalue's Intel Atom processor N2000 and D2000 series boards are designed to meet embedded market requests for low-power consumption. They are well suited for designers who need an embedded board with a small form factor for high-performance, multimedia applications. They are also suitable for applications, including POS/KIOSK, digital signage, ordering machine, medical, and transportation.

Contact Avalue for pricing.

Avalue Technology, Inc.
www.avalue.com.tw

2-Mb SERIAL NONVOLATILE FRAM MEMORY

The **FM25V20** is a 2-Mb high-performance serial FRAM device. It is a member of Ramtron's V-Family of FRAM products, which offers a wide operating voltage range. Manufactured on a low-power 130-nm CMOS manufacturing process, the Ramtron V-Family of F-RAM products includes serial I^2C, serial peripheral interface (SPI), and parallel memories.



The FM25V20 features fast access, NoDelay writes, virtually unlimited read/write cycles (1e14), and low power consumption. The latest FRAM device is a drop-in replacement for 2-Mb serial flash and serial EEPROM memories in applications including industrial controls, metering, medical, military, gaming, and computing.

The FM25V20 features an advanced ferroelectric process that yields virtually unlimited endurance of 100-trillion (1e14) read/write cycles and reliable data retention for 10 years. The device uses a fast SPI, operating up to full bus speed of 40-MHz frequency. The low-power operation features a wide voltage operation of 2.0 V to 3.6 V, a typical standby current of 100 µA, and a sleep mode current of 3 µA. A read-only Device ID feature, standard on the serial V-Family devices, enables the host to determine the manufacturer, product density, and product revision. The FM25V20 operates over an industrial temperature range of –40°C to 85°C and is available in standard "green"/RoHS-compliant eight-pin EIAJ SOIC and eight-pin TDFN packages.

Pricing for the FM25V20 starts at **$8.99** for 10,000-piece quantities.

Ramtron International Corp.
www.ramtron.com

NEW PRODUCT NEWS

OFFLINE LED DRIVER DIMS TO ZERO LIGHT INTENSITY

The **MAX16841** is a fixed-frequency offline light-emitting diode (LED) driver that provides flicker-free dimming from maximum light intensity down to zero intensity with both leading-edge (triac) and trailing-edge (transistor) dimmers. Constant frequency control optimizes efficiency at both low- and high-AC line voltages. With a wide 90-VAC to 265-VAC input range, the MAX16841 provides a universal solution for Japanese, Chinese, U.S., and European dimmer models. The driver enables seamless replacement of incandescent and halogen lamps, eliminating compatibility issues with preinstalled dimmers. Because it can operate without electrolytic capacitors, the MAX16841 prolongs the life of LED lamps and provides space and cost savings.

The MAX16841 features a unique, proprietary approach to control a lamp's input current. By actively shaping the input current, the driver ensures flicker-free operation with most electronic transformers and dimmers. This enables LED lamp designers to create drop-in replacements for halogen and incandescent lighting with dimming down to zero light intensity.



The MAX16841 can be designed in isolated (flyback) and nonisolated (buck) configurations making it ideal both for high-AC line voltages (220 VAC to 230 VAC) and low-AC line voltages (100 VAC to 120 VAC). A constant-frequency control scheme improves efficiency by using the optimal conduction mode (DCM or CCM), depending on the input voltage. Dimming efficiency also improves due to the MAX16841's current-control technique. This technique does not require current bleeders for the hold current of the dimmer since sufficient hold current is already guaranteed by the switching regulator itself.

The MAX16841 reduces inventory and design costs since it does not require separate LED drivers to satisfy international voltage requirements. In addition, it can be designed in without electrolytic capacitors. This extends the lifetime of the LED lamp, since electrolytic capacitors are usually the first component to fail in the driver circuit. If electrolytic capacitors are available from an existing design, the MAX16841 enables the lamp to continue to operate even when those capacitors fail. The LED driver saves space and ensures longer lamp life by eliminating bulky electrolytic capacitors that consume valuable space and often fail. It is ideal for retrofit dimmable LED lamps and universal LED bulbs in industrial, commercial, and residential lighting applications.

The MAX16841 is fully specified over the –40°C to 125°C temperature range and is available in an eight-pin SOIC package. Pricing starts at **$1.35** in 1,000-unit quantities.

Maxim Integrated Products
www.maxim-ic.com

5-kV RMS SIGNAL-ONLY ISOLATED CAN TRANSCEIVER

The **ADM3054** is a 5-kV RMS signal-isolated CAN physical-layer transceiver that fully complies with the ISO 11898 CAN standard. The new CAN transceiver leverages Analog Devices's *i*Coupler digital isolator technology and combines a three-channel digital isolator with a CAN transceiver in a 10 mm × 10 mm single-surface-mount, 16-lead, wide-body small-outline IC (SOIC) package.

By reducing printed circuit board (PCB) component count by as much as 70% compared to traditional discrete, optocoupler based component circuits, the ADM3054 transceiver simplifies design while reducing board space by up to 61%.

The ADM3054 is targeted toward applications where an isolated power rail already exists or an additional overwind from an existing isolation transformer can be used. The device's logic is powered with a single 3.3-V or 5-V supply on VDD1 while the bus side uses a single 5-V supply on VDD2 only. Loss of power on the bus side (VDD2) can be detected by the microcontroller using the integrated VDD2SENSE signal. The transceiver has integrated ±36-V fault protection on the bus pins, CANH and CANL, to guard against shorts to power/ground in 12-V and 24-V systems. The ADM3054 also has current-limiting and thermal-shutdown features that protect against output short circuits and situations where the bus may be shorted to ground or power terminals. The transceiver creates an isolated interface between the CAN protocol controller and the physical layer bus. It is capable of running at data rates of up to 1 Mbps and is fully specified over the industrial temperature range of –40°C to 125°C.

The ADM3054 costs **$2.10** in 1,000-unit quantities.

Analog Devices, Inc.
www.analog.com

±5-G MEMS CAPACITIVE MODULES

Designed to support a variety of lower-frequency aerospace, automotive, defense, energy, industrial, and general test and measurement requirements, the low-noise, single-axis model **2210-005** accelerometer module features high-quality MEMS capacitive sensing elements. Sensing elements are packaged within compact, lightweight, anodized epoxy-sealed aluminum housing, occupying a total footprint of just 1 square inch. The accelerometers feature a simple four-wire connection and internal voltage regulator to minimize supply voltage variation effects. Units are relatively insensitive to temperature changes with quick, easy self-calibration. The 210-005 accelerometer modules also feature high-drive, low-impedance buffering to support low-frequency acceleration measurement requirements from 0 (DC) to 400 Hz.



The accelerometer module produces two analog voltage outputs that vary with acceleration. The module offers a choice of single-ended or differential output, which doubles the sensitivity over single-ended output to 800 mV/g.

The sensitive axis is perpendicular to the bottom of the package, with positive acceleration defined as a force pushing on the bottom of the package. The signal outputs are fully differential about a common mode voltage of approximately 2.5 V. The output scale factor is independent from the supply voltage of 9 V to 32 V. At zero acceleration, the output differential voltage is nominally 0 V DC; at ± fullscale acceleration, the output differential voltage is ±4 V DC.

The 2210-005 accelerometer module costs approximately **$500** per unit.

Silicon Designs, Inc.
www.silicondesigns.com

MINIATURE LED DRIVERS

XP Power added four drivers to its LDU series of printed circuit board (PCB) mount DC input constant current light-emitting diode (LED) drivers. The LED drivers provide power outputs of 5 W, 7 W, 14 W, and 20 W. They have a typical efficiency of 93% for the 5-W, 7-W, and 14-W models and 95% for the 20-W devices.

The **LDU20** series provides up to 700 mA output current, and the **LDU05**, **LDU07**, and **LDU14** series has a maximum output of 1,000 mA. The drivers can accommodate a wide range of input voltage from 7 VDC to 30 VDC.

Dimming options include two analog methods (voltage or resistance) and a pulse width modulation (PWM) input signal. The PWM approach has a maximum operating frequency of 1 kHz, with a minimum of 200-ns on/off time. When using an analog dimming method, the output current range can be adjusted from 25% to 100% of nominal current.

A remote on/off control input enables the designer to control the output or sequence the start-up. All models provide protection in the event of a short or open circuit of the LED device.

The LDU series is well suited for a broad range of indoor, outdoor, underwater, and automotive LED applications as well as battery-powered applications.

Pricing starts at **$7.02** for 500-piece orders of the LDU05 5-W part.



XP Power Ltd.
www.xppower.com

DESIGN SOFTWARE WITH ARRIA V AND CYCLONE V DEVICE SUPPORT

Quartus II software version 11.1 delivers support for the latest 28-nm Arria V and Cyclone V devices, and enhanced support for Stratix V devices. The design software includes: support for the Arria V and Cyclone V device families; support for extended temperature on Cyclone IV E devices; enhanced support for the Stratix V device family; and System Console, a scalable, easy-to-use hardware debug and monitoring tool.

The Qsys system integration tool and the MegaWizard Plug-In Manager generate scripts to help set up a simulation environment for third-party simulators. In addition, the software's updated Tasks window now includes flows for gate-level and register-transfer level (RTL) simulation.

Additional enhancements to the software include fractional phase-locked loops (fPLLs) which enable dynamic phase shifting and reconfiguration of many individual counter parameters; a transceiver reconfiguration controller to support the adaptive equalization engine and decision feedback equalizer engine; the EyeQ viewer which provides the ability to see post-equalized eye opening, including the ability to monitor the receiver eye and supporting both horizontal and vertical measurements; and ARM-based system-on-a-chip (SoC) devices including Early power estimator and AXI bus functional model (BFM) beta.

The Quartus II Web Edition software (no license required) can be downloaded for free. Contact Altera for a paid Quartus II Subscription Edition software.

Altera Corp.
www.altera.com

CIRCUIT CELLAR

Microprocessor Design Using Verilog HDL

With the right tools, such as this **new book, designing a microprocessor can be easy.** Okay, maybe not easy, but certainly less complicated. Monte Dalrymple has taken his years of experience designing embedded architecture and microprocessors and compiled his knowledge into one **comprehensive guide to processor design in the real world.**

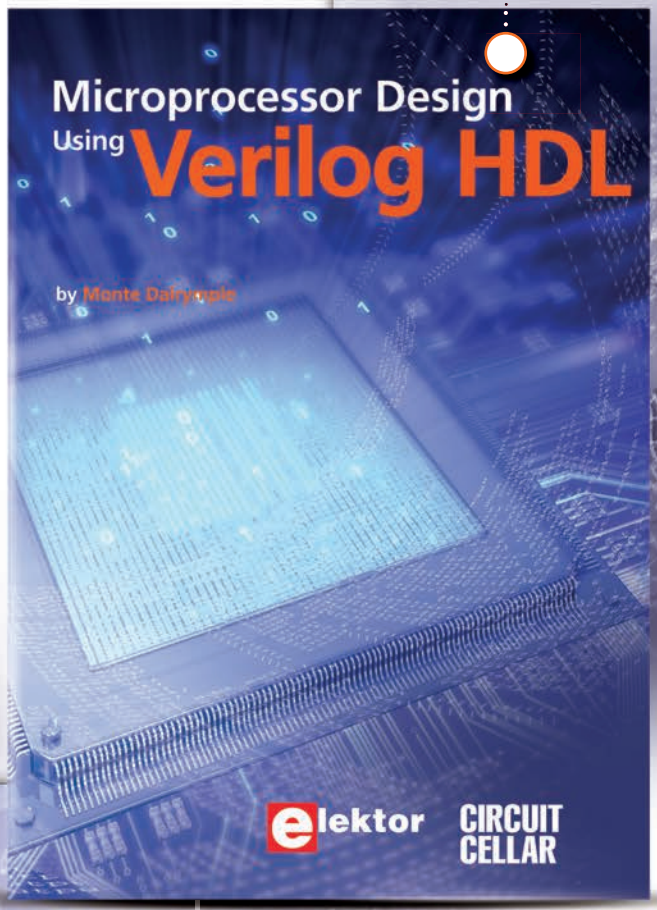
Buy it before March 15, 2012 and **SAVE 15%** off the cover price.

Yours for just **$38.25!**

Monte demonstrates how Verilog hardware description language (HDL) enables you to **depict, simulate,** and **synthesize an electronic design** so you can **reduce your workload and increase productivity.**

Microprocessor Design Using Verilog HDL will provide you with information about:

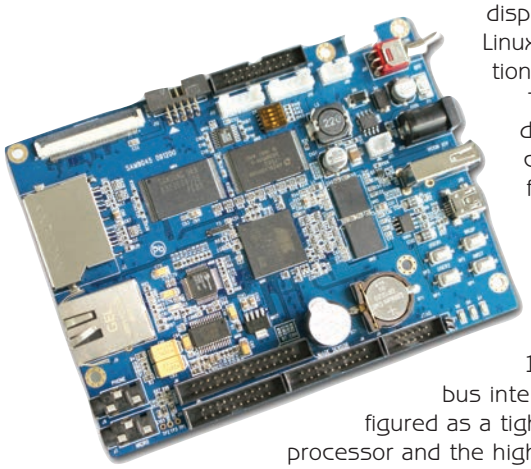
- Verilog HDL Review
- Verilog Coding Style
- Design Work
- Microarchitecture
- Writing in Verilog
- Debugging, Verification, and Testing
- Post Simulation and more!



www.cc-webshop.com

FULL-FEATURED EVALUATION BOARD

The AT91SAM9G45-EVK is a co-branded evaluation kit from Atmel and element14. The full-featured evaluation board is based on Atmel's AT91SAM9G45 processor which runs at up to 400 MHz and features 128-Mb DDR2 memory and 256-Mb SLC NAND flash. The board provides a range of peripherals, including a high-speed USB host/device port, an Ethernet 10/100 interface, two high-speed SD/MMC slots, and a QVGA 320 × 240 liquid crystal display (LCD) panel with resistive touchscreen. In addition, the board supports Linux, WinCE, and Android operating systems (OSes) and offers example applications under these three OSes.



The ARM926EJ-S based AT91SAM9G45-EVK features the frequently demanded combination of user interface functionality and high data rate connectivity, including a LCD controller, resistive touchscreen, camera interface, audio, Ethernet 10/100, and high-speed USB and SDIO. With the processor running at 400 MHz and multiple 100+ Mbps datarate peripherals, the AT91SAM9G45-EVK has the performance and bandwidth to the network or local storage media to provide an adequate user experience. The AT91SAM9G45-EVK supports the latest generation of DDR2 and NAND flash memory interfaces for program and data storage. An internal 133-MHz multilayer bus architecture with 37 DMA channels; a dual external bus interface; and distributed memory including a 64-Kb SRAM which can be configured as a tightly coupled memory (TCM); sustains the high bandwidth required by the processor and the high-speed peripherals. The I/Os support 1.8-V or 3.3-V operation, which are independently configurable for the memory interface and peripheral I/Os. This feature eliminates the need for any external-level shifters.

In addition, the evaluation kit supports 0.8 ball-pitch package for low-cost printed circuit board (PCB) manufacturing. The AT91SAM9G45 power management controller features efficient clock gating and a battery back-up section which minimizes power consumption in active and standby modes.

Contact Atmel or element14 for pricing.

Atmel Corp.
www.atmel.com

Element14
www.element14.com

WIRELESS OPTIMIZATION FOR 3G/Wi-Fi/BT/GPS SUPPORT

The Matrix MXE-5300 series of rugged, quad core, fanless embedded computers is based on the high-performance 2nd Generation Intel Core i7/i5/i3 processor and QM67 chipset. The high-performance computers feature wireless optimization capability, robust mechanical design, and Intel AMT 7.0 support. The MXE-5300 is well suited for intelligent transportation, in-vehicle multimedia, and surveillance and factory automation applications.

Other features of the MXE-5300 include: performance-enhancing mechanical and thermal design, including customized copper conduits to enhance thermal dissipation efficiency; memory and storage stabilizers to withstand the challenges of high-vibration environments; and an anodized aluminum alloy chassis to prevent corrosion. With industrial solid-state drive (SSD) storage devices, the MXE-5300 provides an operating temperature range of –20°C to 70°C for reliable performance in mission-critical and harsh environments. The MXE-5300 also features a unique thumbscrew design, which makes replacement or swapping of hard disk drive (HDD) and RAM/mini PCIe modules easy, lowers maintenance costs, and increases flexibility.



The MXE-5300 supports rich I/O interfaces, including four serial ports, six USB ports (two USB 3.0), four isolated digital I/Os, and four GB Ethernet ports. In addition, the MXE-5300 provides two Mini-PCIe and one USIM slot for wireless needs. Intel AMT 7.0, available on the MXE-5300, enables remote monitoring and control over embedded devices via the Internet, remote system reboot in the event of operating system (OS) failure, and system disconnect from the web when malicious intrusion is detected to prevent migration to other devices.

Contact ADLINK for pricing.

ADLINK Technology, Inc.
www.adlink.com

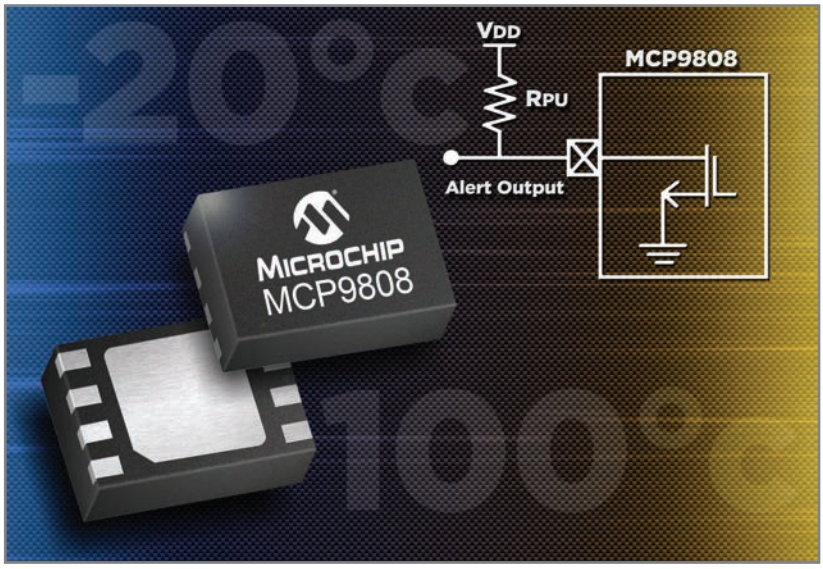
TEMP SENSOR GUARANTEES HIGH ACCURACY & RESOLUTION

The **MCP9808** silicon temperature sensor guarantees a high accuracy of 0.5°C from –20°C to 100°C, along with a high temperature resolution of 12 bits (0.0625°C/LSB). The MCP9808 comes in small MSOP and 2 mm × 3 mm dual-flat no leads (DFN) eight-pin packages for space-constrained applications. Additional features include shutdown, an under/over-temperature monitor and a critical-temperature alert.

The MCP9808 is well suited for applications that require high accuracy over a variety of temperatures, such as thermostats (–20°C to 45°C), personal computers (85°C), and industrial/automotive (125°C). In addition, many applications can benefit from reduced energy usage via accurate temperature monitoring and control that enables more efficient operation.

The sensor features shutdown and temperature monitoring/alert features, to help adjust the real-time power utilization of everything from residential HVAC systems to industrial equipment. The shutdown feature reduces power consumption in both line- and battery-powered systems. The on-chip over/under temperature monitor can improve system performance through thermal optimization. The MCP9808 also features a critical-temperature alert that can increase the system's life by preventing harmful operating temperatures.

Pricing for the MCP9808 starts at **$0.84** each in 5,000-unit quantities.



Microchip Technology, Inc.
www.microchip.com

NPN

Get PUBLISHED. Get NOTICED. Get PAID.

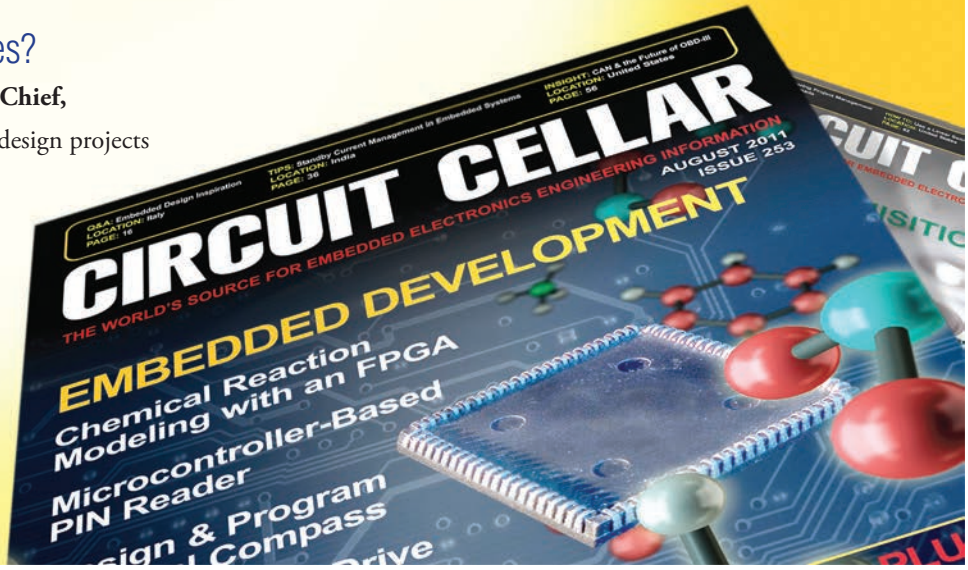
Circuit Cellar feature articles are contributed by professional engineers, academics, and students from around the globe. Each month, the editorial staff reviews dozens of article proposals and submissions. Only the best make it into the pages of this internationally respected magazine.

Do you have what it takes?

Contact **C. J. Abate, Editor-in-Chief,** today to discuss the embedded design projects and programming applications you've been working on and your article could be featured in an upcoming issue of *Circuit Cellar* magazine.

editor@circuitcellar.com

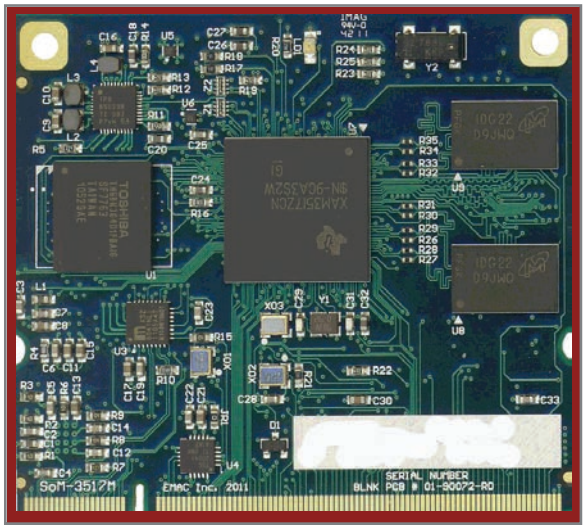
CIRCUIT CELLAR



LOW-POWER ARM CORTEX-A8 SoM WITH 3-D VIDEO

The **SoM-3517M** is a system-on-module (SoM) based on the Texas Instruments ARM Cortex-A8 processor. This fanless 600-MHz SoM features an Ethernet PHY along with four serial ports. The SoM utilizes up to 512 MB of external DDR2/SDRAM and 1 GB of NAND flash. It also includes an on-module SD flash socket and an MMU that enables it to run Linux and WindowsCE operating systems.

The SoM-3517M incorporates the same small 200-pin SODIMM form factor utilized by other EMAC SoM modules. All of the ARM processor core is included on this tiny board including flash, memory, serial ports, Ethernet, SPI, I²C, I²S audio, CAN 2.0B, pulse width modulators (PWMs), timer/counters, ADCs, digital I/O lines, video, and a clock/calendar. In addition, the SoM-3517M provides a math coprocessor, and 2-D/3-D accelerated video and image scaling/rotation.



The SoM-3517M is designed to plug into a carrier board that contains all the connectors and any custom I/O required for the application. This approach enables the customer or EMAC to design a custom carrier board that meets the customer's I/O, dimensional, and connector requirements without having to worry about the processor, memory, and standard I/O functionality. Using this SoM approach, a semi-custom hardware platform can be developed in as little as a month.

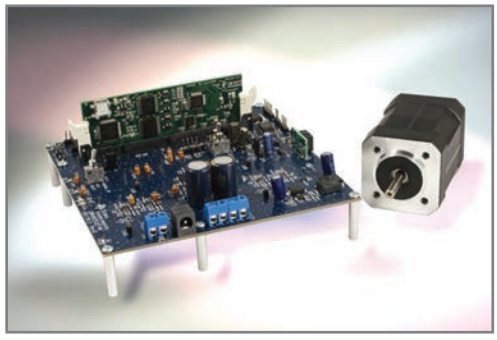
Contact EMAC for pricing.

EMAC, Inc.
www.emacinc.com

MOTOR CONTROL KIT FOR SPINNING THREE-PHASE BRUSHLESS MOTORS

The **DK-LM3S-DRV8312** motor control kit features a 32-bit Stellaris LM3S818 microcontroller and DRV8312 motor driver to spin sub-50-V, 6.5-A, three-phase brushless DC (BLDC) motors in minutes. The easy-to-use kit is a low-cost solution for high-volume motor control applications including low-voltage fans, blowers, pumps, tools, and compressors.

The Stellaris LM3S818 controlCARD module, which is included in the kit, has the required firmware preprogrammed in flash memory to run TI's InstaSPIN-BLDC solution out-of-the-box, along with other customer-developed applications, when plugged into the DRV8312 motor driver base board. The InstaSPIN-BLDC solution is a free back-electromagnetic field (EMF) technique that spins any motor in seconds without knowledge of any motor parameters. Unlike traditional back-EMF zero-crossing techniques, InstaSPIN-BLDC extends sensorless operation down to lower speeds and exhibits high immunity to miscomputation caused by rapid speed changes. Reliable motor start-up has been demonstrated over a wide range of conditions, including full torque.



The controlCARD module with Stellaris LM3S818 MCU features 50-MHz performance, 64-KB flash memory, 8-KB SRAM, and six pulse-width modulation (PWM) generators, enabling developers to create complex motion-control designs with a small footprint.

The DRV8312 motor driver delivers up to 6.5 A current, without the need for an external heat sink. It is a reliable, fully protected motor driver with cycle-by-cycle over-current, over-temperature, cross-conduction, and under-voltage protection, reducing design complexity and board space and ensuring higher system reliability. The kit includes a 24-V, three-phase BLDC motor operating at up to 3.5 A continuous.

The supporting software includes StellarisWare software with TI's IQmath Library for easy fixed to floating-point conversion and InstaSPIN-BLDC software for quick and reliable motor start-up on any motor.

Any system design can be accelerated with an intuitive graphical user interface (GUI) powered by third-party Crosshairs Embedded. The existing GUI can be customized or created with easy drag-and-drop development by downloading Crosshairs Interface Designer.

The DK-LM3S-DRV8312 development kit costs **$299**. For those who own the DRV8312-C2-KIT, which includes the DRV8312 and C2000 Piccolo MCU, the existing controlCARD can be exchanged with the Stellaris LM3S818 controlCARD module for **$49**.

Texas Instruments, Inc.
www.ti.com

CIRCUIT CELLAR

Test Your EQ

Problem 1—In an RS-232 interface, why is the idle or "mark" level a negative voltage?

Problem 2—Similarly, why does the "mark" level correspond to a logical "high" level on the TTL side of the interface?

Problem 3—In the listing to the right, what does the C function compute? You may assume the input argument is a positive integer.

Problem 4—Suppose you are given some calibration constants for a sensor in the form of four-digit hexadecimal (16-bit) integers, and you are told that the format of these numbers is "7 integer bits and 9 fractional bits." How would you go about converting these constants to floating point so that you could, for example, work with them in a spreadsheet?

```
int function (int x)
{
   int y = 1;
   int z = 0;
   int i = 0;

   while (x > 0) {
      z += 6;
      y += z;
      x -= y;
      ++i;
   }

   return i;
}
```

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





QUESTIONS & ANSWERS



Advanced Robot Technologies

An Interview with Hanno Sander

Electronics have always captivated Hanno Sander, who began programming at an early age. His interests in science and technology led him to start his own company, which enables him to collaborate with other engineers to design robots and create development environments. In January, I interviewed Hanno about his engineering interests, book writing, and the future of artificial intelligence.—Nan Price, Associate Editor

NAN: You moved to New Zealand in 2005. Why did you end up there?

HANNO: My wife and I fell in love with the New Zealand lifestyle during our honeymoon, so we moved to Christchurch after the birth of our first child. Even with the recent earthquakes, we love living in a community that places top value on family life. We take full advantage of the beautiful beaches and scenery that New Zealand offers. It's also exciting as an engineer to be surrounded by innovative Kiwis who love to tinker. It seems like everyone here has a steam engine in their garage.

NAN: You have always had a passion for electronics. In fact, you programmed a lunar lander game for the z80 when you were just 6 years old! When did you realize you were so interested in engineering? Were you inspired by a family member or teacher when you were young?

HANNO: My mom taught high school physics and math, while my dad pioneered optical recording at Philips Electronics. From an early age, I've loved learning how things work and building new things. Growing up, I spent countless hours with Legos, Meccano erector sets, Märklin train sets, computers, and remote-controlled toys. My dad continues to be a patient mentor with a passion to distill complex concepts onto a single piece of paper.

NAN: Tell us about that game you designed at age six.

HANNO: I think I've always loved to teach people about science and technology. I wrote my Lunar Lander game after learning about the Apollo space missions to share my knowledge with my younger sisters. The game included a simulated rocket launch, a flight through a meteor field, a controlled descent onto the moon, and then a return to earth with a parachute-assisted splashdown into the ocean. This was programmed on my dad's self-built z80 complete with 1022-byte sectors read from tape, a homemade keyboard, and four potentiometers for analog control.

NAN: Where did you go to school and what did you study?

HANNO: I studied computer science at Stanford University in California. I loved learning from my professors and fellow students but had even more fun developing a hybrid car and a microsatellite. I graduated in three years to start my first company, Daptyx, an interactive online newspaper. After making some money from my start-up, I brought several advanced technology projects to market during my Silicon Valley career at Oracle, Yahoo!, and Verity.

NAN: Tell us about your company, HannoWare. Describe some of your projects.

HANNO: HannoWare is my attempt to share my hobbies with others while keeping my kids fed and wife happy. It started with me simply selling software online but is now a business developing and selling software, hardware, and courseware directly and through distributors. I get a kick out of collaborating with top engineers on our projects and



A Turing machine debugged with ViewPort

DESIGNSPARK

Deadline for Entries: March 27, 2012

Turn a hot idea into a cool solution.

DesignSpark chipKIT™ Challenge

Get ready to win your share of $10,000 in cash prizes!

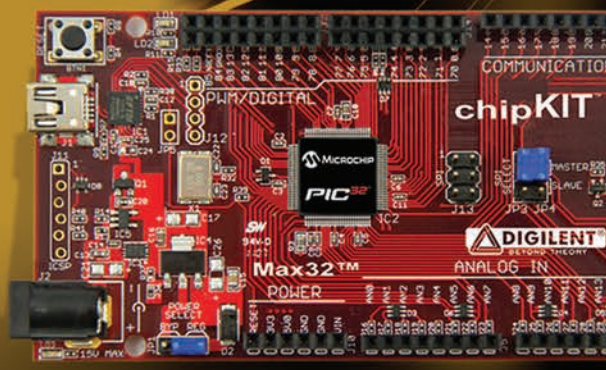
It's time to see if your hard work and superior engineering skills have paid off. The deadline for the **DesignSpark chipKIT™ Challenge** is just around the corner. It's time to finalize your design and prep your entries for the judges!

Will your design change the world? Reduce power consumption? Improve energy efficiency? There's only one way to find out.

Manage your project entry by clicking on the 'My Project' tab at **www.designspark.com/chipkitchallenge-projects/latest**. Be sure to upload, and clearly label, all materials necessary for judging your entry including an abstract, complete documentation, and source code.

For more information and tips on how to enter, visit **www.designspark.com/chipkitchallenge/faq.**

Don't delay! The DesignSpark chipKIT™ Challenge ends on March 27, 2012 at 18.00 GMT (13.00 EST).



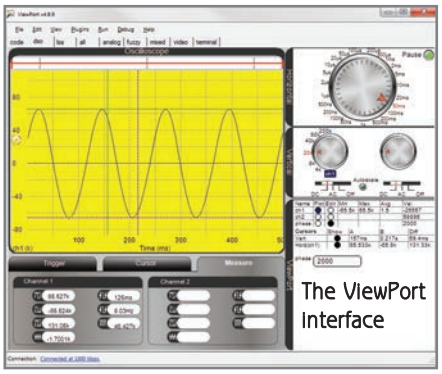
Visit www.chipkitchallenge.com
for complete rules and details.

IN ASSOCIATION WITH:

elektor · MICROCHIP · RS · CIRCUIT CELLAR · ALLIED ELECTRONICS · DIGILENT BEYOND THEORY

DesignSpark chipKIT™ Challenge

chipKIT™ is a registered trademark of Microchip Technology Inc. Max32™ is a registered trademark of Digilent, Inc.

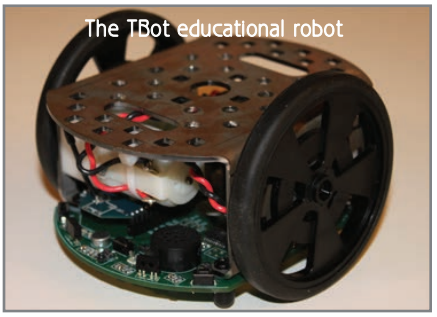


love hearing from customers about their success.

Our first product was the ViewPort development environment for the Parallax Propeller, which features both traditional tools like line-by-line stepping and breakpoints as well as real-time graphs of variables and pin I/O states to help developers debug their firmware. ViewPort has been used for applications ranging from creating a hobby Turing machine to calibrating a resolver for a 6-MW motor.

12Blocks is a visual programming language for hobby microcontrollers. The drag-n-drop style of programming with customizable blocks makes it ideal for novice programmers. Like ViewPort, 12Blocks uses rich graphics to help programmers understand what's going on inside the processor. The ability to view and edit the underlying sourcecode simplifies transition to text languages like BASIC and C when appropriate.

TBot is the result of an Internet-only collaboration with Chad George, a very talented roboticist. Our goal for the robot was to excel at typical robot challenges in its stock configuration while also allowing users to customize the platform to their needs. A full set of sensors and actuators accomplish the former while the metal frame, expansion ports, and software libraries satisfy the latter.



NAN: Tell us about your PropScope design, which is one of the projects featured on your company's website (www.hannoware.com).

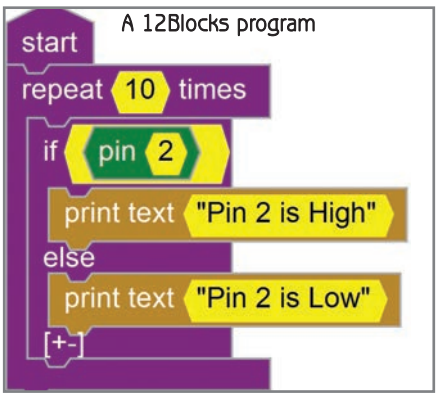
HANNO: The PropScope is a multifunction USB oscilloscope built around the Parallax Propeller microprocessor. The Propeller's eight cores allow it to sample and analyze both digital and analog signals at high speed while generating customizable waveforms. Students and engineers use the PropScope as an affordable tool to understand and share electronic signals.

NAN: You have given several "Tech Talks" at Google. Tell us a little about the talks and how you came to be involved.

HANNO: I first heard about Google when my senior project advisor at Stanford told me to talk to a couple guys who were indexing the Internet—sadly, I didn't follow his advice. I now make an annual pilgrimage to the Googleplex to lunch with friends, marvel at a SpaceShipOne replica, and talk about the projects I'm involved with. The Tech Talks are filmed on an elaborate stage with a slew of robotic cameras and lights resulting in a quality production.

NAN: In addition to designing and presenting, you've also done a fair amount of writing, including an article for *Circuit Cellar* and a soon-to-be published book about advanced robotics. In your 2009 article, "Vision-Guided Robotics: A Next-Generation Balancing Robot" (*Circuit Cellar*, 224), you introduced the DanceBot, a balancing robot you built with a Parallax Propeller, a handy design kit, and a camera. Tell us how the DanceBot has evolved since then.

HANNO: I developed the DanceBot right when my son was learning to walk, so I challenged myself to get my robot to take a step before my son. The DanceBot did win, but my son has evolved much further since. I've been reluctant to bring a vision-guided



balancing robot to market in this economy, but it's still on my to-do list. I did contribute several chapters about the DanceBot to another book, *Programming and Customizing the Multicore Propeller Microcontroller: The Official Guide*, and the hardware and software are available as part of the open-source Propeller Smorgasboard project.

NAN: Your upcoming *Circuit Cellar* book covers all things robotics, from microcontrollers and peripherals to visual debugging and test instruments. The book is designed for DIYers, computer science students, and professional engineers alike. What was your impetus for writing the book? What can readers can expect to learn from it?

HANNO: There are plenty of books that introduce people to robotics. I wrote my book to appeal to those looking for more meat on more advanced subjects like computer vision, control



algorithms, and wireless control. My goal with each topic was to provide some technical background, code, and schematics to explain the concepts and then discuss applications and best practices.



The Parallax PropScope

NAN: Why such a strong interest in robotics?

HANNO: I think robots are a wonderful tool to help people relate to the abstract concepts inherent in science, technology, engineering, and math (STEM) subjects. Since the industrial revolution, people have fantasized about interacting with mechanical creatures and we're lucky enough to live in the age where they're becoming real.

NAN: Aside from advanced robotics, what other electronics engineering topics interest you and why?

HANNO: I'm a big advocate of science and technology and love learning about new developments in both. I'm currently excited about the rapidly decreasing cost of photovoltaic power, applying the storage capacity of lithium batteries to revolutionize existing products, and the flexibility of additive manufacturing (e.g., 3-D printers).

NAN: We often ask interviewees about their predictions for upcoming "hot topics." Some recent answers have included: DSP and compressive sensing, smaller and faster CPUs, and hardware design using programmable logic. What are your thoughts?

HANNO: Today's computing devices feature incredible horsepower, memory, and connectivity to basically all of human knowledge in addition to high-definition cameras and other sensors. We're starting to see artificial intelligence in commercial offerings with IBM's Watson and Apple's Siri. I think that, for better or worse, the trend over the next decade is to bring human-level intelligence to devices all around us.

NAN: Final question: If you had a full year and a good budget to work on any design project you wanted, what would you build?

HANNO: I like to think that HannoWare gives me the flexibility to pursue my passion. I'm currently very excited to work with teachers from kindergarten through university on a set of videos, example programs, and supporting resources that conform to New Zealand's curriculum standards. We've set up the OneRobot organization to help typical schools to enhance their offering with resources that have been proven to get results.





by Larry Foltzer (USA)

Robot Navigation

Position Determination from Acoustic Delay Triangulation

Robot navigation doesn't have to be complicated to be effective. You can implement the mathematics of navigation described here to chart a course to a target destination while staying within the native bounds of the 8-bit microcontroller and 8-bit mathematics.

Autonomous robot behavior has interested me for years, primarily as an observer. But prior to retirement, I had little time outside my profession to explore extracurricular interests. After retiring, and about two years ago, I decided to explore what it takes, algorithmically speaking, to make a robot that is capable of discovering its position on a playing field and figuring out how to maneuver to another position within the defined field of play. Later on I will build a minimalist-like platform to test algorithms performance.

In the interest of hardware simplicity, my goal is to use as few sensors as possible. I will use ultrasonic sensors to determine range to ultrasonic beacons located at the corners of the playing field and wheel-rotation sensors to

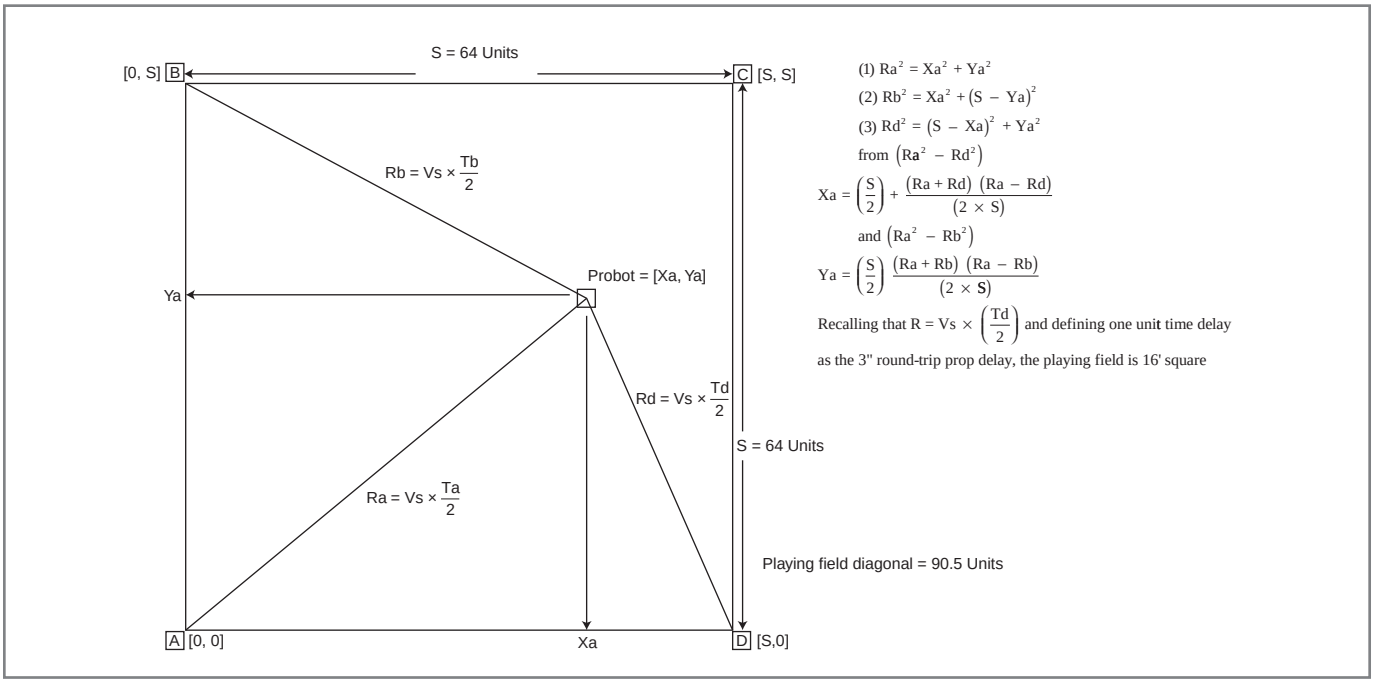


Figure 1—The square playing field

measure distance traversed, if wheel-rotation rate times time proves to be unreliable.

From a software point of view, the machine must be able to determine robot position on a defined playing field, determine robot position relative to the target's position, determine robot orientation or heading, calculate robot course change to approach target position, and periodically update current position and distance to the target. Because of my familiarity with Microchip Technology's 8-bit microcontrollers and instruction sets, the PIC16F627A is my choice for the microcontrollers (mostly because I have them in my inventory).

To this date, the four goals listed—in terms of algorithm development and code—are complete and are the main subjects of this article. Going forward, focus must now shift to the hardware side, including software integration to test beyond pure simulation.

SENSOR TECHNOLOGY & THE PLAYING FIELD

A brief survey of ultrasonic ranging sensors indicates that most commercially available units have a range capability of 20′ or less. This is for a sensor type that detects the echo of its own emission. However, in this case, the robot's sensor will not have to detect its own echoes, but will instead receive the response to its query from an addressable beacon that acts like an active mirror. For navigation purposes, these mirrors are located at three of the four corners of the playing field. By using active mirrors or beacons, received signal strength will be significantly greater than in the usual echo ranging situation. Further, the use of the active mirror approach to ranging should enable expansion of the effective width of the sensor's beam to increase the sensor's effective field of view, reducing cost and complexity.

Taking the former into account, I decided the size of the playing field will be 16′ on a side and subdivided into 3″ squares forming an (S × S) = (64 × 64) = (2^6, 2^6) unit grid. I selected this size to simplify the binary arithmetic used in the calculations. For the purpose of illustration here, the target is considered to be at the center of the playing field, but it could very well be anywhere within the defined boundaries of the playing field.

ECHOES TO POSITION VECTORS

Referring to Figure 1, the corners of the square playing field are labeled in clockwise order from A to D. Ultrasonic sonar transceiver beacons/active mirrors are placed at three of the corners of the playing field, at the corners marked A, B, and D.

The beacons are unique in that they have an assigned name or address, and, as slaves, only speak when spoken to. The robot is the system master and polls the beacons in turn, then measures the delay between the poll and the corresponding response. For each beacon's poll/response delay count, a propagation-delay arc is constructed from their vertices where the point of intersection of the arcs marks the robot's position.

The distance between the robot and each of the beacons is determined by measuring the extended propagation delay of the ultrasonic signal in air, which is roughly 150 µs per inch, round trip time. (Note: Extended/padded, round-trip propagation delay = Tprop + 50 ms. Having the beacon wait 50 ms to issue a response enables "real," direct playing field echoes to decay. After 50 ms, received echoes should all be beacon responses.) The lengths of the adjacent radius vectors are Ra, Rb, and Rd. Assuming the origin of the grid to be coincident with corner A of the playing field, vector Ra terminates on the robot's position (X, Y) at a distance from the origin, courtesy Pythagoras.

$$Ra = \sqrt{(X^2 + Y^2)}$$

However, from beacon B's perspective, the position of the robot is (X, S – Y), making the length:

$$Rb = \sqrt{\{X^2 + (S - Y)^2\}}$$

Similarly, from beacon D's perspective, the robot's position is at (S – X, Y) at a distance of:

$$Rd = \sqrt{\{(S - X)^2 + Y^2\}}$$

Working with the squares of the 'R'adius vectors and solving for X and Y we get:

$$X = \frac{S}{2} + \frac{(Ra^2 - Rd^2)}{2 \times S}$$

$$Y = \frac{S}{2} + \frac{(Ra^2 - Rb^2)}{2 \times S}$$

$$X = 32 + \left[\frac{\{(Ra + Rd)\ (Ra - Rd)\}}{128}\right]$$

$$Y = 32 + \left[\frac{\{(Ra + Rb)\ (Ra - Rb)\}}{128}\right]$$

Note that the numerator of the second terms in the above equations are the "difference of two squares," and can be factored into sum and difference terms: (Ra + Rb) (Ra – Rb) = $Ra^2 - Rb^2$. Also note the possible range of values for these terms on our 64 × 64 playing field. For example, if the robot is in close proximity to position "C" [coordinates (S, S)], then:

$$Ra = \sqrt{(2 \times S^2)} = 91$$

Rb and Rd = 64. This makes (Ra – Rb) = 27 and (Ra + Rb) = 155. These values are easily accommodated in a single byte, and their 16-bit product can be computed using a simple 8-bit × 8-bit multiplication routine. The denominator of the second terms is fixed and equal to 128 (0x80). Division by 128 using binary arithmetic is accomplished either by performing seven right shifts of the second term's numerator, or by a single roll-left-file (RLF)

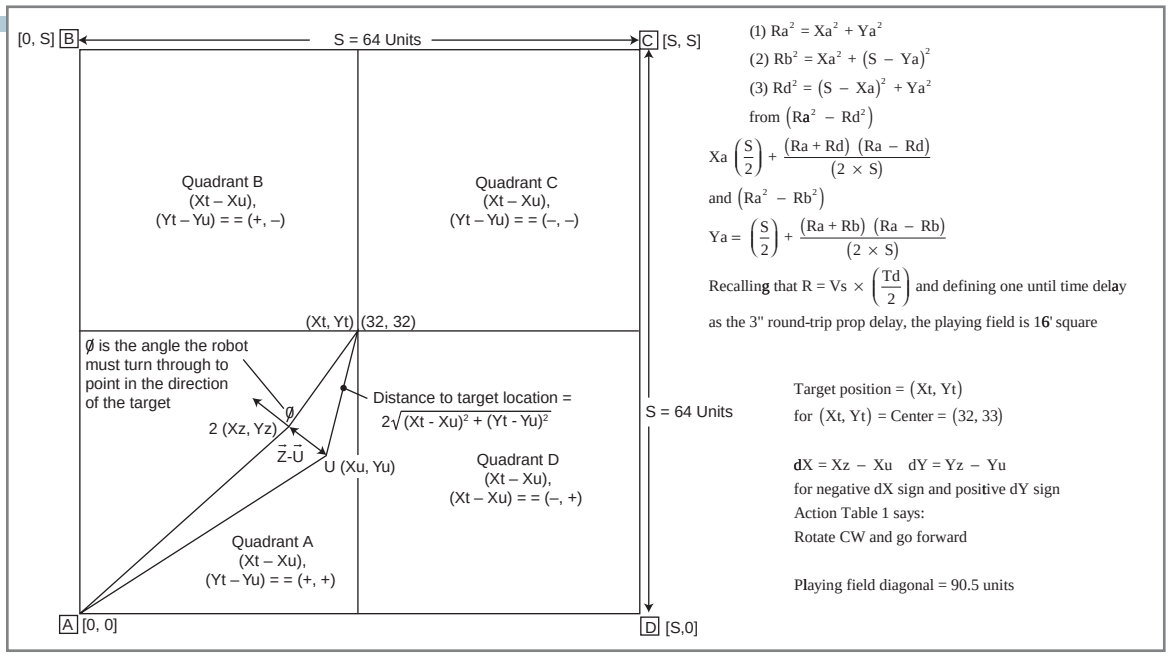


Figure 2—The quadrants of the square playing field

instruction of the numerator:

$$\text{For Ra} = 77 \text{ and Rd} = 67\text{: X} = 32 + \frac{1{,}440}{128} = 43$$

$$\text{or X(Hex)} = 0\text{x}0020 + \frac{0\text{x}05\text{A}0}{0\text{x}80}$$

$$\frac{0\text{x}05\text{A}0}{0\text{x}80} = 0\text{x}05\text{A}0 >>>>>>> 0\text{x}000\text{B} = 11 \text{ (base 10)}$$

$$\text{or } 0\text{x}05\text{A}0 < 0\text{x}0\text{B}40 = 11\ 64$$

$$\text{and } 32 + 11 = 43$$

In the second case, the second-term numerator is left shifted once, changing 0x05A0 into 0x0B40, with the integer part of the result in the most-significant bit (MSb) and the fractional part of the division in the least-significant bit (LSb). If bit 7 of the 16-bit result is set, add one (1) to the integer value.

Repeat the process using Ra and Rb, and calculate the value of Y. The robot's position, relative to the playing field origin at the location of beacon A, may be calculated. Furthermore, with prior knowledge of the target position coordinates, we can compute the distance to the target, but I am getting ahead of myself.

DETERMINING ROBOT HEADING

Having determined the robot's current position at point U (Xu, Yu), we know in which quadrant the robot resides. However, being blind, we do not know in which direction the robot is pointing, so we must move the robot to a second point to find out. We call this second point Z (Xz, Yz) and see how our position coordinates change as a result. In the real world, one would consult one or more sensors to see if the coast is clear before moving into harm's way and perhaps falling off the edge of mother earth. But that's another story.

By taking the difference between the coordinates of Z and U, we can determine the robot's general direction from the signs of X = (Xz – Xu) and Y = (Yz – Yu) and more precisely from the magnitude of the slope of the robot's current heading. The resulting change in position may be interpreted as a velocity vector (H), containing both speed and direction information. At this point, however, we are only interested in the directional information (slope) contained in the vector:

$$H(\Delta X, \Delta Y) = V - U = \left[(Xv - Xu) + j\,(Yv - Yu)\right]$$

$$H(\text{slope}) = \frac{\Delta Y}{\Delta X} = \frac{(Yv - Yu)}{(Xv - Xu)}$$

In support of further discussion, we label the quadrants

Quadrant	Sign ΔX	Sign ΔY	Action
A	+	+	Go straight
	+	–	Rotate CCW and go
	–	+	Rotate CW and go
	–	–	Back up and rotate CW
B	+	+	Rotate CW and go
	+	–	Go straight
	–	+	Back up and rotate CCW
	–	–	Rotate CCW and go
C	+	+	Back up and rotate CCW
	+	–	Rotate CW and go
	–	+	Rotate CCW and go
	–	–	Go straight
D	+	+	Rotate CCW and go
	+	–	Back up and rotate CW
	–	+	Go straight
	–	–	Rotate CW and go

Table 1—Position and heading-dependent action table

Bit #	Fractional Value	Cos(Angle)	Cos(Angle)	Angle	Sin(Angle)
7	0.5	255/256	0.9961	5.065929	0.088302
6	0.25	254/256	0.9922	7.166643	0.124756
5	0.125	253/256	0.9883	8.780177	0.152644
4	0.0625	252/256	0.9844	10.14179	0.176085
3	0.03125	251/256	0.9805	11.34258	0.196675
2	0.015625	250/256	0.9766	12.42926	0.215234
1	0.0078125	249/256	0.9727	13.42954	0.232249
0	0.00390625	248/256	0.9688	14.36151	0.248039
		247/256	0.9648	15.2377	0.262824
sum	0.99609375	246/256	0.9609	16.06725	0.276765
	= cos(5.066)	245/256	0.957	16.85704	0.289985

Table 2—Fractional value of a byte and sine and cosine of binary angles

of the playing field A, B, C, and D in accordance with the corner references used in previous discussion (see Figure 2). If the robot finds itself in quadrant A, it knows it must move in the general northeast direction to close the distance to the "targeted" center of the field, at coordinate (32, 32).

Table 1 defines, in a general way, the next course of action based on the signs of ΔX and ΔY. We use this sign information later on in the test code to set or clear a direction flag bit that is used to select between CW or CCW equation in the code.

Note that three of the four prescribed actions in each quadrant include a rotation of the platform. This gives rise to the question: how much rotation is required? To answer that question, one needs to calculate the angle between the robot's current heading and a vector from the robot's position to the target position. At first, I considered using the dot-product method, but I discovered that I would need an arc-cosine look-up table and an interpolation routine to calculate the turn angle. I hate look-up table solutions except where there is no other solution; therefore, I looked into other solutions, and once again Euler came to the rescue!

Manipulating Euler's exponentials one comes up with Equations 1 and 2 below, which calculate the components of a new heading vector, in terms of the components of

AP CIRCUITS
PCB Fabrication Since 1984

As low as...
$9.95
each!

Two Boards
Two Layers
Two Masks
One Legend

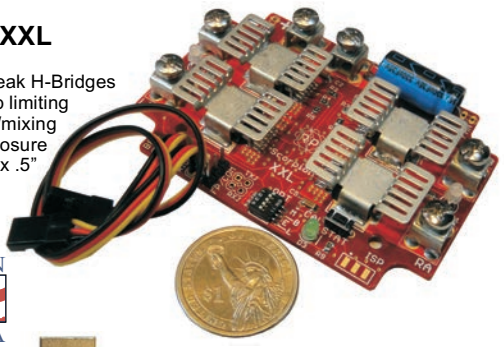
Unmasked boards ship next day!

www.apcircuits.com

VISA MasterCard PayPal IPC MEMBER ASSOCIATION CONNECTING ELECTRONICS INDUSTRIES BBB

ROBOT POWER™
Extreme Motor Speed Control!

Scorpion XXL
♦ 7V - 28V
♦ Dual 40A+ Peak H-Bridges
♦ Current/Temp limiting
♦ R/C inputs w/mixing
♦ Optional enclosure
♦ 3.25" x 2.25" x .5"



MADE IN THE USA



MegaMoto Plus Shield
♦ 7V-28V - H-bridge or Dual Half-bridge
♦ 20A Continuous - 40A peak
♦ Current sensor outputs can be sent to any analog pin
♦ Up to three units can be stacked on one Uno or compatible
♦ Independent half-bridges can control brushless/steppers too!

Check out our other motor controls 2A - 160A

www.robotpower.com
Phone: 253-843-2504 ♦ sales@robotpower.com

the robot's current heading vector, the desired direction and angle of rotation.

$$R \times \cos(\theta \pm \theta) = X(n+1) = X(n) \times \cos(\theta) \pm Y(n) \times \sin(\theta) \quad [1]$$

$$R \times \sin(\phi \pm \theta) = Y(n+1) = Y(n) \times \cos(\theta) \pm X(n) \times \sin(\theta) \quad [2]$$

Unfortunately, we do not know the total angle of rotation. However, we do know what our heading should be to match the course to the target. And as luck will have it, we can use Equations 1 and 2 in an iterative process to calculate the angle to the target destination. (Note the use of ± in Equations 1 and 2.)

When computing the cos(Φ + Θ) and sin(Φ + Θ), we are talking about a counter-clockwise rotation of the robot heading vector by Θ degrees. Furthermore, cos(Φ – Θ) and sin(Φ – Θ) performs a clockwise rotation of the heading vector by Θ degrees. Table 2 is used to select the most direct direction of rotation.

To use Equations 1 and 2 iteratively, we need to select a suitably small and fixed angle that gives reasonable angular resolution and whose binary values are consistent with an 8-bit solution. Luckily, by interpreting a byte's contents as a fraction, we discover that we can obtain a resolution of ±5°.

We are now in a position to look at the quality of the results of our algorithm, as implemented in machine language, and relative to a floating-point solution performed in an Excel spreadsheet. The machine code, which can be found on *Circuit Cellar*'s FTP site, was tested using the Microchip IDE code simulator, and is the source of the data used in the plot of Figure 3. Figure 3 contains three plots, the reference trace being the upper-most floating-point plot delineated by the diamond data points. All three plots start from coordinate (20, –10). The plot directly below the reference shows the results taken from the machine code for an increasing step-angle (CCW) of 10°. The lower plot is again from the machine code, but for a decreasing step angle, CW. The magnitude radial distance from the center of the plot for the floating-point solution is very stable, as expected. The magnitude of the machine code solution spirals inward in the CCW direction. Nevertheless, the angular progression of the vector is quite regular, which is the most important feature since it is the heading or slope of the destination vector that matters here.

VECTOR PARALLELISM TEST

The test for vector parallelism uses Equation 3 as a zero-crossing detector.

$$(\Delta Ytarget \times \Delta Xrobot) - (\Delta Yrobot \times \Delta Xtarget) = 0 \quad [3]$$

where $\Delta Ytarget = Ytarget - Yrobot$,
$\Delta Xtarget = Xtarget - Xrobot$,
$\Delta Yrobot = Yrobot2 - Yrobot1$,
and $\Delta Xrobot = Xrobot2 - Xrobot1$

We exit the iteration process on a change of sign of the indicated sum. Upon completion of the iterative calculations, the angle through which the robot must turn to align its heading with the target destination vector is the product of angular step size used in the calculations and the iteration count. It is now up to the robot hardware to effect the required change of course and make

> "To keep things simple on the sensor front, it is best if the robot has an omnidirectional ultrasonic sensor to communicate with the system beacons. In this way, the robot need not be concerned with having to find, and then point at, the appropriate beacon before it can issue a robust poll of the beacon for ranging purposes. One way—the expensive way—is to use multiple ultrasonic sensors in a circular array to achieve coverage. "

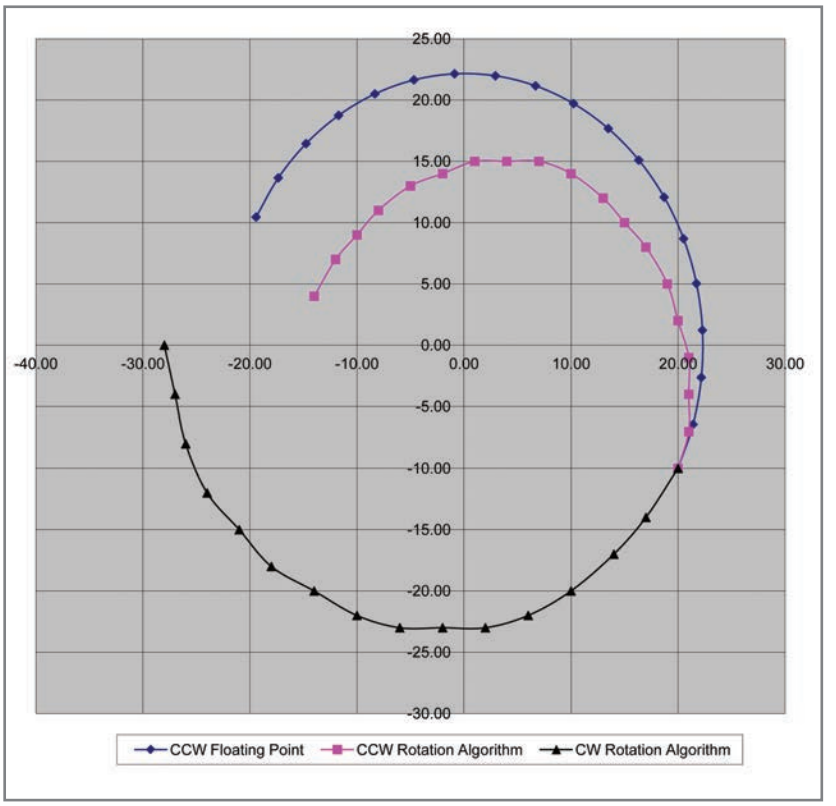


Figure 3—The 10° step size, floating point versus machine language algorithm

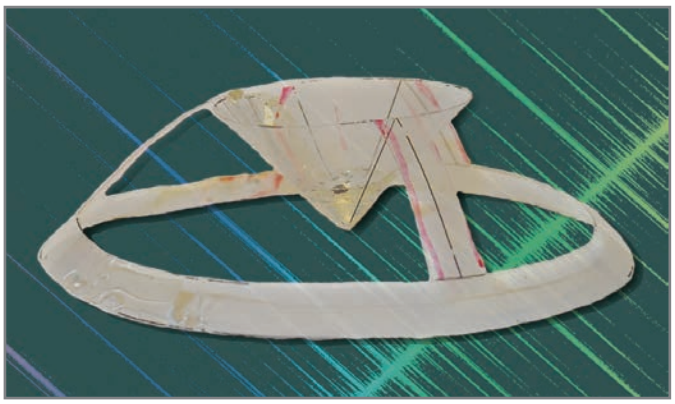


Photo 1—A conical acoustic mirror

its way to the target.

The code on *Circuit Cellar*'s FTP site does not perform the zero-crossing function. That task is a simple process that is left to you.

ALGORITHM TEST CODE

The code on the FTP site is for your enjoyment. It was written and simulated using Microchip's IDE for the PIC12F609 8-bit microcontroller. The selection of a processor for testing was arbitrary and simply to ensure it was code-compatible with other microcontrollers considered for this application.

I placed the variables in the GPR space so they show up in an orderly place when displayed in the simulator window. Furthermore, I broke up the code into functional blocks to aid readability. All is displayed in a two's complement in the simulator.

ULTRASONIC SENSOR MODIFICATIONS

More "work in progress"? To keep things simple on the sensor front, it is best if the robot has an omnidirectional ultrasonic sensor to communicate with the system beacons. In this way, the robot need not be concerned with having to find, and then point at, the appropriate beacon before it can issue a robust poll of the beacon for ranging purposes. One way—the expensive way—is to use multiple ultrasonic sensors in a circular array to achieve coverage. An alternative is to use a conical acoustic mirror (see Photo 1). Will it work? I don't know yet, but do know there is no free lunch. Spreading the beam will decrease transmitted power density. However, that was the reason to go with an active mirror concept in the first place, where signals are unidirectional and are not subject to the loss associated with the albedo of the target.

LEARN, BUILD & PROSPER

Well, there you have it. There is nothing new here that has not been done before. However, the application of the mathematics and its implementation in machine code to robotics was new and a fun thing to do. And "I did it my way." I hope newcomers to the field find this work useful enough to build upon it and prosper!

Larry Foltzer (lefoltzer@dishmail.net) was employed for 30 years in the fiber optics communication industry. First, he cofounded Optelecom, Inc. A decade later, his focus shifted to data and telecommunications. For Motorola, he participated in the IEEE-802 MAC subcommittee on Token-Passing access control methods. Later, Larry was the System Architect of the Raychem/Raynet Passive Optical Network System.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/260.

SOURCES

PIC16Fxx Family of microcontrollers, IDE code simulator, and PIC12F609 microcontroller
Microchip Technology, Inc. | www.microchip.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*. For more need-to-know information about some of the topics covered in this article, the *Circuit Cellar* editorial staff recommends the following content:

—

A Sensor System for Robotics Applications
by Guido Ottaviani
***Circuit Cellar* 236, 2010**
This article covers how to add "senses" to an interactive robotics platform, from sight to hearing to touch and more. Topics: Robotics, Sensors, ADC, Arduino I²C, Ultrasonic Photoresistor

—

Vision-Guided Robotics
A Next-Generation Balancing Robot
by Hanno Sander
***Circuit Cellar* 224, 2009**
Are you interested in building a sophisticated, vision-guided, balancing robot that can interact with its environment? You can tackle this project with a Parallax Propeller, a handy design kit, and an inexpensive camera. Topics: Robotics, Vision, Fuzzy Logic, Propeller, ViewPort

—

Robot Localization and Control
by Ethan Leland, Kipp Bradford, and Odest Chadwicke Jenkins
***Circuit Cellar* 188, 2006**
This team shows you how to use wireless nodes to simultaneously localize and control a WowWee Robosapien humanoid robot. The ZigBee nodes' outputs mimic the control signals. Topics: Robot, Wireless, ZigBee, GUI

—

Go to *Circuit Cellar*'s webshop to find these articles and more: www.cc-webshop.com

by Ian Bromley

Last Chance to Enter the DesignSpark chipKIT™ Challenge

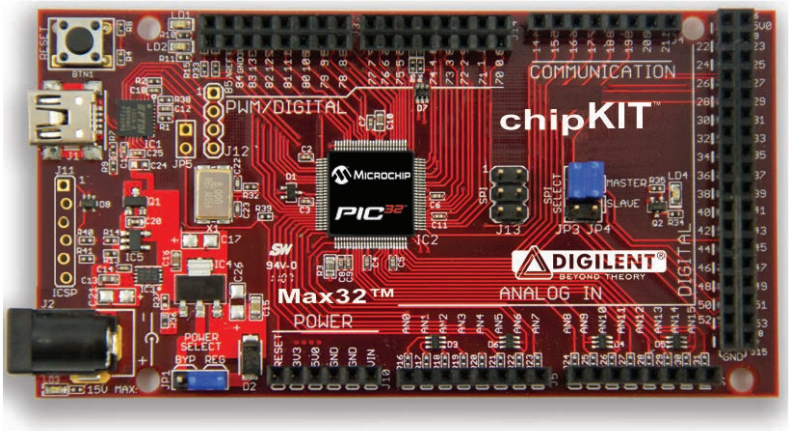
Now in its final month, the DesignSpark chipKIT™ design challenge for energy-efficient applications is still open for submissions. So, if you're quick,…

The DesignSpark chipKIT™ competition is now in its final weeks, and there remains just enough time to enter for a chance to win a prize. Total cash prizes of $10,000 are there to be won, including a first prize of $5,000. The response and enthusiasm for the competition has been fantastic and we've been delighted by the high quality of the submitted ideas. As a reminder of what has gone before, or for those hearing about this for the first time, the DesignSpark chipKIT™ challenge is all about encouraging engineers, students, and hobbyists to develop new and innovative energy-efficient solutions, while also maintaining an eco-friendly footprint.

So, as the competition approaches its climax, entrants will now be well advanced in their designs for energy-efficient and environmentally friendly applications based on the chipKIT™ Max32™ Arduino-compatible development platform from Digilent. The kit features Microchip Technology's 32-bit PIC32 microcontroller and enables developers to easily and inexpensively integrate electronics into their projects. Just to whet your appetite for the smorgasbord of ideas we've had, here are just a few of them we've received—which are also available at the online DesignSpark community at www.designspark.com.

One very interesting project is the development of an unmanned underwater vehicle (UUV) that employs gliding as its method of propulsion, enabling the UUV to operate for an extended period of time due to its extremely low power consumption. An extender board being developed for the vehicle will interface to the sensors and actuators required for operation of the UUV. The design will include a three-axis accelerometer, a three-axis gyroscope, and a three-axis magnetometer for underwater orientation, in addition to a global positioning system (GPS) receiver to determine global position on the surface. A buoyancy engine, together with pitch and roll actuators, will control the UUV's movement through the water. In addition CTD sensor will measure conductivity, temperature and depth, with the data being recorded in flash memory on board the UUV, enabling profiling of salinity characteristics.

Another project is a hydroponics water and nutrient-control system that will monitor climatic conditions to determine the level of nutrient-feed requirements for plants in a hydroponic environment, thereby saving on water usage and nutrient chemicals. As most hydroponics farms currently use established feed regimes that run without adjustment for environmental conditions, the design could deliver both financial and water savings.



chipKIT™ Max32™ Development Board

A "smart garden" project for the control of the natural cycle of a plant with minimal human intervention is yet another innovative entry. The project aims to combine renewable energy sources—such as solar power—with low energy consumption of the board control circuitry operating at significantly reduced power. The system will include various sensors such as those for humidity, temperature, and light, and actuators such as an irrigation pump. Additionally, the modular system will also be able to be expanded, for example, adding Bluetooth technology to enable user notifications—a possibility created by the ongoing "Internet of Things" revolution—about which we expect to be hearing a great deal more this year.

And these are just three of many highly innovative ongoing projects. As a quick final reminder on the competition, all entries must include an extension card developed using RS's free-of-charge and award-winning DesignSpark printed circuit board (PCB) software tool with code compiled using Microchip's MPLAB® IDE software. Entrants are also strongly encouraged to engage and interact with other members of the online DesignSpark community by posting information on their projects, providing updates on progress, and sharing comments and ideas on their respective designs. Participants will automatically qualify for entry into bonus Community Choice Awards, in addition to admission into spot prize draws for the best collaboration to win vouchers exchangeable for products ordered from RS Components/Allied Electronics.

The competition entries will be judged on the level of energy efficiency and the quality of the extension card's PCB design. Entries are due on March 27, 2012 and the winners will be announced in April 2012. Further details and registration for the DesignSpark chipKIT™ challenge are available at: chipkitchallenge.com.

Ian Bromley is a Technical Marketing Engineer at RS Components and the Project Manager for the DesignSpark PCB software tool. Prior to working for RS, Ian worked for many years as a design support consultant with Texas Instruments, in addition to working as a field applications engineer immediately following his graduation in 1994 with an honors degree in Microelectronic Engineering.



To register and enter the DesignSpark chipKIT™ Challenge, go to: www.chipkitchallenge.com



by Chris Cantrell (USA)

All-in-One Joystick

Build a Propeller-Based TV Gaming Platform

The only thing more exciting than gaming is gaming on a system you develop. Ready to hack an old joystick? You can remove the circuit board and add a custom game system that reads games from an SD card. This article details how to create a virtual machine in a Propeller chip for the arcade hardware of a classic game.

The middle schoolers in my carpool asked how big my cell phone was when I was their age. Even in their 12 short years, they have seen how fast technology grows. I told them, when I was 12, there was no such thing as a cell phone. Now I had their attention. What did we have?

We kids had pretty much the same kind of things as kids today, just less powerful. We had phones, but they were wired to a wall in the kitchen. We had game consoles that hooked up to the television. They were not high-definition or online, and they used plastic cartridges instead of CDs, but the principle was the same. We had arcades with lots of noisy flashing video games that would hypnotize you into dribbling away your allowance one quarter at a time. Now the classic machines huddle together behind the skee-ball tables in restaurants where animatronic mice hypnotize you into buying pizza. But the classics are still there. Nothing ever dies in this age of eBay, Wikipedia, and YouTube.

The big difference is the size and cost. The same TV game console I played as a kid now fits inside its joystick along with a dozen of the best games. You can buy the



Photo 1—This is the complete system in action. The portable DVD/TV display is turned sideways to match the rotation of the arcade monitor in the real system.

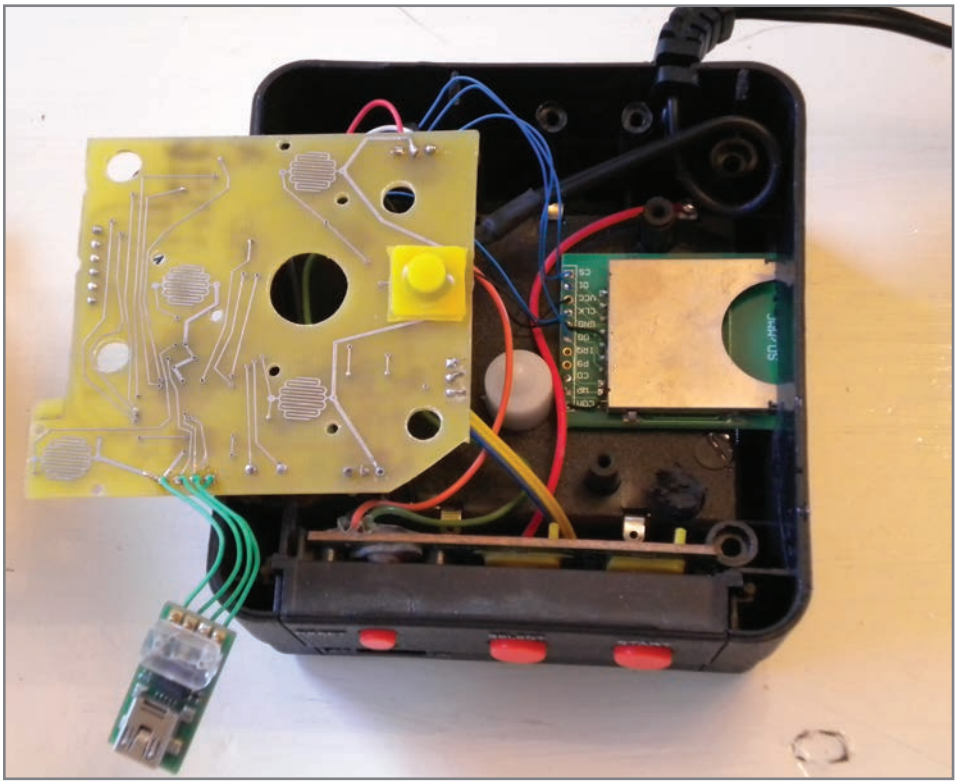


Photo 2—The replacement circuit board has five switch pads. One yellow conductive button is shown mounted. This yellow casing acts like a spring for the conductive disc inside. The circuit components are soldered to the surface of the other side of the board.

Jakks Atari Classics joystick for the same price you would have paid for a single Atari 2600 game cartridge back in the day.

Jakks Pacific also makes several arcade collection joysticks. Each has a different handful of classic machines shrunk down into a joystick. You can buy one from the web for the same roll of quarters you would carry to the arcade. What wonderful times we live in!

What Jakks doesn't make is a joystick with an SD card slot for switching games. They don't own the rights to the old classic games, after all. They own their hardware and they are in the business of selling plastic joysticks—not SD cards.

But we electronics enthusiasts have shrunken inexpensive electronics in our arsenal too! In this article I'll show you how I replaced the circuit board in a Jakks joystick with my own Parallax Propeller-based game system that reads games from an SD card (see Photo 1). I'll also show you how I created a virtual machine in the Propeller for the arcade hardware of the classic Space Invaders.

HACKING THE ATARI JOYSTICK

I chose the Atari 10-in-1 joystick because of its bland casing. All the other game sticks I have seen have colorful images for the games they contain. The Atari game stick is just a generic black joystick with a few orange buttons.

There are two circuit boards inside the stick. A small board at the front of the stick provides contacts for the three round buttons labeled Reset, Select, and Start. I left this board in place but cut traces and rerouted wires so that one side of each switch is grounded to the negative terminal of the 6-V battery pack. I left the power switch on the front of the joystick intact, but I rewired the tiny red power LED as a general-purpose output to be controlled by a Propeller I/O pin. Thus, the hacked mini board provides six wires to the main board: 6 V, ground, an LED output, and three switch inputs.

I pulled the main board out and used it as a template for a new circuit board (see Photo 2). The main board has five contact pads for the five joystick controls (left, right, up, down, and fire). Each pad is a circular area with two traces that weave around one another without actually touching. A conductive foam disc encased in a springy yellow case mounts above each pad. When the button is pressed, the disc presses against the circuit board and shorts the two woven traces together. When the button is released, the yellow casing pulls the disc up and breaks the contact.

The joystick shaft sits above the four directional buttons. Mechanical linkages push the buttons down when the joystick is moved in the four directions.

I carefully measured all the screw holes, pads, and mounting holes and transferred them to my circuit board using the free ExpressPCB software. I went through dozens of tedious print, compare, and tweak cycles. But I ended up with a board that matched the original circuit board's pad and mounting configuration.

THE PROPELLER CIRCUIT

I used the same basic Propeller processor circuit in this project that I have used in many others (see Figure 1). The Propeller chip requires very few external components: a serial EEPROM, a crystal, and a pull-up resistor. Another four resistors are all you need for complete NTSC video and audio. Each switch from the joystick is grounded on one contact and pulled up to 3.3 V with a 10-kΩ resistor at the Propeller's port pin.

I used an SD-MMC card breakout board from SparkFun Electronics. There is just enough room in the joystick under the main board for the breakout board. (Refer to Photo 2.) I cut a slot in the side of the joystick and glued the breakout board to the top of the battery compartment. I cut another slot in the other side of the casing for

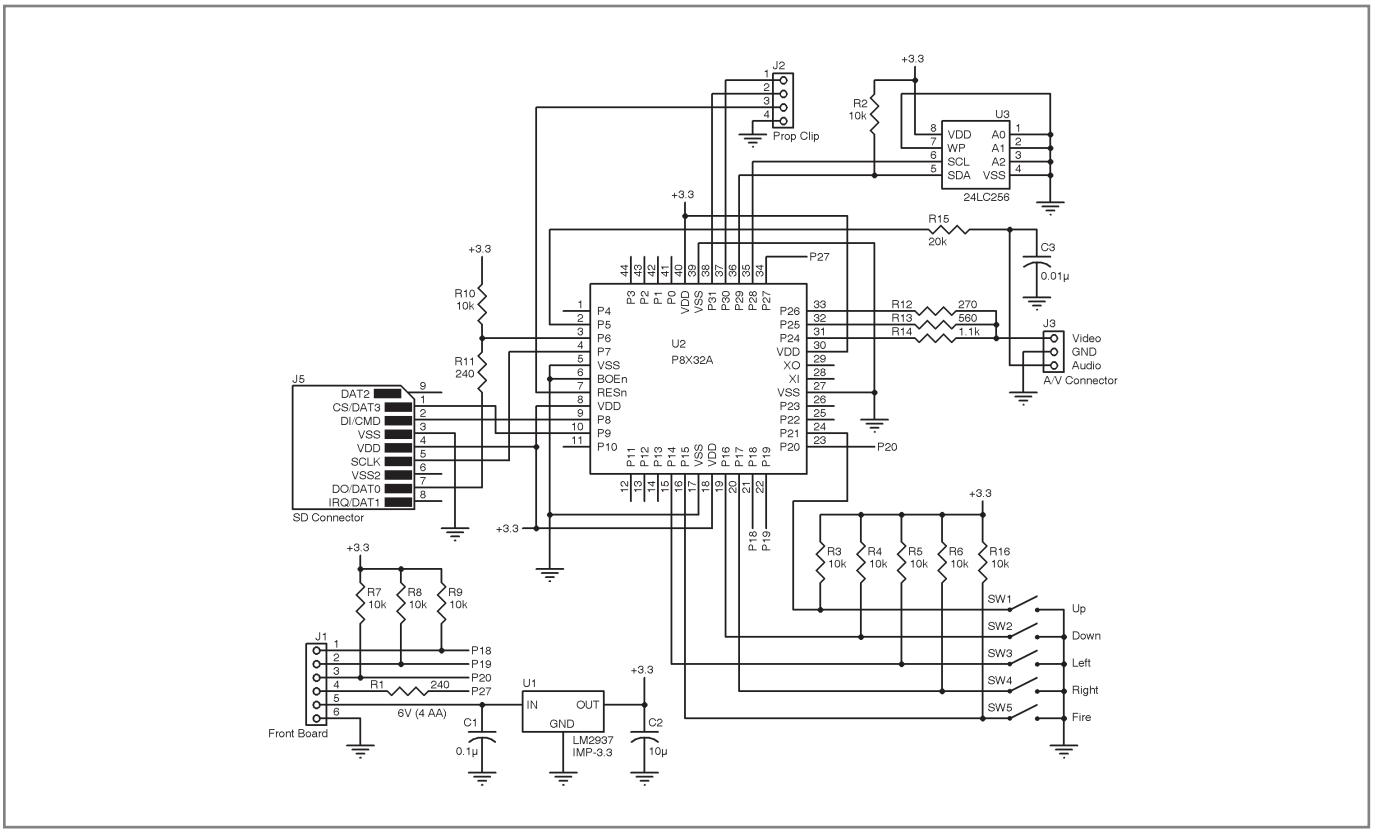


Figure 1—The core of this circuit is straight off the Parallax Propeller demo board schematic. I have built this old friend in several projects, but this time I built it on a 2" square board with surface-mount components.

the four-wire programming interface.

The real challenge was soldering the tiny surface-mount components, especially the 44-pin QFN Propeller chip. I bought a magnifying lamp with an adjustable arm from an office supply store. I also bought a fresh soldering iron tip and a set of precision tweezers/probes from the local electronics store. Then I watched a few SMT tutorials on YouTube and began what felt like performing brain surgery.

EMULATING THE ARCADE HARDWARE

The Space Invaders arcade game released by Taito Corp. in 1978 is surprisingly simple. The memory map is split between 8-KB ROM and 8-KB RAM. Seven of the 8 KB of RAM are tied up in the visual raster buffer. The video generator, implemented in discrete 74*xx* logic, interrupts the processor twice per video frame: once at the end of the screen and once in the middle. There are three input ports for reading switches and DIP settings. There is an output port that resets a watchdog timer (WDT). There are 10 output port bits that trigger 10 distinct sound effects implemented with specialized analog circuits. There is a 16-bit shift register mapped into port space that helps draw the large number of game sprites.

The entire 16-KB RAM and ROM of Space Invaders fits neatly in just half of the Propeller's shared memory. The boot process (Boot.spin) loads the ROM from an SD card and launches the other COGs I will discuss later. The project files for this article, which are available on *Circuit Cellar*'s FTP site, include all the Spin code I used in the joystick, but they do not contain the Space Invaders ROM. You must own the original hardware in order to legally download the ROM from the web. I bought a board set for $50 on eBay years ago. The classics still pop up there from time to time.

There are several SD card projects available for download from the Propeller object exchange on the Parallax site. I have used this same driver (Disk_hdw.spin) in several projects now. It is an early derivation from Tomas Rokicki's work.

VIDEO (SIVideo.spin)

The Space Invaders hardware produces a 256 × 224 pixel black-and-white (actually black and CRT green) display. Invader machines released in the U.S. used color transparencies to tint areas of the display, but the pixels themselves were either on or off.

The Propeller chip has built-in hardware to generate NTSC or PAL composite video. The "TV.spin" object in the Propeller library generates four-color pixels on a 256 × 208 display. My "SIVideo.spin" is a modification of "TV.spin" that generates a black-and-white display from the Space Invaders raster format. The driver spends most of its time "doubling" pixels so that "0" from the memory buffer becomes "00" going to the "waitvid" Propeller instruction and "1" becomes "11." The process is surprisingly time consuming.

The generated TV display is 16 rows shorter than the arcade display. I chose to chop eight rows from the top and

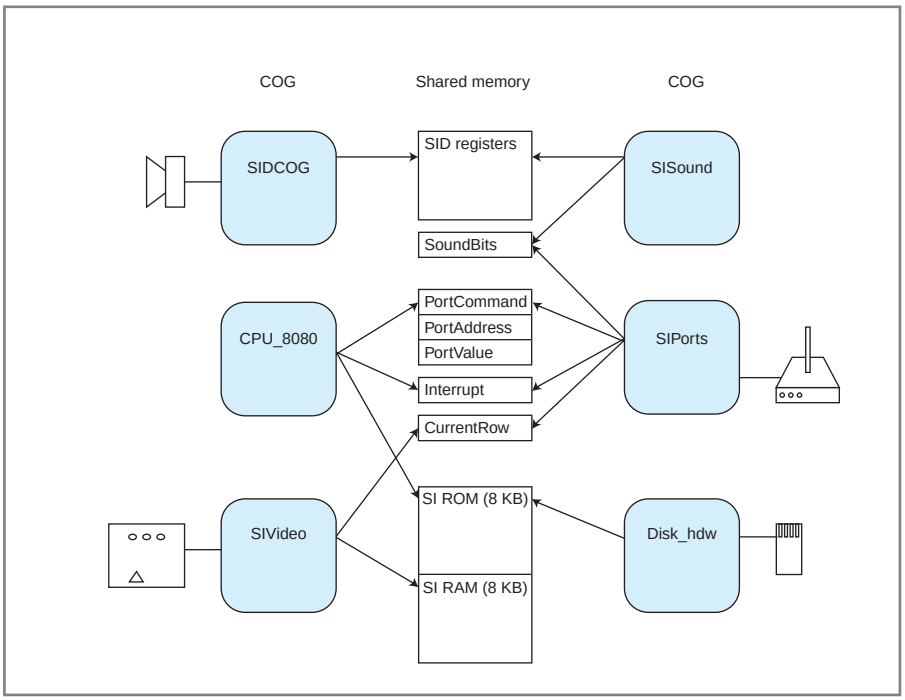


Figure 2—The COGs communicate with one another through parameter blocks in shared memory. The 16-KB RAM/ROM of the emulated system lives in the Propeller shared memory too.

eight rows from the bottom. The arcade display reads 32 × 224 bytes beginning at RAM address 0x2400. My modified display reads 32 × 208 bytes beginning eight rows in at 0x2500 (i.e., 0x2400 + 32 × 8).

The arcade monitor is rotated 90° counter-clockwise inside the cabinet. This provides more pixels up and down than left and right, and was common in the arcade classics (e.g., Pac-Man, Galaga, Frogger, etc.). The Propeller hardware, as well as my programming prowess, lacks the horsepower to rotate the raster buffer before display. I took the easy way out and just turned the TV on its side (see Photo 1)!

The video driver also writes the current row number to shared memory. Another COG reads this value and sends hardware interrupt requests to the CPU at the end and at the middle of every frame (see Figure 2). The game code updates objects at the bottom of the screen with the end-of-frame interrupt. Game objects at the top of the screen are updated at the middle-of-frame interrupt. This keeps an image from moving in the middle of being drawn, which would produce an unpleasant flicker.

PORTS

The arcade CPU reads the switches and DIP settings from input ports 0, 1, and 2. These are all binary inputs producing 0 when open and 1 when closed (pressed). Bit 2 of port 2 is connected to a tilt switch that triggers when the machine is rocked. In the early 1980s you would have found the Space Invaders cabinet right next to the pinball machines. So a "tilt" switch, like those found in pinball machines, would not have seemed that out of place. If you shake, slap, or otherwise abuse an SI cabinet, you will get a Tilt message and your game will end.

The port COG (SIPorts.spin) maps the buttons and switches on the joystick to the port bits as the SI code expects them. The Reset button on the stick is the coin bit for the game. Select and Start are player-2-start and player-1-start, respectively. Pressing both at the same time simulates a tilt. Port 6 in the arcade game is used to reset the WDT. The code simply ignores these writes. Ports 3 and 5 control individual sound effects. The port COG code







Listing 1—Instructions are decoded in three steps: Get, Operation, and Set. An example of each is shown here. The Get instruction combines the H and L registers into an address and reads the data from memory. The Operation instruction waits on the port COG to return the I/O value. The Set instruction uses one of the parameter fields from the table to identify an 8-bit register.

```
get_word_hl
        mov     address, h_reg          ' H is the upper register of pair
        shl     address,#8              ' Shift it into place
        or      address, l_reg          ' OR in the lower byte
        call    #read_memory_word       ' Read the word from 8080 ram
        jmp     #vect_2                 ' Done

ioIn    wrlong  data_8,io_port          ' Write the port address
        mov     data_8,#io_cmd_in       ' Write the ...
        wrlong  data_8,io_command       ' ... IN command
ioWI    rdlong  data_8,io_command wz    ' Wait for port handler ...
  if_nz jmp     #ioWI                   ' ... to respond
        rdlong  a_reg,io_data           ' Get the value
        jmp     #vect_3                 ' Done

setR8   movd    sr8a, paramRegD         ' Set the address of the destination register
        nop                             ' Required gap before access
sr8a    mov     0,data_8                ' Copy data_8 to target register
        jmp     #fetch                  ' Done
```

combines the bit values from these two ports and passes them on to the audio system.

Drawing a pixel image requires a lot of bit shifting. Imagine a single byte image with all 1s. This is a short, solid line eight pixels long. If the image is drawn at X=0, the 8 bits all fit in a single byte. That's the easy case. If the image is drawn at X=1, then the image must be shifted one bit to the right making a 16-bit value. The first 7 bits fit in the first byte of screen memory and the last bit shifts into the second screen byte.

The Space Invaders game has up to 60 moving objects on the screen at the same time (55 aliens, the player, and four shots). But the 8080 instruction set has no direct support for 16-bit shifting. To aid with the rendering, the hardware maps a 16-bit shift register to ports 2, 3, and 4.

You write image data into the shift register through port 4. When you write to port 4, the most-significant bit (MSb) of the 16-bit value becomes the least-significant bit (LSb) and the new value you write becomes the new MSb. You write the shift amount to port 2. The value can be 0 to 7. You then read the shifted data from port 3. If the shift value is 0, then you read the MSb of the 16-bit shift register. If the shift value is 1, then you read the seven lower bits of the MSb and the one upper bit of the LSb.

The port COG (SIPorts.spin) implements the shift register logic. Interestingly, only the four shot sprites in the game take advantage of the shift register. The aliens are always drawn on a byte boundary and the shift value is always 0.

Ports are like memory in that they have unique addresses and you read and write them. But, unlike memory, they are "triggered" by the read and write process. When you read from memory, you read the last value you wrote. But, when you read from a port, the hardware latches in the values of switches and such. The port COG waits for port requests from the CPU. The CPU writes the request to memory and then waits for the port COG to respond. The serial exchange takes longer than an access to shared RAM. But the command/response pattern is necessary to capture the access triggering of port hardware.

Listing 2—These are the first eight entries in the 8080 instruction decode table. Each 4-byte entry contains three jump vectors (for Get, Operation, and Set) plus two parameter fields. The complete table consumes 1 Kb of shared memory.

```
dispatch_table  '  29  26       18         9         0
'                  yyy_xxx_cccccccc_bbbbbbbbb_aaaaaaaaa
'
'     VECTOR-1 (a)           VECTOR-2 (b)      VECTOR-3 (c)     PARAM-V1 (x)   PARAM-V3 (y)   OP       ASM        FLAGS
long (nop_v1<<Dv1)        + (nop_v2<<Dv2)   + (nop_v3<<Dv3)                                   ' 00       NOP        ————
long (get_code_word<<Dv1) + (nop_v2<<Dv2)   + (setR16<<Dv3)  +                (dec_b<<Dr3)   ' 01w1wm   LD   BC,w  ————
long (getR8<<Dv1)         + (nop_v2<<Dv2)   + (setBR16<<Dv3) + (dec_a<<Dr1) + (dec_b<<Dr3)   ' 02       LD   (BC),A————
long (getR16<<Dv1)        + (opINC16<<Dv2)  + (setR16<<Dv3)  + (dec_b<<Dr1) + (dec_b<<Dr3)   ' 03       INC  BC    ————
long (getR8<<Dv1)         + (opINC<<Dv2)    + (setR8<<Dv3)   + (dec_b<<Dr1) + (dec_b<<Dr3)   ' 04       INC  B     sz-h-v0-
long (getR8<<Dv1)         + (opDEC<<Dv2)    + (setR8<<Dv3)   + (dec_b<<Dr1) + (dec_b<<Dr3)   ' 05       DEC  B     sz-u-v1-
long (get_code_byte<<Dv1) + (nop_v2<<Dv2)   + (setR8<<Dv3)   +                (dec_b<<Dr3)   ' 06bb     LD   B,b   ————
long (nop_v1<<Dv1)        + (opRLCA<<Dv2)   + (nop_v3<<Dv3)                                   ' 07       RLCA       —0—0x
```

SOUND

The sound effects of Space Invaders are as unique and alien as the space monster graphic on the side of the cabinet. Each of the 10 separate sounds is implemented in discrete analog circuitry with multiple op-amps and resistor/capacitor values you have probably never seen.

For this project, I chose to use a more generic sound system suitable for future growth. I have used the General Instrument (now Vishay Intertechnology) AY38910 programmable sound-generator chip in other projects. But in this project, I used the Propeller implementation of the MOS Technology (Commodore Semiconductor Group) 6581 sound interface device (SID). This chip, used in Commodore home computers, has a cult following on the web. I pulled in the SIDCog.spin file directly from the Parallax object exchange.

The SID chip produces three independent sound channels (tone or noise) with lots of envelope and waveform control. The SISound.spin is my attempt to reproduce the Space Invaders sound effects with the SID chip. The SISound driver watches the sound bits updated by the port COG. Channel 1 handles the fleet-stepping tones and the player explosion (one or the other). Channel 2 generates the sound of the players shot and the alien explosion. Channel 3 handles the mystery ship sound and the extended-play beeping.

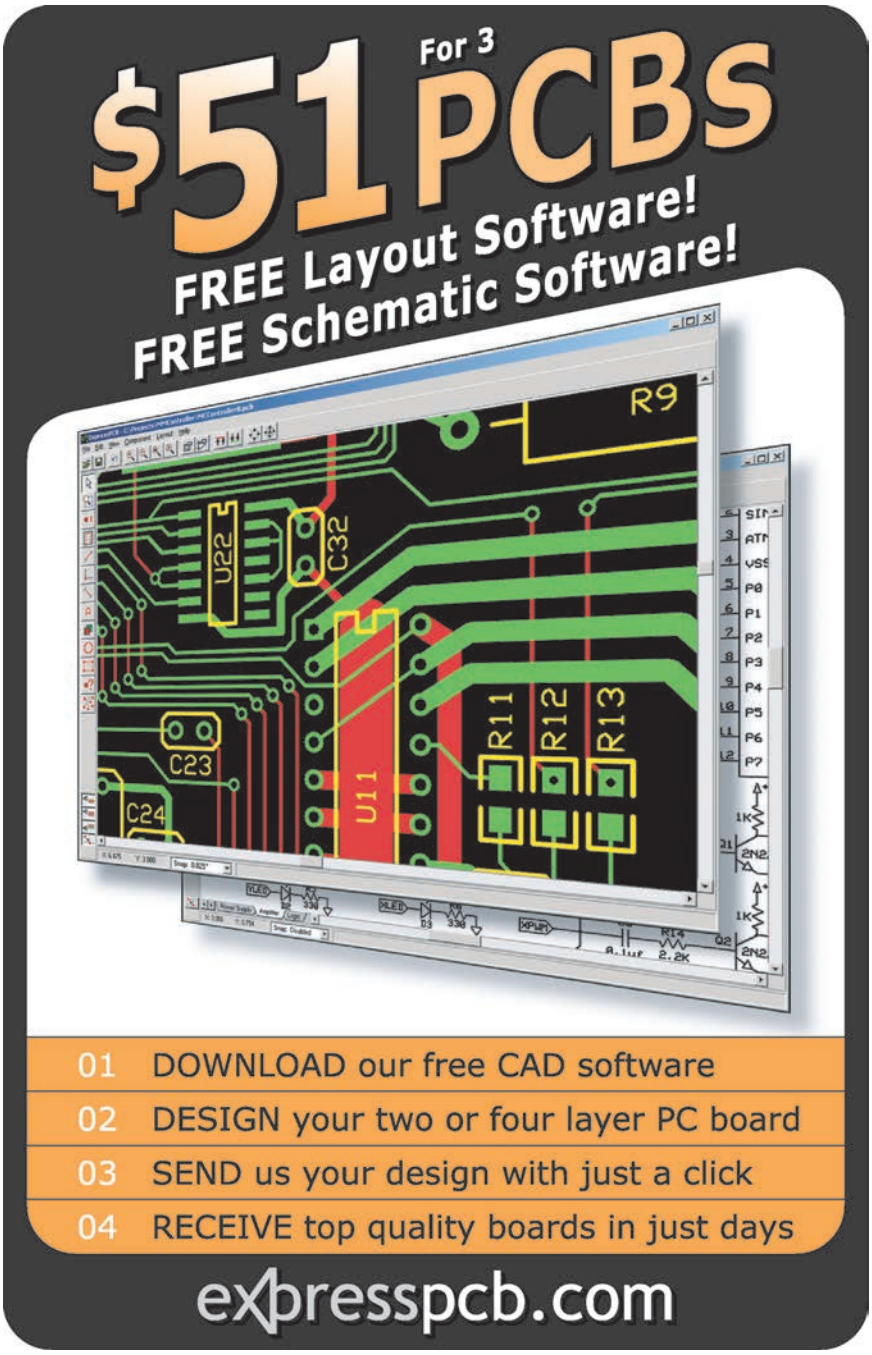
8080 CPU

The 8080 emulator I created for this project is based on Michael Rychlik's early Z80_emu_v0.0 work, which is available freely in the Parallax forums. The implementation (CPU_8080.spin) is in Propeller Assembly and barely fits in a single COG. The emulator uses a 1-KB table in shared memory to decode the 8080 instructions.

An 8080 instruction executes in three basic phases (see Listing 1). First, there is a Get phase where data is pulled into the execution path. The data could come from a direct or indirect memory read or it could come from a register. Then there is an Operation phase where the fetched data is incremented, added, subtracted, ANDed, and so forth. Finally, there is a Set phase where the resulting data is written to memory (direct or indirect) or a register.

In the 8080, there are relatively few of these individual functions. But they can be wired together in a large number of combinations. The idea in Michael Rychlik's code is to read a table with entries for each instruction. Each entry contains three vectors, one for each phase of the instruction (see Listing 2).

For instance, the emulator reads the opcode byte 0x86 from the program. This is the 8080 instruction "add the byte pointed to by HL to the A register." The Get vector points to the function in the emulator COG for "read the byte from memory pointed to by HL." The Operation vector points to the "add data to A and set flags" function.



HUMANDATA®
FPGA/CPLD Boards from JAPAN
SAVING TIME=COST with readily available FPGA boards
- Basic and simple features, single power supply operation
- Same board size and connector layout - ACM/XCM
- All stocked items are ready to be shipped immediately
- Over 100 varieties of FPGA/CPLD boards are available
- Free download technical documents before purchasing

ALTERA FPGA Board

CycloneIV GX F484 FPGA board
ACM-024 series
Cyclone IV GX | DDR2 | SIF40
EP4CGX50CF23C8N
EP4CGX75CF23C8N
EP4CGX110CF23C8N
EP4CGX150CF23C7N
Credit card size (86 x 54 mm)
RoHS compliant


Cyclone IV E F484 FPGA board
ACM-107 series
Cyclone IV E | MRAM
EP4CE55F23C8N
EP4CE75F23C8N
EP4CE115F23C8N
Compact size (43 x 54 mm)
RoHS compliant


XILINX FPGA Board

Virtex-5 LXT FFG665 FPGA board
XCM-017 series
Virtex-5 | SDRAM | RocketIO | SIF40
XC5VLX30T-1FFG665C
XC5VLX50T-1FFG665C
Credit card size (86 x 54 mm)
RoHS compliant


Spartan-6 FGG484 FPGA board
XCM-110/110Z series
Spartan-6 | MRAM | DDR2
XC6SLX45-2FGG484C
XC6SLX75-2FGG484C
XC6SLX100-2FGG484C
XC6SLX150-2FGG484C
Compact size (43 x 54 mm)
RoHS compliant


PLCC68 series

XP68-01 Spartan-6 PLCC68 FPGA Module

Spartan-6 | PLCC 68
XC6SLX16-2CSG225C
16Mbit Configuration Device
Two User LEDs
One User Switch(Slide)
RoHS compliant

AP68-03 Cyclone III PLCC68 FPGA Module

Cyclone III | PLCC 68
EP3C10U256C8N
4Mbit Configuration Device
Two User LEDs
One User Switch(Slide)
RoHS compliant

www.hdl.co.jp/CC/

HuMANDATA LTD.
E-mail: s2@hdl.co.jp
Fax: 81-72-620-2003

> "There are two circuit boards inside the stick. A small board at the front of the stick provides contacts for the three round buttons labeled Reset, Select, and Start. I left this board in place but cut traces and rerouted wires so that one side of each switch is grounded to the negative terminal of the 6-V battery pack. I left the power switch on the front of the joystick intact, but I rewired the tiny red power LED as a general-purpose output to be controlled by a Propeller I/O pin. Thus, the hacked mini board provides six wires to the main board: 6 V, ground, an LED output, and three switch inputs."

The Set vector points to the "store result in A" function.

The emulation code is a large list of primitive operations that are reused in many combinations. The "subtract" and "compare" instructions, for instance, use the same Get and Operation primitives. But the Set phase for the "compare" has a "no operation" for the Set vector. Only the "subtract" instruction keeps the result.

All addresses within COG memory are 9 bits (512 longs—2-KB in a COG). I made sure all Set routines were in the first half of the COG code making the leading bit of the Set vectors always 0 reducing that vector to 8 bits. Each 4-byte entry in the decode table holds the three vectors consuming 26 (i.e., 8 + 9 + 9) bits. That leaves 6 bits in each entry for additional parameter information used by the functions. Instead of having a separate function for Get of each 8080 register, I created a single Get function that uses the parameter value to specify which register. This moves information from code (in crowded COG memory) to the decode table in spacious shared memory.

The biggest challenge in implementing the 8080 emulator was debugging it. Here is a program written in Assembly that uses a decode table to execute another Assembly program written in 8080. The first time I turned on the joystick and fired up the emulation code, I got a few random dots on the screen. Now what?

I found a Space Invaders emulator written in Java on the web. The source code includes an 8080 emulator. I could run the Java emulator for several million instructions and stop it and dump the contents of all the instructions. Then I could run my emulator for the same number of instructions and compare the results.

The Propeller doesn't have a built-in debugging facility. I added a "print hex number" routine to the 8080 CPU code just for debugging. When debugging, the emulator's main loop runs a hardcoded number of instructions and then prints the contents of the registers to the raster buffer.

With a little binary-search trial and error we all learned as kids with the clock game on "The Price is Right," I could close in on a single point of failure. Fix the code and repeat. It was a painful exercise, but it was self rewarding. With each fix, more and more of the original game came to life on the TV screen.

The code does not emulate the timing of the 8080 instruction set. The emulator runs "wide open" running the 8080 code as fast as possible. Fortunately, the Space Invaders game is interrupt-driven synchronized to the video generator's interrupts and does not depend directly on the execution time. The original code spends a lot of cycles waiting on the next interrupt. The 8080 emulator is much faster than the original CPU. The game code wastes all the extra cycles in the wait loop before each interrupt.

ROOM TO GROW

There is a lot of exciting room to grow with this project! The Z80 processor is the big brother of the

8080. Many arcade games used the Z80. With just a little work, the CPU emulator could tackle many more games like Galaga that used three Z80 CPUs—three Propeller COGs. COGs are inexpensive at $1 a piece in sets of eight. The next revision of the joystick will have several Propeller chips.

Other CPUs are easier to emulate. The 6502 used in the Atari2600 and the Nintendo NES is much simpler than the 8080 used here. I believe the Atari2600 and NES hardware is within simulation reach of the Propeller. I envision several gaming console systems and their games packed onto a single SD card. The joystick definitely needs a graphical game selection menu.

In the days of the original Space Invaders, designers worked hard to squeeze precious cycles from their code. In our day and age, their efforts (hardware and code) can be emulated in an interpretive virtual machine implemented by a hobbyist with a microcontroller. My, how times have changed.

Chris Cantrell (tophercantrell@gmail.com) earned an MSEE from the University of Alabama. By day he writes Java and Flex for Emerson Network Power in Huntsville, Alabama. At night he builds Propeller-chip based projects and reverse engineers old arcade games on http://computerarcheology.com.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/260.

RESOURCES

ExpressPCB, www.expresspcb.com.

Parallax, Inc. Forums, "Thread: SIDcog – The Sound of the Commodore 64! (Now in the OBEX)," 2009, http://forums.parallax.com/showthread.php?118285-SIDcog-The-sound-of-the-Commodore-64-!.

———, "ZiCog: A Z80 Emulator for the Propeller," 2009, http://forums.parallaxinc.com/forums/pr.aspx?f=25&m=332138.

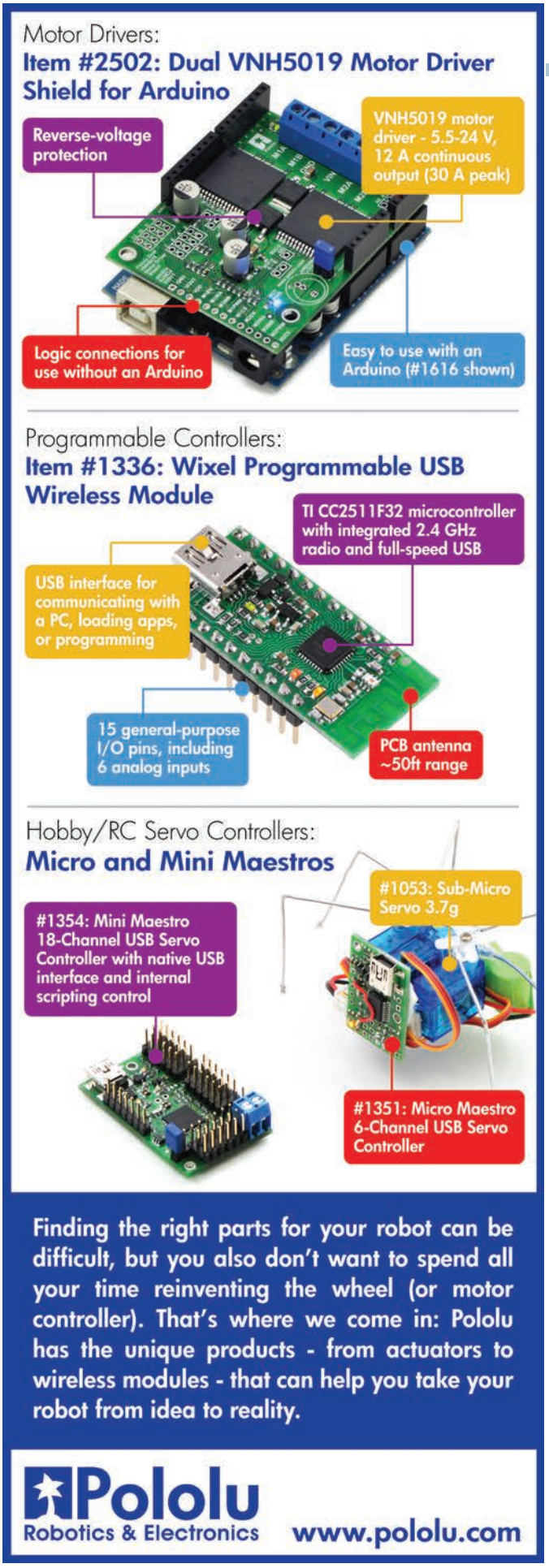
YouTube, TopherCantrell's Channel, "Propeller Space Invaders," www.youtube.com/watch?v=n8_lrEy30JE.

SOURCES

6581 Sound interface device
MOS Technology, Inc. (Commodore Semiconductor Group) | www.commodore.ca

Propeller chip
Parallax, Inc. | www.parallax.com

Breakout board for SD-MMC cards
SparkFun Electronics | www.sparkfun.com



by Charles Edmondson (USA)

The Rainbow Color Reader (Part 1)

A Talking Color Identifier for the Visually Impaired

The Rainbow Color Reader is a small, lightweight, inexpensive color identifier designed for the visually impaired. The design is built on a single processor that stores sound samples. It uses red, green and blue LEDs to identify colors and a speaker to provide sound. The first part of this article series details everything from the hardware to the speech software.

It can be dangerous to tell your spouse "Yes, that would be easy to do!" I had told my wife for years that I could build her a talking color identifier that wouldn't cost too much. When we formed our own company, with her as the boss, she told me to put my money where my mouth was and build her one. And, if it was that easy and that inexpensive, why don't we sell them too? So, I found myself taking a wild and wooly tour into the mysteries of color and embedded mixed-signal design.

The basic design seemed so simple: Shine a red light-emitting diode (LED) at the target, measure the reflection back, then do the same with green and blue LEDs. Analyze the measurements, determine what color the sample is, and speak that color. Like most "simple" designs, no part of this was going to be easy!

We started with a few ground rules. First, there would be only a single processor, with the sound samples stored on that chip. Next, there would be a single red, green, and blue (RGB) LED which would go through a clear window and return to an ambient light sensor phototransistor. A single button would start the process, with a small (but not too small) speaker to provide sound for a normal household environment. Finally, it should be small, lightweight, and most of all, inexpensive!



Photo 1—The completed Rainbow Color Reader (RCR), which is a compact, low-cost talking color identifier

This last requirement was because there were several talking color identifiers already on the market. They were all produced in Europe, they all cost in the multi-hundred-dollar range, and most required several steps to take a color reading. A retail target of less than $100 seemed easy.

In this article, I'll describe my Rainbow Color Reader (RCR) project. (The completed RCR is shown in Photo 1). And I'll cover everything from the hardware to the software, and calculating a hue, saturation, and luminance (HSL) value.

BASIC RCR DESIGN

The RCR is basically composed of three hardware blocks: First, there is the processor, with its programming, calibration, and power supply (see Figure 1). Second, there is the front end of the LEDs and phototransistor with buffer amplifiers and power supplies to take the actual measurements (see Figure 2). Third, there is the audio section with the speech filter, amplifier, speaker,

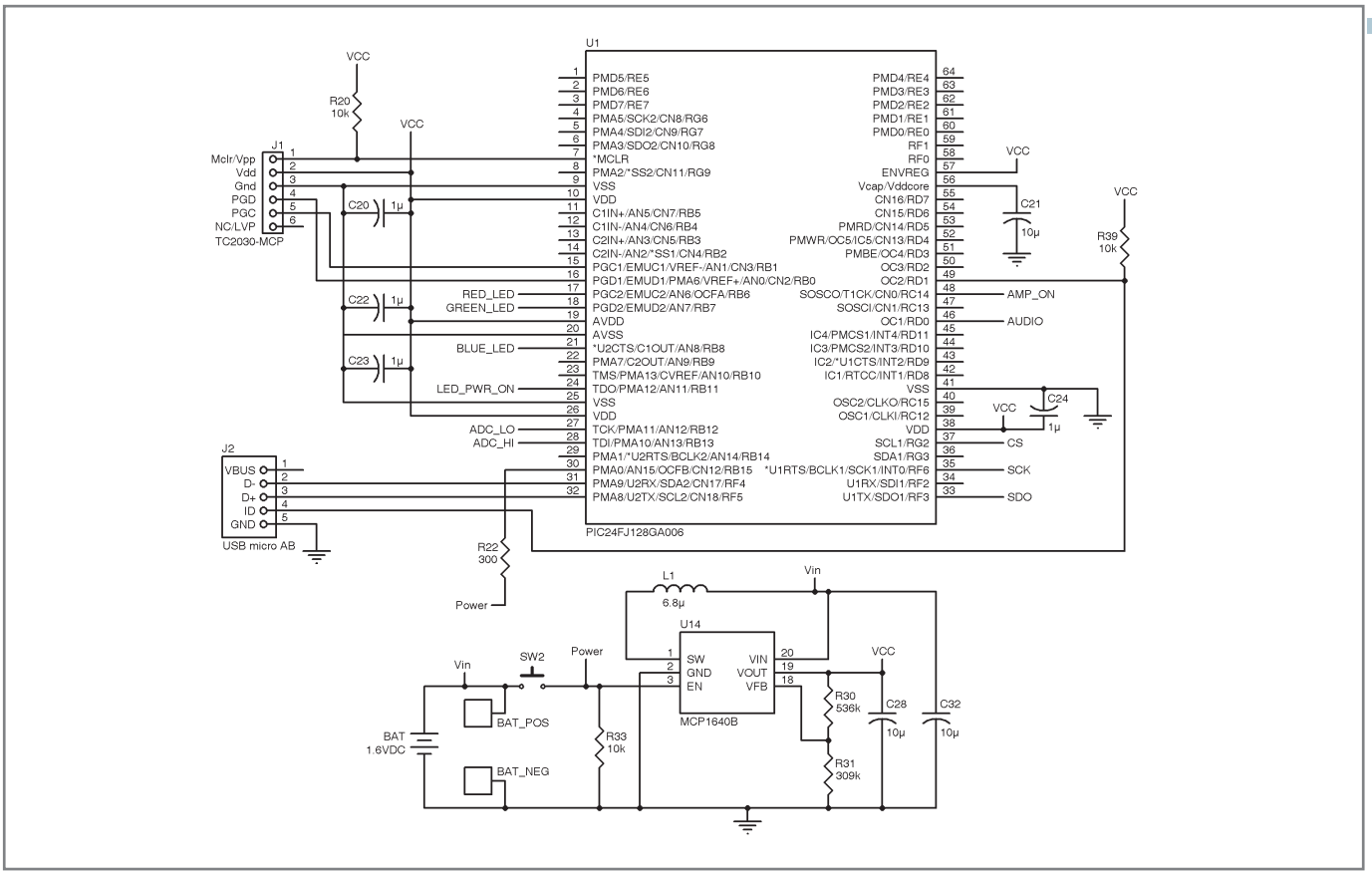


Figure 1—The main processor and power supply showing programming and calibration ports

and power supply (see Figure 3). Each of these sections and their associated software evolved greatly during the two years of development. I'll start with the easiest and probably most useful section for the general developer: the audio system.

SIMPLE SPEECH PLAYBACK SYSTEM

Whenever possible, I didn't want to reinvent the wheel. As a Microchip Design Partner, I knew there were a lot of resources on Microchip Technology's website. One of these was a complete hardware and software design for a talking thermometer for the Explorer 16 development board. Now, I didn't have an Explorer 16 board, but I did have several other development kits with Microchip's PIC24 16-bit processors on them, so I decided to adapt the software to run on my board. It will just be a few edits, right?

The hardware for the speech system is pretty straightforward. A pulse-width modulation (PWM) output from the PIC24 is filtered through a four-stage low-pass filter to remove the clock, and it goes to a class AB amplifier. My filter design was right out of the application note, but I used a Texas Instruments TPA301 audio amplifier for lower cost. It was the software that caused me the most problems.

The speech system starts with the recorded sound files, compressed with adaptive differential pulse-code modulation (ADPCM). These are stored on the chip using the MicrochiP File System (MPFS) and are called by the speech software which reads a file, decompresses the samples, and outputs the speech on a PWM channel on pin OC1. Over the first several months, I got way too familiar with each of these parts, as I alternately broke, and then fixed them.

First, let me explain the file system. For those who have worked with it, the MPFS is a part of the Microchip TCP/IP Ethernet stack. It enables you to create a FAT16 type of file system on your PIC processor. Since the chip on my development board only had 64 KB of flash, while the Explorer 16 came with 128 KB of flash, my first exercise was taking out part of the existing sound files so I could compile the software for my processor. I quickly learned there were two files that needed modification: mpfs.c and vdata.c. Mpfs.c contains the actual code for accessing the data, where there are two locations that list all the mpfs variables to be used. I needed to comment out the lines on the files I was editing. The actual data files were contained in vdata.c, where I had to comment out whole sections of hex data, as well as the file name listing at the bottom of the file. I would be doing a lot of editing of these sections of code!

After a few days of editing, I had a version of the code I could compile and test on my development board. I was using a PIC24 board I had received earlier for a *Circuit Cellar* design contest, and my first attempts to run the software were a disaster. All that came from my speaker was a loud squeal. After several days of debugging—and actually purchasing an Explorer 16 board—I found my

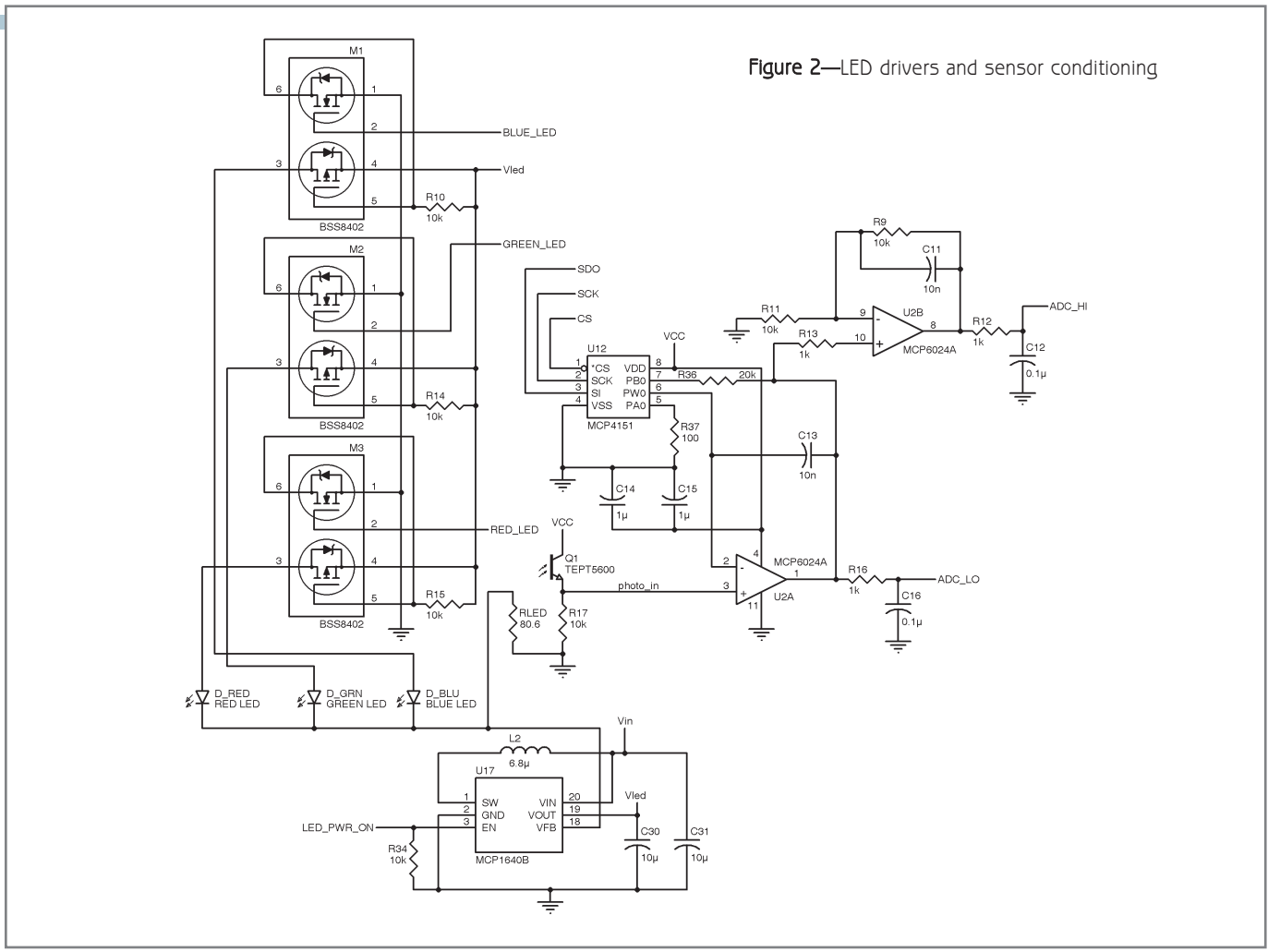


Figure 2—LED drivers and sensor conditioning

problem: I had the clocks on that 64-KB chip set wrong! A few changes to the configuration on my test program, and *voilà*! I had a talking system.

The next step was replacing all those prerecorded files with my new custom voice samples. I needed as many different colors as I could get, but they still needed to fit into a 64-KB chip with my software. After a few weeks of trial and error (mainly error!), I developed the following process.

To begin with, some audio software is needed to record a sample and save it in an unsigned 16-bit encoded file. I used Goldwave's digital audio editor for this step, as it had an evaluation version I could use to verify the process before putting up any money. You have to be careful, most audio software will only write a signed 16-bit file.

After recording and saving my samples, I needed to ADPCM encode them for the Microchip software to read. The example files I had downloaded included a utility called Winspeech.exe that performed this encoding. I saved the encoded files with names of A.RAW, B.RAW, C.RAW, and so forth in their own directory called VDATA2. Notice that these are in all caps—the adpcm.c file assumes that all file names are all caps. Once all the files have been encoded and saved in this directory, it is time to convert them to the MPFS.

This step is done using the MPFS.exe utility provided by Microchip. This utility takes the source directory and encodes it as a .C file with the various files translated to hex. I have to caution you: There are several versions of this software out there, with various differences in their input requirements and output formats. I used an earlier version. The output from this version required extensive hand editing of the resulting C code, as the headers for each file were incorrect, as was the directory structure at the end. It only took a few minutes of using search and replace to do the edits, but it created "interesting" problems when I forgot to do it, or missed a line or two. Once it was done, the code was compiled, and I could now easily and quickly access my custom speech files from my own software.

As development continued, and we added additional colors, eventually I ran out of room in the 64 KB of flash on my original chip. There was no other choice; I needed a little more room. Being a realist (as well as a Murphyologist!), I decided I needed twice as much flash, and went up to a Microchip PIC24FJ128GA006 16-bit microcontroller with 128-KB flash processor. The largest drawback was that now the package was a four-sided thin quad flat pack (TQFP) package, which was much more difficult to hand-solder. I also added voice samples for the numbers 0 through 9 which I used as feedback during calibration.

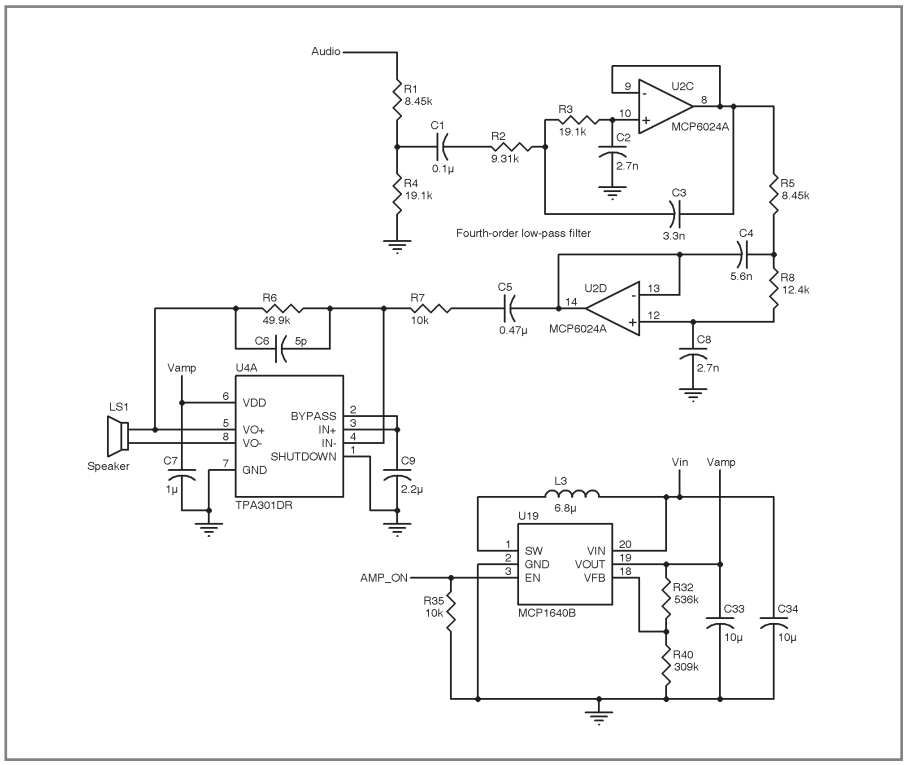


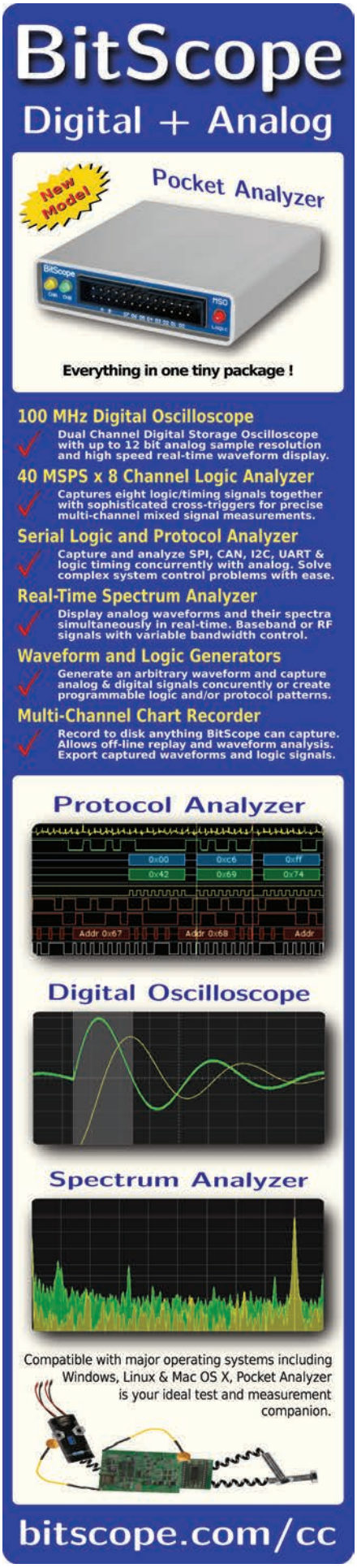
Figure 3—The audio filter, amplifier, and power supply

LEDS & PHOTOTRANSISTORS

When you are talking about a sensitive optoelectronic system like this, it pays to eliminate as many variables as possible. While I originally designed around an RGB LED bent 90° to aim out the front of the board, as I came to the final prototypes, I found that side looking surface-mount device (SMD) LEDs were much easier to work with and provided just as good results. A single 20-mA switching supply was switched between the three LEDs using dual-complementary MOSFETs, enabling the LEDs to be powered by a single AA battery. The LEDs were mounted at a 45° angle to the front window on the top side of the board beside a trapezoidal cutout.

The light from each LED shown out through the window and was reflected back to a Vishay Intertechnology TEPT4400 ambient light sensor mounted on the bottom of the board. Due to the angle, most of the reflection from the window was

> "The RCR is basically composed of three hardware blocks: First, there is the processor, with its programming, calibration, and power supply (see Figure 1). Second, there is the front end of the LEDs and phototransistor with buffer amplifiers and power supplies to take the actual measurements (see Figure 2). Third, there is the audio section with the speech filter, amplifier, speaker, and power supply (see Figure 3). Each of these sections and their associated software evolved greatly during the two years of development."



reflected to the side which reduced as much light as possible that wasn't coming from the target hitting the sensor. I also ended up painting the board's top, bottom, and edges with a black Sharpie marker to reduce transmitted light from reaching the sensor.

The electronics were kept as simple as possible (refer to Figure 2). A resistor changes the photocurrent from the sensor into a voltage, which is amplified by a single noninverting Microchip MCP6024 quad op-amp stage to our analog-to-digital converter (ADC) input. The gain of this amplifier is controlled via a Microchip MCP4151 SPI digital pot, which enables calibrated control. A low-pass filter on the ADC input helps to reduce any ripple in the LED output from the switching power supply. A second stage with a gain of two provides more accurate measurements of dark objects.

ENCLOSURES & POWER SUPPLIES

The final hardware step was deciding what type of enclosure to use and how to power the system. These requirements went hand in hand, as the type and size of the enclosure mandated what batteries I would be using to power everything. My first designs were all built around the Serpac Electronic Enclosures M-6 series. These had a nice battery compartment for use with either two AA or one 9-V battery. My very first prototypes just powered everything directly from two AAs, but this proved too unreliable a drive for green and blue LEDs. Light levels changed greatly as the batteries were depleted. I went through a series of design steps, with different power supplies for the LEDs and the processors, until I finally settled on using supplies built around a Microchip MCP1640 DC-DC converter. This chip enabled a small, simple switching power supply that provided sufficient power for a fairly low price and parts count. One advantage was that the step-up range was great enough that I could power these supplies from only a single AA battery! With this capability, I was able to change to the Serpac M-4 case with only a single AA battery compartment, which was a major reduction in size, weight, and cost. The only disadvantage was that each of these power supplies only had a limited power output. I have to have separate supplies for the LEDs, the processor/filter section, and the audio power amplifier.

This concludes the "easy" part of building the RCR. Next time, the fun part—color theory and how to actually tell one color from another in software!

Charles Edmondson (charles@edmondsonengineering.com) is the VP of Engineering at Edmondson Engineering, Inc. (EEI), where he specializes in embedded and mixed-signal design. Prior to starting EEI, he worked for 10 years at Cadence/Orcad/Microsim as a lead applications engineer for PSpice. He also did design and installation on the 91 Express Lanes and other toll roads, and A/V system design at University of California, Santa Barbara.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/260.

SOURCES

Goldwave digital audio recorder
Goldwave, Inc. | www.goldwave.com

Explorer 16 development board, PIC24 and PIC24FJ128GA006 microcontrollers, MCP6024 quad op-amp, MCP1640 DC-DC converter, and MCP4151 SPI Digital POT
Microchip Technology, Inc. | www.microchip.com

M-6 and M-4 Series of enclosures
Serpac Electronic Enclosures | www.serpac.com

TPA301 Audio amplifier
Texas Instruments, Inc. | www.ti.com

TEPT4400 Ambient light sensor
Vishay Intertechnology, Inc. | www.vishay.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*. For more need-to-know information about some of the topics covered in this article, the *Circuit Cellar* editorial staff recommends the following content:

—

Sound Synthesis Made Simple
A Multi-MIPS Music Box
by Peter McCollum
***Circuit Cellar* 228, 2009**

Peter entered the world of simple music synthesis and built a compact electronic sound-generating box. The design features DC-to-DC converter circuitry, an EEPROM containing sequences of notes, 18 LEDs, and a microcontroller and audio circuitry. Topics: Sound, DC, Converter, EEPROM, LED, Audio

—

Internet-Connected Sonic Anemometer
by Ingo Cyliax
***Circuit Cellar* 186, 2006**

Ingo's Internet-based sonic anemometer gauges wind velocities by measuring the speed of sound in air. This article details how the system works. Ingo has included all the information you need to get started on an anemometer of your own. Topics: Anemometer, Sonic, Internet, Lidar, Pitot, Ultrasonic, RCM3000, SX28/20

—

Go to *Circuit Cellar*'s webshop to find these articles and more: www.cc-webshop.com

ROUTE FASTER !



WITH PROTEUS PCB DESIGN

Our completely new manual router makes placing tracks quick and intuitive. During track placement the route will follow the mouse wherever possible and will intelligently move around obstacles while obeying the design rules.

All versions of Proteus also include an integrated world class shape based auto-router as standard.

PROTEUS DESIGN SUITE

- Hardware Accelerated Performance.
- Unique Thru-View™ Board Transparency.
- Over 35k Schematic & PCB library parts.
- Integrated Shape Based Auto-router.
- Flexible Design Rule Management.
- Polygonal and Split Power Plane Support.

Features:

- Board Autoplacement & Gateswap Optimiser.
- Direct CADCAM, ODB++, IDF & PDF Output.
- Integrated 3D Viewer with 3DS and DXF export.
- Mixed Mode SPICE Simulation Engine.
- Co-Simulation of PIC, AVR, 8051 and ARM7.
- Direct Technical Support at no additional cost.

Visit our website and use Promotional Code CLR2011JGB for an extra 10% Discount. Prices from just $249!

labcenter Electronics www.labcenter.com

Labcenter Electronics Ltd. 411 Queen St. Suite 201, Newmarket, Ontario, Canada
Toll Free 866.499.8184, www.labcenter. com or Email: info@labcenter-electronics.com

by Alexander Pozhitkov (USA)

The NakedCPU (Part 2)

Experimentation and Communication

The first part of this two-part series provided an overview of the NakedCPU platform, which is designed to provide full access to hardware and a CPU without any operating system restrictions. This article explores the BIOS power-on code, noise production, and reading and writing from the parallel port. Experiments with a LAN adapter and network packets are also covered.

Part 1 of this two-part series covered the hardware of a computer based on an Intel CPU. I detailed the NakedCPU, which is an operating system-free experimental platform. The platform consists of two computers connected via a serial port. One computer, the NakedCPU itself, is booted up with a tiny 32-bit protected-mode start-up code. After the start-up, this computer waits for a stream of executable and data bytes coming from the master computer, which is running Microsoft Windows.

Now it's time to consider actual experiments that open up the possibilities for further research! Using the NakedCPU Explorer I presented in Part 1, I will explore the BIOS power-on code, describe how to make the speaker produce some noise, and explain how to read and write the parallel port. In the end, I will detail how I experimented with the LAN adapter and received and dissected network packets.

Listing 1—This is the protocol created as a result of reading ICH documentation and conducting a few experiments.

```
>write              //Speaker enabled. NakedCPU begins producing a continuous sound.
Port address: 0x61
Value: 0x3
>write              //Change the default frequency by writing a timer configuration word. Sound stops,
Port address: 0x43  //because timer is expecting to receive a 2-B divisor. Write a new value 0x0110 in two
Value: 0xbe         //transactions, after which the NakedCPU will begin producing a high-pitch noise.
>write
Port address: 0x42
Value: 0x10
>write
Port address: 0x42
Value: 0x1

>write              //Write a larger divisor 0x2000 to lower the frequency. Writing the configuration
Port address: 0x42  //word is not necessary.
Value: 0x0
>write
Port address: 0x42
Value: 0x20
```

Listing 2—This protocol demonstrates writing to and reading from the parallel port.

```
>write                  //Parallel port turns into Extended Capabilities Port mode via modification of
                        //the Extended Control Register (0x77A). Line D2, goes HIGH and LED glows.
Port address: 0x77A
Value: 0x34
>write
Port address: 0x378
Value: 0x4

>write                  //Reading from LPT is also possible. By writing 0x2c into the Port Control
                        //register (0x37A), we make parallel port read. The LED slightly glows
                        //indicating that pull up resistors are enabled in the reading mode.
Port address: 0x77A
Value: 0x34
>write
Port address: 0x37A
Value: 0x2c
```

EXPERIMENTS: MAKING NOISE

Although it may sound trivial, making a PC speaker produce sound involves an understanding of timers and some low-level work. Ironically, there isn't a way to make a speaker beep using the Windows API on Vista or XP 64-bit version because Microsoft decided that the speaker hardware is obsolete.[1] Certainly, in the past, DOS programmers must have known how to do it, but now it seems to be forgotten. Reading input/output controller hub (ICH) documentation and conducting a few experiments resulted in the protocol for the NakedCPU Explorer shown in Listing 1.[2]

LIGHTING AN LED

The parallel port is becoming more and more obsolete. Nevertheless, it offers a possibility to read and send more than eight lines of data. Strangely, ICH documentation does not say anything about programming a parallel port. Browsing the Internet reveals that there is still some interest regarding the parallel port and programming information is available. Connect an LED to the D2 port line via a 470-Ω resistor and follow the protocol in Listing 2, which demonstrates writing to and reading from the parallel port.

THE "FIRST CRY"

Which instruction does the processor execute first after powering on? Intel documentation says that the processor reads its first instruction from the address 0xFFFFFFF0 (i.e., 16 bytes below 4 GB).[2] Attempting to examine this address with a debugger is fruitless. (I tested it, and it did not work.) In order to reach this high address, which is in the range of high BIOS, a small executable for the NakedCPU was prepared. The executable defined a segment of memory addressing high BIOS and sent the content of the 16 bytes below 4 GB back to the master computer. As expected, there was a short jump to approximately 30 KB below. The executable was modified to download the entire chunk of memory 30 KB below 4 GB up to the top. Curious inquirers are welcome to investigate the saved content and the code of the executable, which are available on *Circuit Cellar*'s FTP site. As a general impression, one can see many accesses to PCI bus and calls for CPUID instruction. It certainly makes sense. Various devices have to be set up and BIOS is attempting to determine which processor is being used.

NETWORK

Communication via the network is accomplished using the media access controller (MAC). Documentation is available from Intel. It is also helpful to read the first three chapters of the IEEE 802.3-2008 standard to get an idea of the low-level network lingo as well as the packet format sent over the wires.[3]

The NakedCPU Explorer enabled me to investigate the internals of the MAC and conduct some experiments. The MAC requires data structures in memory and configuration transactions via the I/O address space. The first step is to determine the I/O address of the MAC, which is called Base Address 2 (BAR2). The address is stored in the PCI configuration space at bus 0, device 25, function 0 (B0:D25:F0) register 0x18. By the way, there is confusion in the documentation referring to the same register. ICH calls this particular register MBARC, while

Listing 3—Instructions for conducting PCI transactions

```
>pci                                                    //PCI transaction consists of two steps: define the
                                                        //location and read the content.
Enter Bus Device Function 0xRegister: 0 25 0 0x18
>read32
Port address: 0xcfc
ecc1
```

Field	BitEnd	BitStart	Initial Value	Description	4100240
FD	0	0	1b	Full-Duplex controls the MAC duplex setting when explicitly set by software. Loaded from the NVM word 13h 0b = Half duplex. 1b = Full duplex.	0
Reserved	1	1	0b	Reserved, write as 0b for future compatibility.	0
Master Disable	2	2	0b	When set, the MAC blocks new master requests on the PCI device. If no master requests are pending by this function, the Master Enable Status bit is set.	0
Reserved	5	3	000b	Reserved, write as 0b for future compatibility.	000
Reserved	6	6	1b	Reserved.	1
Reserved	7	7	0b	Reserved, always set this bit to 0b.	0
SPEED	9	8	10b	Speed selection. These bits determine the speed configuration and are written by software after reading the PHY configuration through the MDIO interface. These signals are ignored when autospeed detection is enabled. 00b = 10 Mbps 01b = 100 Mb/s 10b = 1,000 Mbps 11b = not used	10
Reserved	10	10	0b	Reserved, write as 0b for future compatibility.	0

Table 1—A breakdown of the value 0x4100240 (top right) into bits is shown as a fragment of the worksheet. Bits appear after textual descriptions of the individual fields.

the MAC documentation calls it BAR2. Conduct PCI transactions with the NakedCPU Explorer (see Listing 3).[2, 4]

The number 0xECC1 means the I/O address is actually 0xECC0 with the 0th bit hardcoded to 1 to indicate that the address is indeed in the I/O space as opposed to being memory-mapped.[4] The latter indication is important because all configuration and communication with the MAC can also be done using memory-mapped registers, which is faster. However, for our experiments, it is sufficient to use the I/O space, because it is simpler and accomplishes the same results as memory-mapped operations.

In order to interact with the MAC, the inquirer writes an address of a register within the MAC into the BAR2 I/O address (0xECC0). After that, BAR2+0x4 (0xECC4) becomes a window to the value of that MAC register. It is important to mention that BAR2 and BAR2+0x4 accept only 32-bit double-word read/write operations. The MAC registers have plenty of bits to deal with and some bits are dependent on one another. It is difficult to understand the settings just by looking at the hexadecimal value of a register. An Excel worksheet, InterpretRegister.xls (also available on *Circuit Cellar*'s FTP site), features a macro that helps in this situation. Specifically, a table containing bit descriptions should be copied and pasted from the hardware documentation, and a hexadecimal value of a register will be converted into binary 1s and 0s in appropriate cells right next to the description text (see Table 1).

Let us examine the CTRL(0x0) and STATUS(0x8) registers. The address of each register is provided in parentheses. After power-on with the network cable unplugged, you will see what is shown in Listing 4.

After connecting the master computer with the NakedCPU via a network cable, the CTRL register stays the same as expected, while the STATUS register changes to 0x80080683. The new value means full duplex communication, established link, and 1-Gbps speed. The master computer running Windows XP reported the same communication parameters, which indicates that the NakedCPU network interface was able to negotiate with the master computer's interface on the hardware level.

Listing 4—The code shown after power-on with the network cable unplugged

```
>write32                    //This particular bit constellation
                            //determines among other things
                            //enabled automatic configuration for
                            //speed and full/half duplex.
Port address: 0xecc0
Value: 0
>read32
Port address: 0xecc4
100240

>write32                    //These bits tell that there is no
                            //link established but initialization
                            //is completed.
Port address: 0xecc0
Value: 0x8
>read32
Port address: 0xecc4
80080600
```

RECEIVING & INTERPRETING NETWORK PACKETS

In this section, I'll cover how to experiment with reading network packets originated from the master computer. I found that when the master computer detected the live NakedCPU via the network cable, Windows began generating DHCP requests. These requests are attempts to obtain an IP address and other high-level network settings because Windows assumes (erroneously) that the NakedCPU is a router or a network server. Although Windows is mistaken, this is perfectly fine for our experiments, because we can catch these packets and examine them.

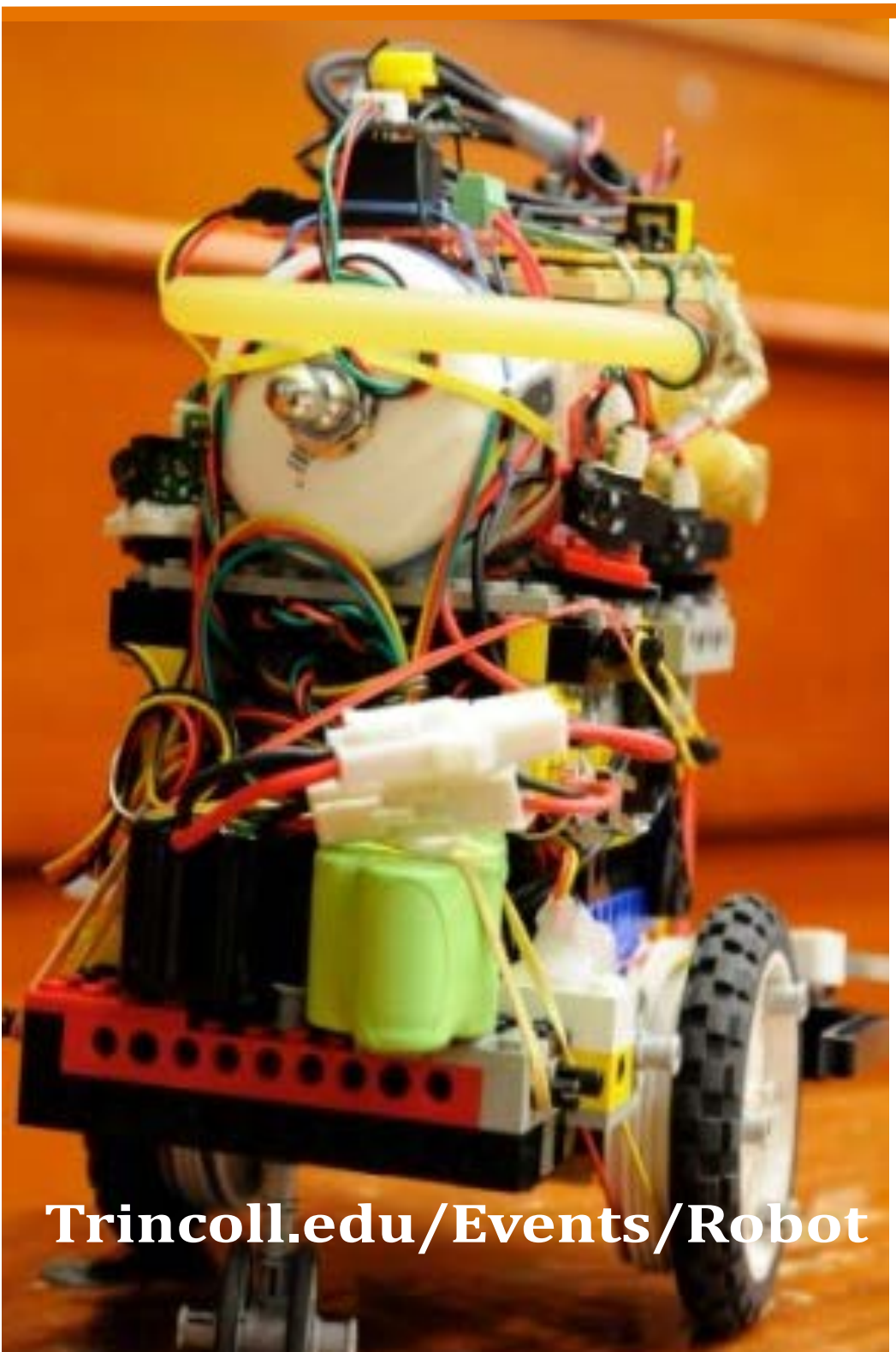
The MAC uses direct memory access to store the received data. We

Trinity College Robotics Competitions

Hartford, Connecticut, USA

ROBOWAITER

Promoting Assistive Robotics for Persons with Disabilities



Trincoll.edu/Events/Robot

Saturday March 31

8:30 a.m. - 7:00 p.m.
Robot Practice

10:00 a.m. - 11:30 a.m.
Robotics Workshops

12:30 p.m. - 1:45 p.m.
Robotics Keynote Speakers

2:00 p.m. - 4:00 p.m.
RoboWaiter Competition

Sunday April 1

11:00 a.m. - 4:00 p.m.
Versa Valves, Inc. presents
Firefighting Home Robot Contest

Proudly Sponsored by:

VERSA Valves

Shown left: 2011 Winner, *MonsterBot* from Massachussetts, USA

Listing 5—After the MAC finishes storing packets, it will update the descriptors to indicate received packet size, errors, and several other parameters such as these.

```
>memwrite                          //The command "memwrite" asks for the address and the number
                                   //of double words to be written. Note, writing below 0x100000
                                   //will cause general protection fault and reboot of the NakedCPU.
0xAddress above 0x100000: 0x100500
Num dwords: 8
0x101000
0
0
0
0x101200
0
0
0
```

have to create several descriptors to tell the MAC where to write the data. Thus, two memory ranges are required: one for the descriptors and the other for packets. Refer back to Table 3 in Part 1 of this series. You can see there is an area of memory above the address 0x100000 available for the inquirer's data. Bearing in mind that the NakedCPU Explorer uses a tiny bit of that memory to store incoming commands, we can safely use addresses above 0x100500. It is sufficient to create two descriptors for the initial experiments. A descriptor is a data structure of four double words (16 bytes). The first two double words are a 64-bit physical address of the location where the packet is to be stored. With the NakedCPU Explorer's ability to write into memory locations, I can create the descriptors at the address 0x100500, pointing to two 512-byte long buffers located at the addresses 0x101000 and 0x101200. A collection of descriptors is called a "queue." After the MAC finishes storing packets, it updates the descriptors to indicate the received packet size, errors, and several other parameters (see Listing 5).

The MAC has to know the location of the descriptors, the size of the receive buffers, and the type of packets to receive. This information has to be stored in several MAC registers: RDBAL0(0x2800) and RDBAH0(0x2804)—low and high portions, respectively, of a 64-bit physical address of the base of the queue; RDLEN0(0x2808)—length of the memory buffer allocated for the queue; RDH0(0x2810) and RDT0(0x2818)—head and tail pointers, respectively; RFCTL(0x5008)—receive filter control register; and RXCSUM(0x5000)—receive checksum control register. Before setting up these registers, bit 26 of the CTRL(0) register has to be set to 1, which causes the MAC to reset. After setting up all registers, data reception is initiated by writing into RCTL(0x100) to set up the "enable" bit, the size or the receive buffers reception mode, and the type of descriptors.

For the CTRL, RCTL, RFCTL, and RXCSUM registers, the InterpretRegister.xls worksheet shows values (and bit states) that will be used in our experiments. The meaning of value for the RDLEN is somewhat confusing. According to the documentation, the length of the queue buffer must be a multiple of 128, which means at least eight descriptors (128/16) must be in the queue. However, we have only two descriptors. I determined experimentally that it is not a problem to tell the MAC that the queue buffer is larger than it needs to be, as long as the RDT0 register is pointing to the end of the actual queue. Hence, for my particular experiment, I set RDLEN = 0x80, RDT0 = 0x2.

In order to set the MAC registers, I had to copy and paste the columns from Table 2 into the NakedCPU Explorer in the same order, column I through V. Note that at the end of the fourth step, the MAC is ready to enable reception. That step ends up in reading from the

I	II	III	IV	V
write32	write32	write32	write32	write32
0xecc0	0xecc0	0xecc0	0xecc0	0xecc0
0	0x2804	0x2810	0x5008	0x100
write32	write32	write32	write32	write32
0xecc4	0xecc4	0xecc4	0xecc4	0xecc4
0x4100240	0	0	0x8000	0x402800A
write32	write32	write32	write32	
0xecc0	0xecc0	0xecc0	0xecc0	
0x2800	0x2808	0x2818	0x8	
write32	write32	write32	read32	
0xecc4	0xecc4	0xecc4	0xecc4	
0x100500	0x80	0x2		

Table 2—Commands and values to be pasted into the NakedCPU Explorer

Descriptor 1	Descriptor 2
0	0
f8270fe8	f8270fe9
20073	20073
156	156

Figure 1—The descriptors updated by the media access controller (MAC)

status register, which should result in a value of 0x80683. This value is similar to the one previously described (0x80080683) with the difference that bit 31 is cleared, which indicates that the DMA clock cannot be lowered to one quarter of its value. The reason why the MAC changed its "mind" concerning the DMA clock is not clear, but this is not relevant for our experiments. I initiated the sending of DHCP packets by typing ipconfig/renew in the Master's command-line tool.

After reading from the network, the MAC updates the two descriptors (see Figure 1). With the `memread` command, observe the new values by reading eight double words beginning from the address 0x100500. You will see that the descriptor's address field is changed and two additional values appeared. An inset shows the new values my computer produced. Detailed information about the fields is provided in the MAC documentation.[4] Briefly, the lengths of the packets are 342 bytes (0x156). No errors occurred, the descriptors are indicated as "done," and the entire packet fits with the buffer (0x20073).

The MAC stored the two actual packets at the addresses 0x101000 and 0x101200, respectively. Their contents are present in the InterpretRegister.xls "Packets" worksheet. Figure 2 shows the beginning of the stored data array and the order of transmission. The first 6 bytes marked in blue are the destination (broadcast) address, which are followed by the source addresses and the 2-byte length/type field. The latter field is transmitted with most-significant byte first, which makes it 0x0800. If the value of the length/type field is less than or equal to 0x05DC, then it indicates the length of the packet, otherwise its type (Ethertype). Information found at the Institute of Electrical and Electronics Engineers

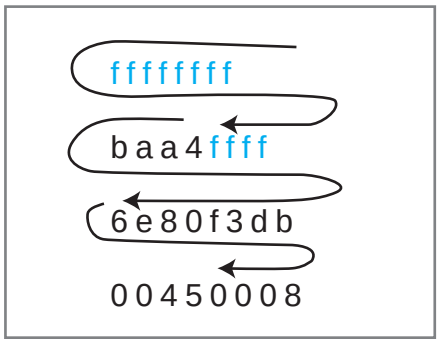
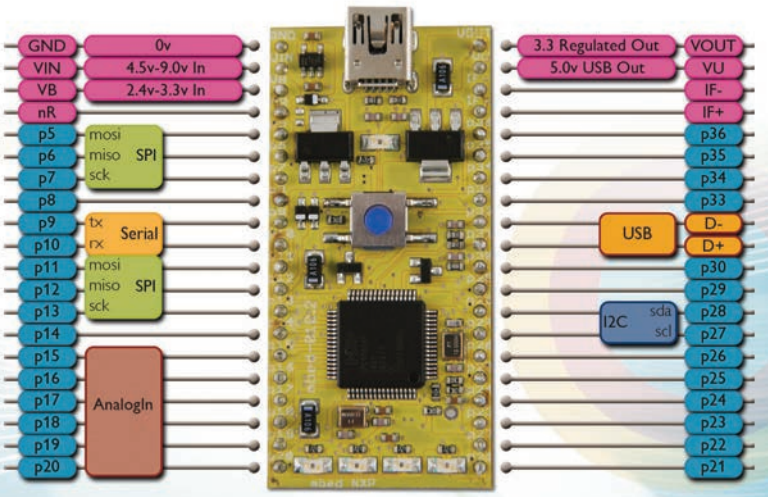


Figure 2—The beginning of the stored data array and the order of transmission

"The NakedCPU platform enables developers to create task-specific applications using only necessary components. For instance, for a large-scale database, there is no need to support a GUI, USB plug-and-play, audio cards, .NET, and many other things. An additional bonus is that the NakedCPU platform is immune to viruses. As in biology, where flexible viruses attack well-evolved organisms, computer viruses attack well-developed operating systems. With the NakedCPU, on the other hand, a particular task-specific solution can be very unique. Therefore, virus creators simply won't have enough information to explore potential security holes."

mbed

mbed NXP LPC11U24 Microcontroller



Rapid prototyping for USB Devices, Battery Powered designs and 32-bit ARM® Cortex™-M0 applications

http://mbed.org

Bit:	0 – 3	4 – 7	8 – 15	16 – 31
… 00450008 E80F4801 …	Version [0x4]	Internet header length (IHL) [0x5]	Type of service [00]	Total length [0x0148]

Table 3—The first 32-bit word of the IP header. The top line shows the bit numbers. The left column is a fragment of the received packet. The underlined values of the packet are broken down into the fields.

(IEEE) website in theory provides specific Ethertype values; however, it is virtually impossible to find the actual list of values. Luckily, Wikipedia points to an exact URL in the innards of the IEEE site. According to the standard, Internet protocol (IP) is designated with the Ethertype 0x0800. Apparently, the next step is to investigate the format of the IP, described in RFC894, which points to RFC791.[5]

The fields of the IP header are transmitted in a similar way as the previously described length/type field (i.e., the most-significant bit and the most-significant byte first). In contrast to the intuitive notation, where bit 0 is the least-significant bit, RFC791 provides the opposite, with bit 0 as the most-significant bit. Table 3 shows the continuation of the received packet and the assignment of its specific values to the fields of the first 32 bits of the IP header. The important fields are IHL and Total Length. IHL indicates the number of 32-bit words in the header. In my case, it is five, which means according to RFC791, the Options and Padding fields are omitted. It is interesting to note that the MAC reported receiving 342 bytes and the IP header indicated 328 (0x0148) bytes. The difference of 14 bytes makes perfect sense. They make up to 2 bytes of the length/type field plus 2 × 6 bytes of destination and source hardware addresses.

All other fields of the IP header are easy to map following this example. Specifically, the field values are: Identification—0x0fe8 (packet 1), 0x0fe9 (packet 2); flags and fragment offset—0; time to live—0x80; protocol—0x11; header checksum—0x29be (packet 1), 0x29bd (packet 2); source IP address—0.0.0.0; and destination IP address—255.255.255.255. It is understandable that the master PC is asking for an IP address while doing the dynamic host configuration process. Therefore, the source IP address is all zeroes. For the destination, the address is all 255.255.255.255, which is a broadcast address, similar to the hardware broadcast address (6 bytes, all 255).

The value for the protocol field is 0x11 (17). According to RFC790, it refers to the user datagram protocol (UDP)—the next level of data encapsulation—which is described in RFC768. According to that document, a UDP header contains four 16-bit words. In the order of transmission, these words are source port, destination port, length, and checksum. The values received by the NakedCPU in my experiment were: source port—0x0044, destination port—0x0043, length—0x0134, and checksum—0x5c1a (packet 1) and 0x581a (packet 2). The length of 308 (0x134) bytes makes sense, since the IP header reported the total datagram length of 328 bytes minus 20 bytes occupied by the IP header. The source and destination port numbers should have been explained in RFC790; however, it turned out that a long chain of other documents makes this one obsolete. At the end of the chain, it is suggested to look at the website of the Internet Assigned Numbers Authority (IANA), where, unfortunately, the information is not well organized. It was possible, however, to find among the obsolete RFCs that port numbers 67 (0x43) and 68 (0x44) correspond to the Bootstrap protocol, server and client ports, respectively. The Bootstrap protocol described in RFC1542 points to the DHCP protocol (see RFC2131). Figure 3 depicts all the fields covered in this section.

FUTURE OUTLOOK

It is possible to directly experiment with the Intel CPU and other PC hardware without any layers of unknown intermediate code intended to make our lives "easier." As of yet, the most comprehensive documentation exists for the processor itself.[6]

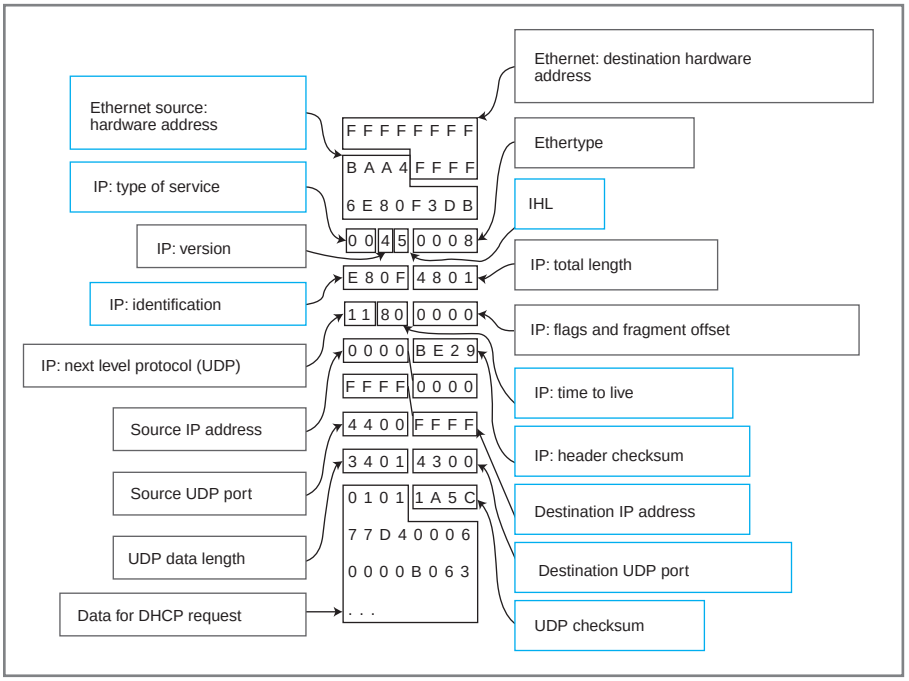


Figure 3—This is a summary of the data fields for Ethernet, IP, and UDP protocols in the received packets.

The other hardware is not well documented, which is why it takes a significant amount of effort to gather pieces of information from the Internet and conduct experiments. The old books that dealt with hardware are DOS-oriented and seriously outdated. The new hardware is hidden behind layers of unknown code.

In theory, the NakedCPU platform enables developers to create task-specific applications using only necessary components. For instance, for a large-scale database, there is no need to support a GUI, USB plug-and-play, audio cards, .NET, and many other things. An additional bonus is that the NakedCPU platform is immune to viruses. As in biology, where flexible viruses attack well-evolved organisms, computer viruses attack well-developed operating systems. With the NakedCPU, on the other hand, a particular task-specific solution can be very unique. Therefore, virus creators simply won't have enough information to explore potential security holes.

Dr. Alexander Pozhitkov (pozhit@uw.edu) has an MS in Chemistry and a PhD in Genetics from Albertus Magnus University in Cologne, Germany. For 12 years he has been involved with interdisciplinary research relating to molecular biology, physical chemistry, software, and electrical engineering. Currently, Dr. Pozhitkov is a researcher at the University of Washington, Seattle. His technical interests include hardware programming, vacuum tubes, and high-voltage electronics.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/260.

REFERENCES

[1] Microsoft Corp., MSDN, Beep function. http://msdn.microsoft.com.

[2] Intel Corp., "I/O Controller Hub 10 (ICH10) Family Datasheet," 2008, www.intel.com/content/dam/doc/datasheet/io-controller-hub-10-family-datasheet.pdf.

[3] Institute of Electrical and Electronics Engineers (IEEE) Standards Association, "IEEE Get Program," http://standards.ieee.org/about/get/.

[4] Intel Corp., "I/O Controller Hub 8/9/10 and 82566/82567/82562V Software Developer's Manual," 2009, www.intel.com/content/dam/doc/manual/i-o-controller-hub-8-9-10-82566-82567-82562v-software-dev-manual.pdf.

[5] The Internet Engineering Task Force (IETF), "Index of /rfc," www.ietf.org/rfc.

[6] Intel Corp., "Intel Core2 Duo Processor E8000 and E7000 Series Specification Update," 2010, http://download.intel.com/design/processor/specupdt/318733.pdf.

RESOURCE

Internet Assigned Numbers Authority (IANA), www. iana.org.

SOURCE

Windows XP
Microsoft Corp. | www.microsoft.com



by George Novacek (Canada)



Microelectronic Component Obsolescence

How to Prepare for an Inevitable Scarcity of Parts

Component obsolescence is a possible problem for which every microelectronics engineer must plan. Two of the most effective forms of preparedness are designing circuits with high tolerance margins and maintaining a preferred parts list.

Component obsolescence—two words dreaded by electronics manufacturers. In the "good old days," active electronic components, vacuum tubes, semiconductors, and integrated circuits, once developed, had been manufactured for years. Manufacturers of industrial, military, and even consumer goods were able to minimize production changes and plan purchasing and inventory for a long time ahead.

Single-source components have always been a threat to long product lifecycle. To widen the supply base and to alleviate potential problems arising from different manufacturers making similar, but not identical components, the Joint Electron Device Engineering Council (JEDEC) was established in the late 1950s. JEDEC-certified components had the same specification, no matter who their manufacturer was. Typically, part numbers for diodes started with 1N, such as 1N914 or 1N4003. Transistor part numbers started with 2N, such as 2N2222, and so on. The military established its own preferred component part numbers. These, often JEDEC, received additional environmental screening to achieve guaranteed reliability.

COMPONENT RELIABILITY & LIFECYCLE

Today, MIL-designated components are rare. Present day manufacturing processes are far better and more consistent than to require additional screening, which increased the component's price often by a factor of 10. Unfortunately, present-day industrial/automotive-grade components are generally rated for –40°C to 85°C (–40°F to 185°F) operation. For applications such as military and aerospace, where a wider temperature range of –55°C to 125°C (–67°F to 257°F) or greater is called for, one solution is custom screening and uprating. Screening has always been an expensive process, and some reports suggest that most of the failures during screening have, in fact, been caused not by components' weakness, but by their handling and accelerated overstressing. I consider component screening tantamount to driving a new car for 100,000 miles before selling it as "reliable." There are often ways to work around such specification constraints, which I prefer over any screening and uprating.

Parts made today, including those that are JEDEC- and MIL-certified, are not always truly identical to the original ones, and their substitutions don't always work. I had the "pleasure" of maintaining a design where the ubiquitous LM124 could be replaced by a new LM124 from one manufacturer only. New components, for instance, have greater bandwidth and may require a circuit modification to prevent instability. Such modifications, albeit small, may require a regulatory approval. Sometimes the design can be "grandfathered," but the recertification by rationale is not always simple. Two engineers in one room will have five incompatible opinions on a single issue, so it's often more expeditious to retest.

esc

a designwest summit

Learn today, design tomorrow.

With cutting edge product demonstrations, visionary speeches and hundreds of essential technical training and accreditation opportunities, ESC is the ideal venue for designers to learn, collaborate and celebrate excellence.

Register for ESC today, and get ready to design tomorrow's technology.

- Over 300 classes
- Over 100 world-class speakers
- Industry experts, live and on demand
- Over 300 exhibiting companies
- Over 1000 products and demos
- And more learning opportunities

-> March 26-29, 2012 <-
McEnery Convention Center
San Jose, CA
ubmdesign.com

Register now



Upgrade to a Design West All Access pass to get even more speakers, sessions, hands-on experiences – plus a FREE Tablet.



It would be a mistake to think that obsolescence only affects manufacturers of regulated products. Even though most consumer products today are throwaway and many circuit boards are replaced rather than repaired, anyone whose product requires support for some time to come must be concerned with component obsolescence. In the early 1990s, commercial-off-the-shelf (COTS) components were permitted (by Perry's Initiative) for military and other certified product manufacturing. This reduced the initial cost substantially (remember the $80 hammer?) but also opened the floodgates to the entire unverified components spectrum.

Today, there are in existence an estimated 500,000 plus microelectronic parts, each with several different packages and temperature ratings. Of those, some 50,000 are discontinued every year—137 components bite the dust daily, while at least as many new ones are added. About 200,000 microelectronic parts have their specs modified every year, and the average component lifecycle is three years. Neither JEDEC, nor any other system, can keep up with the volume.

This situation isn't as disconcerting in the huge consumer market where new models are often introduced annually. A three-year lifecycle is probably of little concern to the cellphone manufacturer. This situation is exaggerated by the volume of short lifecycle parts being such that component manufacturers don't much care about proven, reliable, long-life items. Those are, however, of significant concern to manufacturers of commercial, industrial, or military/aerospace —certifiable or not—electronics. Products that are to be manufactured for years without change and be supported through a multiyear warranty or contracts. Typically, an aircraft model is manufactured for some 20 years, and the contract requires the manufacturer of its electronic subsystem to support it for another 20 years after the last plane was built. If a direct substitute is not available or, as I mentioned earlier, the same but "improved" component from the same manufacturer with the same part number behaves differently, the product needs to be modified. And that can be very expensive.

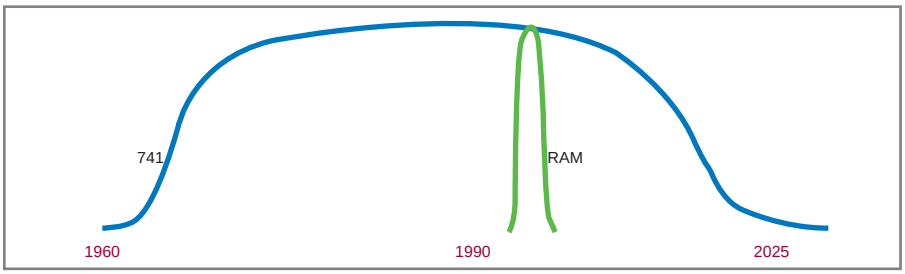


Figure 1—The analog 741 op-amp is expected to be around for another 10-plus years, while some commercial RAM disappeared after a mere two years.

UNPREDICTABLE OBSOLESCENCE

Figure 1 illustrates how unpredictable obsolescence of microelectronic parts is. While the life expectancy of the omnipresent 741 op-amp is more than 60 years, some mostly digital components are merely a "flash in the pan."

So the question is: How can product design engineers avoid the roadblocks caused by component obsolescence?

First, we must design circuits with as much tolerance margin as possible; avoid, if possible, single-source parts and never rely on characteristics that may change. Once I saw an "elegant" circuit where the input resistance of 22V10 programmable array logics (PALs) was used as a part of a timing circuit. The elegance disappeared just a few months into production when a new batch of the PALs arrived.

Second, companies need to maintain a preferred parts list (PPL) and require solid justification from the engineer who wants to use a component not on the PPL. I remember some nifty peripheral chips combining I/O ports with memory. They worked well and saved a lot of board real estate, so we incorporated them into the design. Unfortunately, they were discontinued shortly after our product completed qualification testing.

For help, component engineers and buyers can go to companies specializing in electronics market analyses, such as the Component Obsolescence Group or Information Handling Services (IHS). These companies rate parts in terms of their perceived lifecycle. You should stay away from parts too young, with no pedigree, or with too little track record to predict their performance and lives. And you definitely don't want to use parts with predicted lifecycles that are already on the decline. Obsolescence-monitoring companies make amazingly accurate predictions. The only drawback is that you must subscribe to their service. And it is not inexpensive.

When the inevitable happens and a part is on the chopping block, you have several options. The preferred one is to find a substitute, perhaps by a different manufacturer or in different packaging, which may require a small printed circuit board (PCB) modification. Some memories, for example, could have been replaced with larger ones with the same pinout. And there are brokers searching the industry for unused inventory. But this is risky; there is rarely traceability, you don't know what you're buying, and there are little guarantees.

Usually, when discontinuing a product, manufacturers use the term, "last buy." You can purchase a sufficient quantity to last you until the end of your product's projected lifecycle, plus warranty. Such a move, of course, has a negative impact on the inventory and the cash flow. And you need proper storage cabinets filled with nitrogen.

There are companies purchasing dies of discontinued components and then manufacturing and selling those components for a much higher price than the originals. A microcontroller I used to purchase for less than $10 was, after being discontinued by its original manufacturer, offered for more than $500! Replacing a discontinued microcontroller with a field-programmable gate array (FPGA) can cost even more due to its recertification.

Those options are generally practical for manufacturers of products that

must be certified only. The additional cost of the inventory or a custom-manufactured substitute may pale when compared with the cost of recertification, which can run in hundreds of thousands—and possibly millions—and usually requires compliance with the current, tougher standards. Even a simple change from through-hole to surface-mount technology (SMT) is tough to justify by rationale, as it may introduce a change of the center of gravity, vibration resonance, introduce new electromagnetic interference (EMI) issues, and so forth. In my experience, converting to SMT has always improved the design, but that's not an acceptable proof. Only a costly test program can silence speculations.

BE PREPARED

Although I've focused on microelectronic parts, passive components are not much better off. Quite often, their problems are caused by their manufacturers' rarely reported cost cutting and the subsequent degradation of performance. Component obsolescence is here, the lifecycles will continue to shrink, and the only defense we have is to be prepared for it and always think ahead.

George Novacek (gnovacek@nexicom.net) is a professional engineer with a degree in Cybernetics and Closed-Loop Control. Now retired, he was most recently president of a multinational manufacturer for embedded control systems for aerospace applications. George wrote 26 feature articles for Circuit Cellar *between 1999 and 2004.*

RESOURCES

Component Obsolescence Group International, Ltd., www.cog.org.uk.

Information Handling Services (IHS), Inc., www.ihs.com.

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*. For more need-to-know information about some of the topics covered in this article, the *Circuit Cellar* editorial staff recommends the following content:

—

EMI and Data Integrity
by George Novacek
***Circuit Cellar* 257, 2011**

Electromagnetic interference (EMI) is the most common cause of electronic data corruption. Shielding can protect against EMI and minimize the possibility of data corruption. Topics: Electromagnetic, Interference, EMI, Data Corruption

—

Go to *Circuit Cellar*'s webshop to find this article and more: www.cc-webshop.com



by Richard Wotiz (USA)



Infrared Thermal Detectors

Thermal sensing technology has come a long way since the first radiation thermometer was developed in the early 19th century. Review this survey of thermal sensing technologies—from thermopile sensors to pyroelectric detectors—before starting your next temperature-sensing design.

I remember being fascinated by infrared (IR) noncontact thermometers years ago when they first became popular in the consumer marketplace. The concept of measuring temperature at a distance almost seemed like magic at the time. I wanted to try one out for myself, but they were too expensive back then. Now that advances in sensor technology have brought costs down considerably, I was able to pick up a few different models to study (see Photo 1). This month I'll discuss various types of sensors used for measuring thermal radiation. I'll also describe how some products use them.

Any object with a temperature above absolute zero will radiate energy. The hotter an object, the more energy it emits and the shorter the wavelength of the energy. The total radiant energy is proportional to the fourth power of temperature. A radiometer is any device that measures this energy. If it's calibrated with a temperature scale, it's known as a radiation thermometer. If it's specifically designed to detect IR energy, it's commonly called an IR thermometer. The energy detected from an object also depends on its emissivity. That indicates how much of the energy is radiated from the object itself, rather than reflected off its surface from nearby sources. Many inexpensive IR thermometers assume a fixed emissivity of 0.95, which is reasonably accurate for most nonreflective materials.

THERMAL RADIATION

The first radiation thermometer was developed in 1828. It used the decidedly low-tech method of focusing thermal radiation onto the bulb of a thermometer using a concave mirror. In 1892, Henry Louis Le Chatelier created the first comparison-type optical pyrometer, which

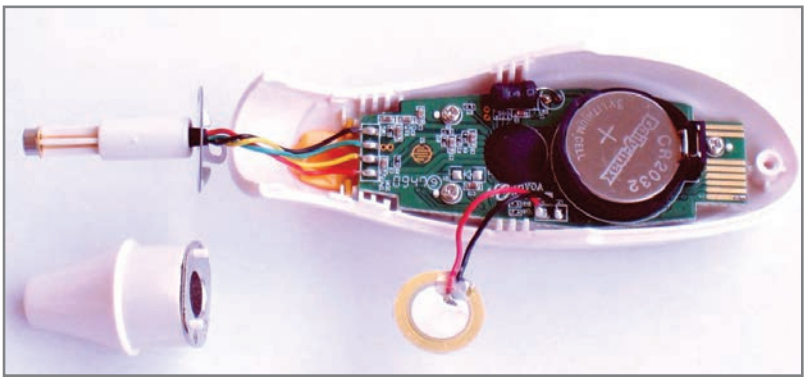


Photo 1—This is an inside view of an ear thermometer. The thermopile sensor fits snugly inside the cone-shaped tip, which you place in your ear to take your temperature. The dark blob next to the battery is the microcontroller. It's mounted chip-on-board, which is less expensive in high production volumes than using a packaged part. There's also a serial EEPROM on the back of the board to store calibration data and recent measurements.

Type	Sensor	Output
Thermal detector	Thermopile	Voltage
	Bolometer	Resistance
	Pyroelectric detector	Charge displacement
Quantum (photon) detector	Photovoltaic detector	Voltage
	Photoelectromagnetic detector	Voltage
	Photoconductive detector	Resistance

Table 1—Noncontact radiant energy sensors

was useful for measuring objects that were hot enough to glow. The user would adjust an iris diaphragm until the brightness of the object, seen through a viewing telescope, matched the flame of an oil lamp. The iris setting would then indicate the temperature. This technique was made more practical when an electric bulb replaced the oil lamp in the early 1900s. This instrument, known as a disappearing-filament optical pyrometer, still provides highly accurate temperature measurements for the steel and petrochemical industries.[1] Both of these devices required manual operation, a limitation that Charles Féry overcame in 1901 with the first radiation thermometer using a lens to focus thermal energy onto a thermocouple. The temperature could then be read directly from a galvanometer driven by the thermocouple.[2] By the 1950s this led to a more sensitive instrument using a photomultiplier tube as a detector, which evolved to the noncontact IR thermometers we have today.

Table 1 shows various commonly available sensors. Thermal detectors function by heating up as a result of absorbing thermal energy, which can take from milliseconds to seconds, depending on detector size. Quantum detectors measure incoming photons directly. This gives them greater speed, typically in the nanosecond-to-millisecond range. They are also more sensitive than thermal detectors, but operate over a narrower range of wavelengths that's dependent on the detector material. They often need external cooling to reduce noise when measuring temperatures at or below room temperature. We'll take a closer look at thermopile and pyroelectric detectors, since these are the most common noncontact IR sensors found in low-cost consumer-grade products.

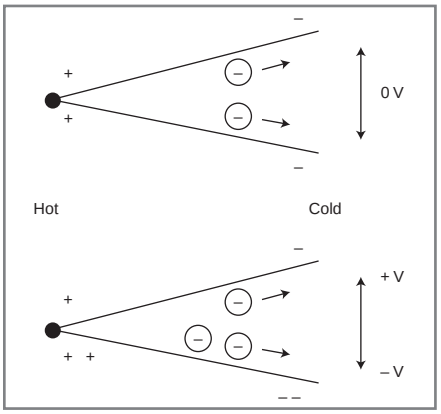


Figure 1—The thermoelectric effect in action. Heating two identical wires joined at one end has no net effect (top), but heating two dissimilar wires results in a measurable voltage (bottom).

THERMOPILE SENSORS

A thermopile consists of a number of series-connected thermocouples. To understand the operation of a thermocouple, we need to go back to 1821, when T. J. Seebeck discovered that a current flowed through two wires of dissimilar metals that were joined at both ends if one junction was at a different temperature than the other. This is known as the thermoelectric, or Seebeck, effect, and results from electrons diffusing from a hotter to a cooler section of wire. If you heat up part of a single wire, electrons will diffuse equally in both directions along the wire, causing a voltage gradient within the wire but no net voltage difference at its ends. But the diffusion rate varies for different materials. So, if you heat the junction of two dissimilar wires, the gradients will be different for each wire, and you'll see a voltage developed across the opposite ends of the wires. Figure 1 shows this phenomenon in action. The voltage is typically in the microvolt range for small temperature differences (below 100°C), so multiple thermocouples can be assembled into a thermopile to get a more manageable output signal. You can learn more about thermopile sensors in Brian Millier's article "Noncontact Infrared Thermometry" (*Circuit Cellar* 175, 2005).

Photo 1 shows the inside of an ear thermometer. It uses a thermopile sensor with a built-in reference thermistor. The thermopile provides a voltage related to the difference in temperatures between its hot and cold junctions, and the thermistor reports the temperature of the cold junction. A microcontroller combines these two values using the appropriate mathematical formulas to produce an accurate temperature reading of the hot junction. When you insert the thermometer properly into your ear, this reading will represent your internal body temperature. I wanted to see how easy it was to use the sensor's output directly, so I measured its output while varying the input temperature over the thermometer's stated operating range of 93.2°F to 109.4°F. I've read about certain Buddhist monks who can vary their body temperature while deep in meditation, but that wasn't very practical for my tests. Instead, I found I could aim the sensor at my desk lamp from a few inches away and cover the entire range by slightly varying the distance. The results of the experiment are shown in Figure 2. The curve has a slope of approximately 50 µV per °F.

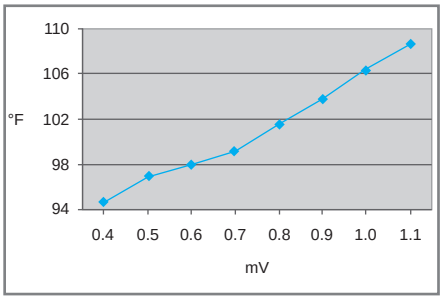


Figure 2—Sensor output of the ear thermometer. The nonlinearity in the curve isn't from the sensor. It came from slight variations in ambient temperature during the experiment as I moved the thermometer closer or farther away from the desk lamp.

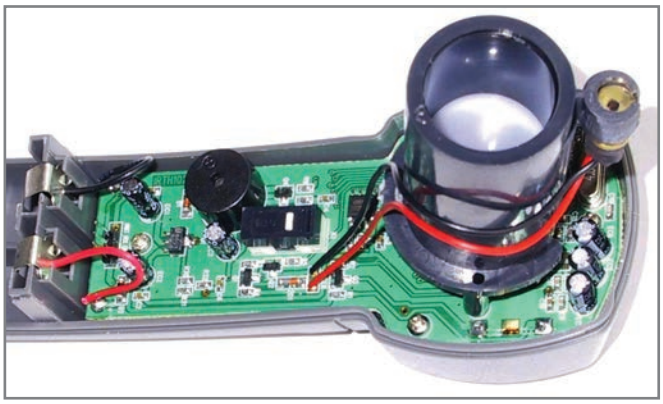


Photo 2—A general-purpose noncontact thermometer. The Fresnel lens focuses incoming IR energy onto the sensor mounted underneath it. The copper-colored laser diode module (far right) points out the center of the area being measured.

The liquid crystal display (LCD) shows the temperature to a tenth of a degree, which would be a resolution of 5 µV. I find that rather impressive—there must be quite a bit of averaging being done to reduce noise. If you extrapolate the curve to the left, it would hit 0 mV at about 88°F. That's a reasonable value for ambient temperature, since the thermometer had been sitting under the desk lamp for a while by the time I started taking measurements.

You can also find thermopile sensors in general-purpose IR thermometers. Photo 2 shows one such device. The sensor is hidden underneath the plastic lens, but it's similar to the one used in the ear thermometer. This model has a laser pointer to locate the center of the sensing area. Thermopiles are also used in gas analyzers, where relative amounts of different gases can be measured by looking at the variations in IR absorption at different wavelengths.

PYROELECTRIC DETECTORS

A pyroelectric detector is a capacitive sensor that changes its polarization in response to a change in temperature. This results in a change in surface charge $\Delta Q = A \times p \times \Delta T$, where A is the sensor area, p is the pyroelectric coefficient of the material, and ΔT is the temperature change. It's inherently an AC-coupled device, so it needs some external process to cause the sensed temperature to change in order to produce any output. This makes it perfect for motion sensors, but it's also used for thermometers that use a mechanical shutter to produce a controlled change in input temperature.[3]

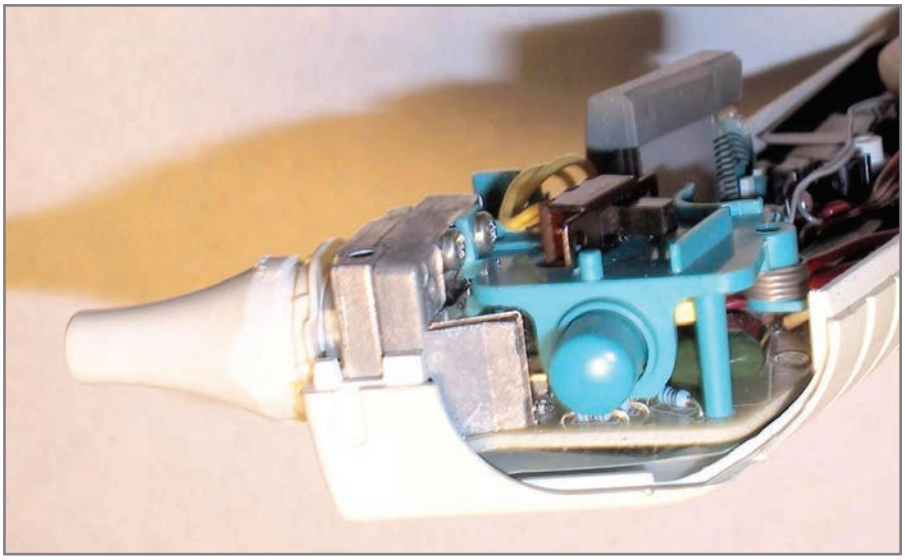


Photo 3—The sensing mechanism of an ear thermometer. It uses a pyroelectric sensor with a pushbutton-activated mechanical shutter. The metal block stabilizes the temperature of the entire mechanism.

Photo 3 shows the business end of an ear thermometer with a pyroelectric detector. This particular model is a few years old, but it serves as a useful example of detector operation. Normally, the detector is hidden behind a shutter that's mounted inside the metal block located to the left of the button. A thermistor mounted within the block tracks the temperature of the block and shutter. When you press the button, a spring-loaded mechanism snaps the shutter out from in front of the detector, giving it a clear view of whatever the probe is aimed at. The detector will output a signal that's a function of the difference in temperatures.

Figure 3 shows the thermometer's analog front end. It uses a dual-slope analog-to-digital converter (ADC) made up of a multiplexer IC and an op-amp integrator. You can follow the measurement process in Photo 4. Normally, the system repeatedly measures the temperature of the thermistor (SEL2:0 = 4,5) and the pyroelectric detector (SEL2:0 = 1,2). The cursors show when I pressed and released the Measure button. When pressed, the system immediately resets the integrator for 100 ms (SEL2:0 = 0) and then integrates the detector input for 200 ms (SEL2:0 = 1). Then it integrates a reference voltage until the comparator output goes high (SEL2:0 = 2). The input voltage is related to the integration times and multiplexer input resistors as follows:

$$V_{IN} = -V_{REF} \times \frac{R_{IN}}{R_{REF}} \times \frac{t_{REF}}{t_{IN}}$$

This formula applies to a steady input voltage, but you'll notice that the input isn't constant during the integration period. Rather, it is decaying with an RC time constant determined by the sensor capacitance and the sensor input op-amp stage's input resistance. Since this time constant is fixed, and the ADC's integration time always starts 100 ms after the shutter is opened and lasts for 200 ms, it's possible for the microcontroller to compute the value of the initial peak. It uses this value along with the thermistor value to compute the temperature.

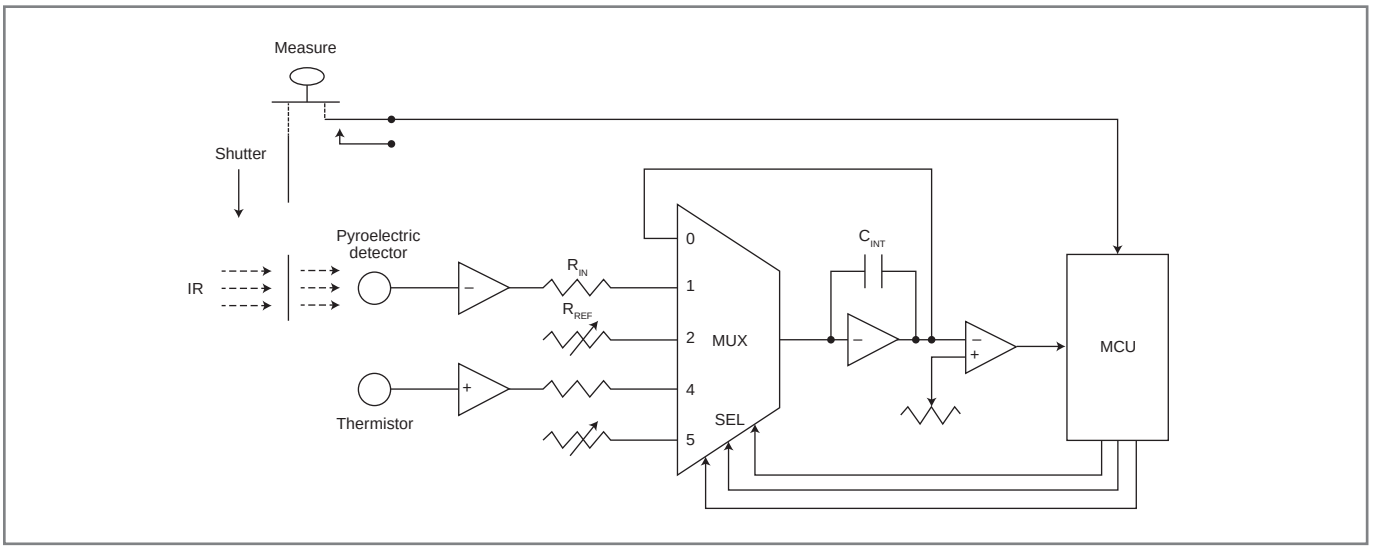


Figure 3—The pyroelectric ear thermometer's analog input circuitry and dual-slope ADC. Calibration is done using several adjustment pots.

Finally, let's look at a motion sensor, which is a more natural application for a pyroelectric detector since here the AC-coupled output is an advantage. The detector is slightly more complex, as it contains two pyroelectric elements connected in series back-to-back inside the package, separated by a fixed distance. Some detectors, including this one, also include an FET, which can be connected as a source follower to lower the detector's output impedance. In a motion sensor, the detector is mounted behind a Fresnel lens whose multiple segments alternately focus any incoming IR energy on one element then the other as a person walks by. Photo 5 shows this process as I waved my arm in front of a sensor. The SENSE signal normally sits at 1.04 V, and you can see it oscillating around this value as the lens alternately focuses the thermal energy from my arm onto each of

Gigabit Technology
• ARM-based
• System on a Chip
• Gigabit Ethernet
• Small, Cheap, Fast
• Both QFP and BGA Packages
• Standard Development Tools
• 10 Year Life Guarantee
• Royalty Free RTOS, TCP/IP V4/6
$10.00 each (Qty 100)
The gridARM™ System on a Chip (SOC) is a high performance, low cost, low power, highly integrated single chip with 10 / 100 / 1000 Mbps Ethernet, USB, CAN, Serial, SRAM Memory, SPI, I2C, RTC and internal peripherals designed to provide a complete solution for embedded applications.
THE NETWORKING EXPERTS
gridconnect
800.975.4743 USA • 1 630.245.1445
gridconnect.com/gridarm.html
Leaders in the embedded and networking marketplace providing network hardware, high quality software and services

System on Module
New - SoM-3517
• TI ARM Cortex-A8 600 MHZ Fanless Processor
• Up to 512 MB of DDR2 SDRAM
• UP TO 1gb of NAND Flash
• 2 Full Speed USB 1.1/2.0 Host ports
• 1 Full Speed USB OTG port
• 4 Serial Ports, 2 I2C and 2 SPI ports
• Processor Bus Expansion
• 10/100 BaseT Fast Ethernet
• CAN 2.0 B Controller
• Neon Vector Floating Point Unit
• 16-bit DSTN/TFT LCD Interface
• 2D/3D Accelerated Video w/ Resistive Touch
• Small, 200 pin SODIMM form factor (2.66 x 2.375")
Windows CE
2.6 KERNEL
The SoM 3517 uses the same small SODIMM form-factor utilized by other EMAC SoM modules, and is the ideal processor engine for your next design. All of the ARM processor core is included on this tiny board including: Flash, Memory, Serial Ports, Ethernet, SPI, I2C, I2S Audio, CAN 2.0B, PWMs, Timer/Counters, A/D, digital I/O lines, Video, Clock/Calendar, and more. The SoM-3517M additionally provides a math coprocessor, and 2D/3D accelerated video with image scaling/rotation. Like other modules in EMAC's SoM product line, the SoM-3517 is designed to plug into a custom or off-the-shelf Carrier board containing all the connectors and any additional I/O components that may be required. The SoM approach provides the flexibility of a fully customized product at a greatly reduced cost. Contact EMAC for pricing & further information.
http://www.emacinc.com/som/som3517.htm
Since 1985 OVER 25 YEARS OF SINGLE BOARD SOLUTIONS
EMAC, inc.
EQUIPMENT MONITOR AND CONTROL
Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: www.emacinc.com



Photo 4—The ADC signals during a temperature measurement. You can see the multiplexer input select, comparator output, and integrating capacitor voltage. VSNS shows the pyroelectric sensor input signal shifting as I press and release the Measure button, indicated by the cursors.

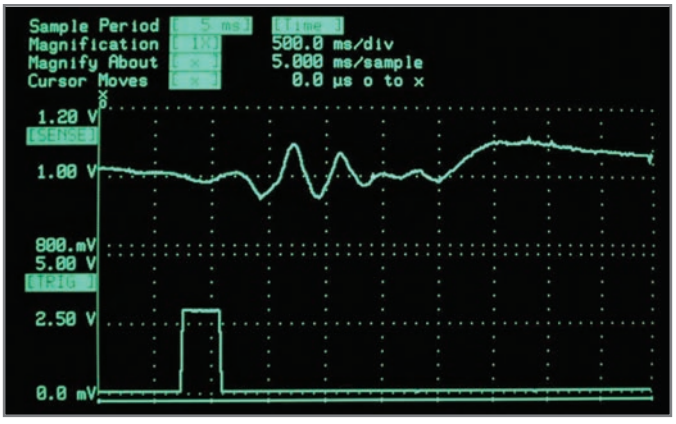


Photo 5—The motion sensor in action. SENSE is the pyroelectric detector output driving a 47-kΩ resistor and amplified by 100. TRIG shows when the output first exceeds the motion detection threshold.

the detector's two sensing elements. As the energy hits the first element, the detector's output goes positive momentarily. Then, as it leaves that element and hits the second one, the voltage across the first element goes negative while the second goes positive. The inverse series connection means these voltages will add, effectively doubling the output voltage. It's still not very large, only around 1 mV to produce the 100-mV peak shown in the photo.

COOLING DOWN

I hope I have helped take some of the mystery out of thermal IR sensors and their everyday applications. I'm ready to do more exploring now that I have a handful of sensors to work with. Maybe you are now inspired to take a closer look at some firsthand.

Richard Wotiz has been taking products apart ever since he was old enough to pick up a soldering iron. He's been helping others put them back together since 1991, when he started his design consulting business. Richard specializes in hardware and software for consumer products and children's toys. He can be reached at rw601@spiraltap.com.

REFERENCES

[1] D. DeWitt and G. Nutter, "Theory and Practice of Radiation Thermometry," Wiley-Interscience, 1988.

[2] C. Féry, "Thermoelektrisches Pyrometer," German Patent 135064, 1901.

[3] J. Fraden, "Infrared Electronic Thermometer and Method for Measuring Temperature," U.S. Patent 4797840, 1989.

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*. For more need-to-know information about some of the topics covered in this article, the *Circuit Cellar* editorial staff recommends the following content:

—

Plastic Extruder Thermal Analysis
by Ed Nisley
***Circuit Cellar* 251, 2011**

Measuring and understanding the thermal performance of the Thing-O-Matic 3-D printer's extruder head provides some interesting results. Topics: Thermal Analysis, Extruder, Thermocouple, Calibration, Insulation, Thermal Resistance

—

Thermal Performance
by Ed Nisley
***Circuit Cellar* 249, 2011**

Understanding the principles of heat transfer and having the ability to work out heat transfer problems are essential skills for any serious electronics engineer. The techniques detailed in this article will enable you to figure out the origin of a project's heat and how to measure its progress. Topics: Heatsink, Heat Transfer, Dissipation, Energy Management, Thermal Resistance, Cooling, Derating

—

Noncontact Infrared Thermometry
by Brian Millier
***Circuit Cellar* 175, 2005**

Noncontact infrared thermometry has enabled doctors to get rid of mercury thermometers. This technology also enables distributors to monitor the temperature of food products without contaminating them with sensors. Topics: Temperature, Infrared, Thermometry, Radiation, Thermopile

—

Go to *Circuit Cellar*'s webshop to find these articles and more: www.cc-webshop.com

NEW APP!
Download at iTunes

Introducing the CC Electronic Toolbox, for iPad and iPhone.

- -Ohm's Law calculator
- -LED/resistor calculation
- -Op-amp circuit calculation
- -Frequency, period wavelength calculation
- -Databases for transistors, FETs, Triacs & more
- -Database for common electrical Symbols
- -PLUS, a CC Tech Guide...

& so much more!



Specially designed for electronic engineers, the CC Electronic Toolbox app is an essential and convenient tool for any iPhone, iPad or iPod Touch user. This need-to-have app provides easy access to essential electrical information and reference material, as well as multiple conversion tools.

www.circuitcellar.com/toolbox

by George Martin (USA)



Printer Port Connections

You've been asked to upgrade an old industrial control system that uses a printer port as a high-speed parallel I/O link to custom electronics contained in an operator's console. But the printer port is dead, right? This article details a few design plans for bridging the gap between old and new technology to meet your project's contemporary requirements.

I'm in the process of finishing up an interesting, fun, and sometimes risky project. The interesting and fun part was working with the customer creating system requirements and writing C code. The risky part was relying on someone else's C code for the success of the project. And that person was long gone and not available for consultation. Let me explain.

THE SYSTEM

The existing system is an industrial controller. What it is controlling is not important to the technical topics I'm going to cover and its discussion (while interesting) would not add very much understanding to the project. Let's just say it's a big console table top that has switch inputs and display outputs. And I'll just refer to this as the operator's console.

The present implementation uses a PC as the controller in the operator's console. It has all the usual keyboard, mouse, and display as input and output devices. There's an RS-485 communications bus to remote devices. The printer port is used as a high-speed parallel input/output (I/O) link to custom electronics contained in the operator's console. This high-speed I/O link scans the keys as inputs and updates custom bar graph displays as outputs. This system has been used in this manner ever since the original IBM PC was introduced way back when.

THE PRINTER PORT

For decades, designers have used the printer port to connect to all sorts of devices that were not printers. What this port offers is an 8-bit bidirectional bus with handshake signals along with control and status bits. I'm working on a mini milling machine for the shop and the stepper motors are controlled with the printer port (very similar to the MakerBot Thing-O-Matic Ed Nisley wrote about in "Plastic Extruder Thermal Analysis," *Circuit Cellar* 251, 2011). As operating systems (OSes) have evolved and tried to make networks more secure, the designers attempted to eliminate the programmer's access to the I/O ports on the PC. And with every increase in OS security, we designers have figured out ways around these restrictions and can directly access the printer ports. Well, with Microsoft Windows 7, it seems the OSes appear to have won. It looks like you cannot directly access the hardware and you don't even have a parallel port on the newest PCs. Sure, you could still buy one as an add-on card, but the handwriting is on the wall. The printer port is dead.

THE PROBLEM

The existing system has a Microsoft Visual Basic (VB) program interfacing to a dynamic linked library (DLL). The DLL provides the lower-level interface to the custom hardware connected to the parallel port. The DLL could be thought of as a driver library. It was written in C and was

compiled into a DLL instead of into an executable file. When the VB program made a call to entry points in the DLL, that DLL was loaded into memory if not already in place and executed. The DLL was designed to be efficient in its manipulation of the printer port hardware. The DLL also had routines (entry points) that were exposed to the calling VB. All other internal routines and variables were kept hidden.

Without a printer port, there is no way to communicate with the operator's keyboard and display. We brainstormed several design approaches and selected the simplest one.

A NEW DESIGN: PLAN A

It's good to have a plan, so we came up with this one. The plan was to just replace the parallel port hardware with a USB-to-parallel-port converter. It was a piece of cake, and cost only about $30. We got one and installed it. But it was looking for characters to send to a printer (as in printing a file) and not the "bit banging" that was going on in the VB and DLL code. As we looked into the DLL code, we saw commands that manipulated bits on the data bus (8 bits) and the control output signals. That DLL then read the results on the status input bits. There was no way the USB-to-parallel-port converter was going to work.

A NEW DESIGN: PLAN B

It's good to have a backup plan, and we also had one of those. We could use a USB connection from the PC to a microcontroller, develop a set of commands and responses to send over the USB, and then develop custom hardware to interface to the existing operators console hardware. That would be a fairly straightforward design. Its input and output were very few bits because of the way the switches and displays were wired. The keyboard is wired in a 24 × 1 pattern. So 5 bits from the printer data bus are used to select a switch and 1 status bit is used to read the results. Remember, we only have the data, status, and control bits for the original PC. But then we would need to write the software to operate the existing hardware. Therein lies a bigger problem. There are several versions of the hardware out in the field. The product line is more than 30 years old. We would never be able to get the interface requirements for all those variations, and we would never be able to test all the combinations. It's just too much work.

A NEW DESIGN: PLAN C

As I said, it's good to have a backup plan…but perhaps I repeat. We next proposed that the microcontroller interface to the PC using USB and use the microcontroller's I/O ports to replicate what the parallel port interface looked like. One port is the data bus, another is the status, and another is the control. Then the microcontroller could just run the DLL code. The DLL was all written in C and took care of all the interfacing to all the 30 years of product line. Could this ever have a chance of working?

It seemed like a possible approach. The input to the DDL code consisted of calls to routines with well documented input and output parameters. And we had the VB code that made those calls and could use it as a debug tool. We could single step through the VB code, but we couldn't single step when it called the DLL. We could then use logic analyzers to capture the data on the parallel port connector.

The input was calls by the VB code to the DLL and the output from the DLL was commands to read and write the parallel port. It seemed like we had the problem bounded. We would need to write a new set of commands that were sent across the USB and received by the microcontroller. We called these packets. We would have to rewrite the I/O routines on the PC to use these packets. Also we needed to convert the I/O code in the DLL to I/O code on the microcontroller.

The design rules were: "Do not touch the DLL code!" Simple enough. The joke was, if you touched it, you owned it. I would add comments in the DLL code to help understand what was going on. My engineering contact for the customer would make a note that I touched a module and now I was responsible for that module. I would then comment out my comments and then say I didn't touch anything. Great fun and it kept us both on track. The style of C coding in the DLL was not one I use, and was also inconsistent throughout the modules. This is understandable for 30-year-old code. I so much wanted to "clean it up," but I resisted.

THE NEW HARDWARE

I selected a member of the Stellaris family of processors from Texas Instruments (TI). It has a USB port, several I/O ports, onboard flash, and RAM. I used TI's Code Composer Studio (CCS) integrated development environment (IDE) for the development environment. For this project, I was using CCS V3 and upgraded to CCS V4. Version 4 is awful. That's not fair of me to say, and I've not defended my position. Just trust me on this one. I also have a solution. CCS V5 has been released and it addresses many of my criticisms. Wait for that version.

I selected an evaluation board that let me compile, download, and do unit testing on my code. TI has libraries for the USB device, and I was able to take one of their examples (USB serial port) and use it with few changes. The changes were deleting references to the display and UART routines that come on the evaluation boards.

My first step was to get the CPU and timer interrupts up and running so we could blink a light-emitting diode (LED). Once that was done, I added the USB code and the DLL code. I created three directories, one for the USB code, one for the DLL code, and one for all the new code. Once all the code was in, I tried to compile. Of course I got errors, lots of errors. Most of the errors were simple to fix. The USB code was referencing a display and all I needed to do was delete those references. Then variables came up as never used, so I deleted them. The DLL code was creating application programming interfaces (APIs), something specific to generating a DLL. That was also easy to identify and I commented it out. In all these cases, I put the following statements around the offending code:

```
#ifdef     NEVER
#endif
```

Again, Rule #1: You touch it you own it. Using this technique, it was easy to run comparison programs on the source and make sure all the changes were as intended.

Next we came to more complicated issues, mostly in the DLL source. This code was written more than 30 years ago. And, as I said, the style was inconsistent. This is not a slam on the author, but more a statement as to how C was coded back then. We know better now. For example, names for constants were not in all capital letters. I really wanted to fix this, but then I would own it. I also came across type casting for variables such as "unsigned long." I know what that means in the TI Stellaris compiler. But what did it mean in the compiler used for the DLL? Do they need 64 bits, or perhaps 32, or maybe only 16? I decided I would not change any of the sizings of these variables and routines. Perhaps there is some critical reason that an "unsigned long" is needed. I don't think so, but changing them one by one could lead to a real mess and perhaps disaster. Also, I was sure (seeing the VB code in operation) that I knew the sizes of all the variables used to call the code and the sizes of all the return variables. If I just kept all these parameters under close supervision (as you do with children), then it would be okay.

I converted all the I/O in the DLL to I/O on the microcontroller. This was straightforward enough and in fact isolated to one code module that I now owned. The data and control ports are only written, while the status port is only read. Also, delay routines were sprinkled throughout the DLL code. It became clear we only needed delay routines when we wrote data to the ports. Delays were set to be just long enough to let the signals settle out.

Next I created the packets that would be used by the VB code to talk to the microcontroller. We tried to keep the packets simple. All the packets took the form of a Start of Packet character, Packet Type character, Payload, and End of Packet character. We didn't do error checking and relied on the USB to correctly transfer the data. Data and commands were sent in both directions on the USB using these packets. The microcontroller did not initiate any packets; it only responded to packets received.

It took two sessions of debugging to get the packets working. The original TI USB code emulated a serial connection. On the microcontroller side, USB data was received in the USB RxBuffer. I could see the data, but when I called the routines to get a character out of the buffer, the response was "no data available." Well, the original code used interrupts to receive USB data (good) and then generate UART interrupts to send that data out to the UART (not good). I defeated the UART portion of the interrupt routines and—ta da—I was reading data.

We could now read the command to scan all the keyboard input keys and report the results. I found the usual "off by one problems" in indexing my arrays. The hardware design wanted to see 0 for key number 1. These were straightforward to find and fix.

We're not completely done testing code. But it was clear the design approach was stable, so we switched our efforts to completing the printed circuit board (PCB) design and parts lists for the prototype build. But that's another story. I suspect you might come across the problem of the missing printer ports in the next few years. Perhaps this is a road map to solving a similar problem.

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.

RESOURCE

E. Nisley, "Plastic Extruder Thermal Analysis," *Circuit Cellar* 251, 2011.

SOURCES

Windows 7 and Visual Basic
Microsoft Corp. | www.microsoft.com

CCStudio IDE
Texas Instruments, Inc. | www.ti.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*. For more need-to-know information about some of the topics covered in this article, the *Circuit Cellar* editorial staff recommends the following content:

—

A Practical Parallel CRC Generation Method
by Evgeni Stavinov
***Circuit Cellar* 234, 2010**

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. Topics: CRC, Generation, Cyclic Redundancy Check, Verilog

—

Create a USB Virtual COM Port
by Jan Axelson
***Circuit Cellar* 217, 2008**

You can use almost any microcontroller with an embedded or external full- or high-speed USB controller to design and program your own USB virtual COM port device. Topics: USB, Virtual, COM, Port, Serial Port

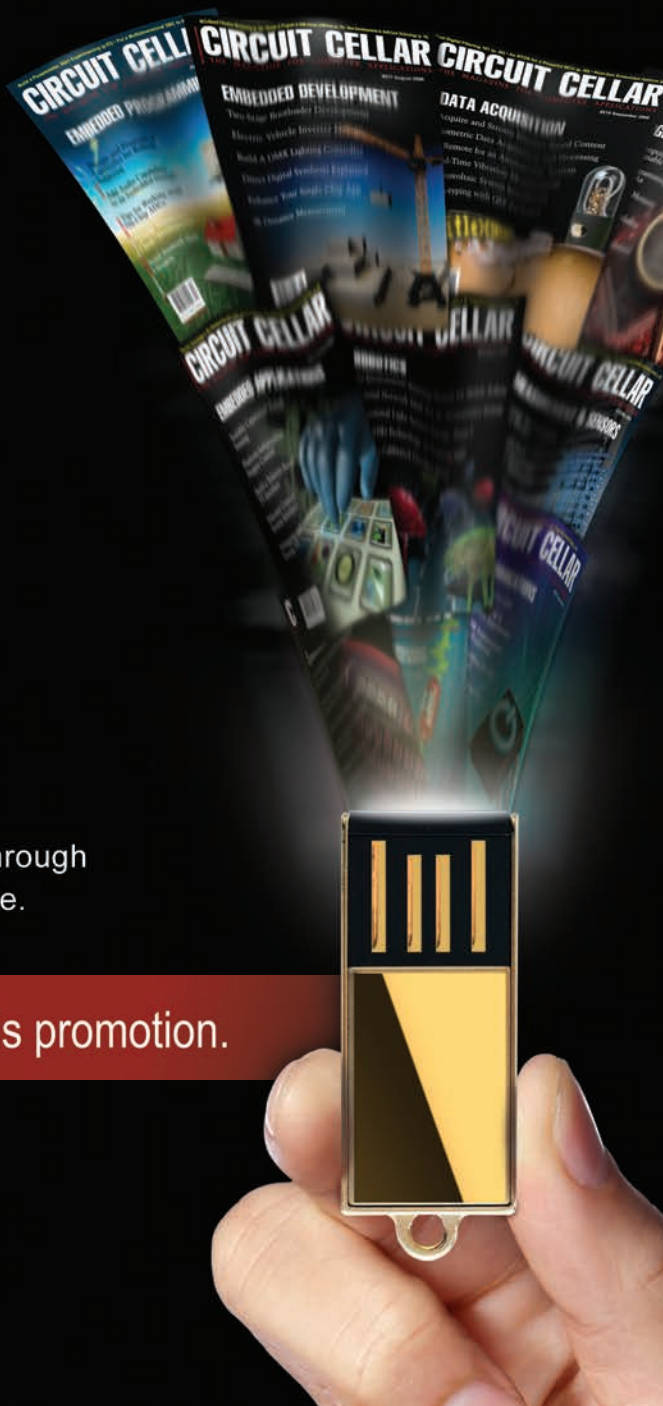
—

Go to *Circuit Cellar*'s webshop to find these articles and more: www.cc-webshop.com

A GOLDEN OPPORTUNITY

The price of gold just went *down* at CIRCUIT CELLAR!

Purchase Circuit Cellar's USB archive, **CC Gold** at $25.00 off the regular price and load it up with select add-ons for only $5.00 each.



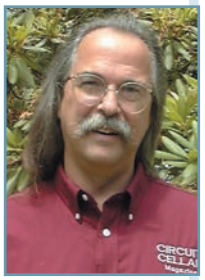
Available add-ons:
Previous Contest Projects
Elektor Back Issues
audioXpress Back Issues

Archive includes:
All Circuit Cellar issues in pdf format that are in print through date of purchase. Also includes associated project code.

Free upgrade to 16 gigabyte USB during this promotion.

Limited time offer.
For details and to purchase visit:
www.cc-webshop.com

by Jeff Bachiochi (USA)



Wheel-Free Mobile Robots

Want to look under the hood of a mobile mechatronics robot? This article covers everything from the interactive programming environment to the robot's hardware design.

In 1999, the Trinity College Fire Fighting Home Robot Competition had a new visionary. While other teams concentrated on getting their wheeled vehicles tuned up for the fastest time possible, one individual's robot literally stood out from the masses. Acroname's Mark Whitney won special recognition for Stampy, the first robot without wheels to enter this competition and attempt to find and put out a "fire" amongst the maze of rooms. While Stampy didn't run away with the grand prize, the walking robot changed the way competitors viewed the problem and opened the door to a flood of alternative locomotion robots.

Many robots today don't have wheels or legs—in fact, most industrial robots consist of only an arm! Take a poll of the general public however, and I'll bet the configuration most people are familiar with is the bipedal robot similar to C3PO from *Star Wars*. Stampy and its heirs are made with radio-controlled (RC) servos to give controlled movement to each joint in the skeletal structure of the robot. While one servo can provide one-dimensional movement like a knee, ball-type (2-D) joints like an ankle, hip, shoulder, wrist, and neck, are much more complicated. Many companies now offer a large selection of servo-driven robots that resemble Optimus Prime from *Transformers*. Those robosapiens, with their sufficient joint control, can actually get up if they fall over.

RC servos are relatively small, powerful, inexpensive, and can be easily controlled using a 1-bit pulse width modulation (PWM) output so they are ideal for use with a microcontroller.

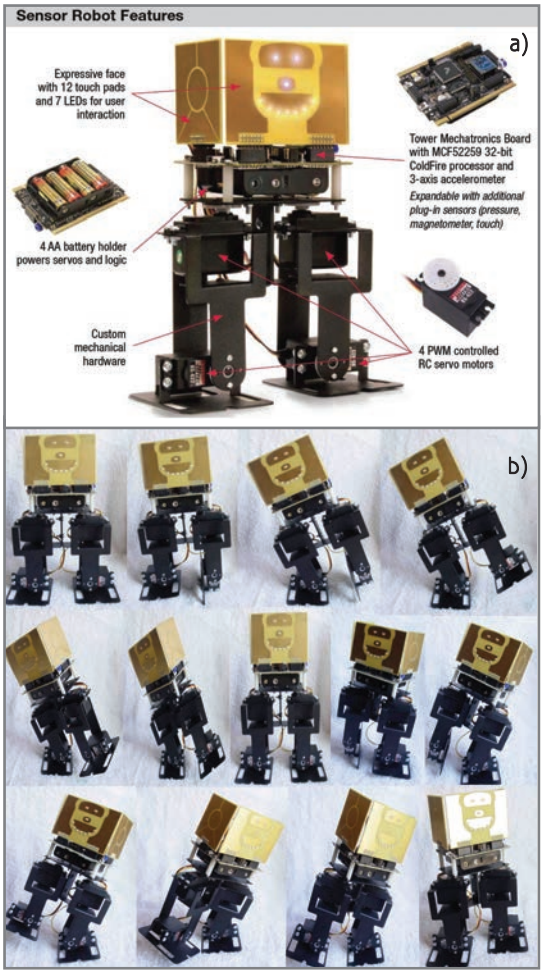


Photo 1a—The mechanical elements that make up the $199 Freescale FSLBOT include the TWR-MECH board (available separately). (Source: Freescale Semiconductor, "Freescale Tower System: Tower Mechatronics Board and Robot Datasheet," 2011.) **b**—This series of photos shows the individual moves necessary for the robot to walk ahead one pace.

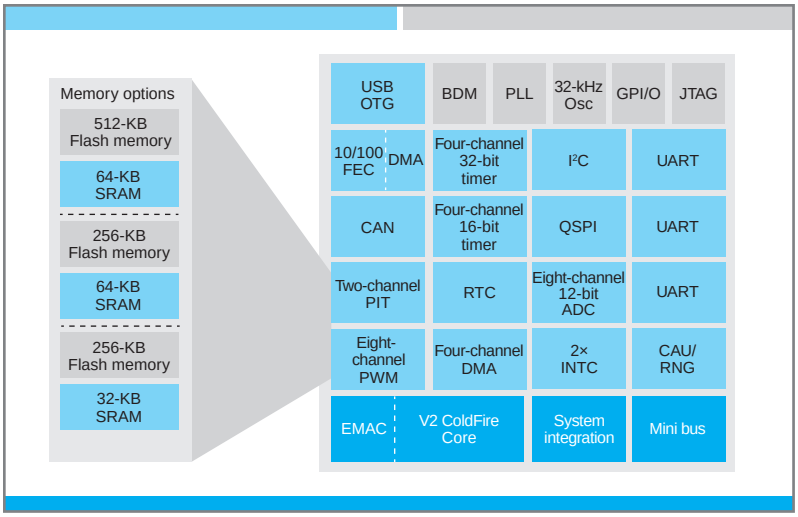


Figure 1—The TRW-MECH board is based on Freescale's 32-bit MCF52259 ColdFire processor. This flash-based device has lots of internal goodies.

You'll find most of these full-movement robots will use six to eight servos just for the legs.

TOWER OF BABEL

A friend of mine recently sent me a link to the Freescale Semiconductor website. The webpage I viewed showed an armless bipedal robot using only four RC servos (see Photo 1)! It seemed a little odd for Freescale to be advertising a robot. By searching around a bit on the website, I found that the microcontroller board used for the FSLBOT: Freescale Mechatronics Robot was one of many offerings for their Tower System, which is a modular development platform for 8-, 16-, and 32-bit microcontroller (MCU) and microprocessor (MCP) modules. The use of a passive backplane or "elevator" tower provides any of these MCU/MCP boards access to Freescale's array of peripheral modules designed for the Tower System.

COLDFIRE

One such MCU module, Freescale's TWR-MECH board, features a Freescale MCF52259 ColdFire 32-bit processor supporting USB, RS-232, a SPI bus, eight PWM outputs (suitable for RC servos), and an I²C bus (see Figure 1). Most of the processor's I/O runs out to the two elevator edge connectors along the front and rear of the printed circuit board (PCB). However, there are many goodies populated on the TWR-MECH board. The SPI bus has a 64-Mb flash memory chip and the I²C bus has an MPR121 TOUCH controller and multiple PCA9554 I/O expanders. The MPR121 supports up to 12 touchpads for user input. The PCA9554s control LEDs and other I/O used in support of the two on-board sensor sockets. One of these was populated with an LFDA7455, a three-axis accelerometer. The sensor sockets make it easy to investigate any of the growing base of sensors Freescale offers. Many of these are tiny SMT devices that would otherwise be difficult work with.

TOOLS

I began my look into the FSLBOT by checking out the tools necessary to use the ColdFire MCU. As it turns out, the FSLBOT comes preprogrammed with a BASIC operating system, CPUStick.com's StickOS. This provides a user access to the robot via ColdFire's USB port using just a serial terminal program. No additional applications are necessary. In fact, StickOS can be updated via the USB connection as well.

Before I dive into how simple it is to use the StickOS BASIC interactive programming environment to investigate the peripherals on the FSLBOT, I want to touch on the lower-level tools you can use with the ColdFire processor, assuming you'll want to replace StickOS with your own application. While there is a move by many, including Freescale, to use the Eclipse framework as a universal integrated development environment (IDE), I began with CodeWarrior V7.2, which is offered for free with the FSLBOT.

The TWR-MECH board also has programmed MC9S08JM60 MCU to support live debugging and application flashing to the ColdFire processor. With the proper USB driver, I was able to reflash the TWR-MECH with the StickOS application through this second USB port. I wanted to make sure I could get back to square one if, for any reason, I lost the StickOS application. Using this USB port also gave me some feel for the CodeWarrior IDE you will want to use for serious development after you have had your fun investigating the FSLBOT's peripherals using the StickOS application for programming in BASIC.

WHY BASIC?

It may seem a bit odd for a company like Freescale to offer a bipedal robot that is programmed in BASIC. I hope their logic catches on. A great datasheet can only go so far in enticing an engineer to design a part into a project. Without real hands-on experience, many will choose an inferior part just because it is what they are familiar with. To succeed, a company must get parts into the hands of engineers and these parts must be easy to evaluate. And, by the way, if all this learning can be fun too, then the chore turns into entertainment. For those of you who don't like fun, well, just remove the TWR-MECH board from the FSLBOT and you've got a great base for serious development without the fun of teaching your newbie how to walk.

If you're older than a half century, you probably remember executing your first BASIC program, something like, "PRINT 1 + 2." Seeing the number "3" popping up on the screen was a wonder to behold, as it came from your two fingers typing on the keyboard. BASIC is simple to use. StickOS is written in C, but like any good application, you can't tell (or don't care). It presents a user-friendly way of accessing the hardware with simple commands that are

translated into machine-meaningful code to efficiently obey your every whim. Table 1 lists the commands available with StickOS V1.90.

START WITH I/O

Every application that involves a user must have some way to communicate with the user. In our case, when the FSLBOT is tethered to a terminal program via USB, the user can type in (or download) BASIC program statements and StickOS will respond. This is a form of interactive I/O that isn't necessarily available if the robot is running a BASIC program on its own, untethered. Therefore, it is important that there be other ways to communicate with the robot. The FSLBOT uses seven LEDs as output devices. These are located on the face panel of the robot in the shape of a smile. An additional three LEDs, a nose and two eye LEDs, reflect "powerON." While the robot does not have ears per se, two side panels have four touchplates on each side that can sense a finger (the face also has four touchplates for a total of 12). Both the LED

BASIC line editor	
	ANSI or VT100ish terminal support
BASIC compiler	
	Compiles to a fast and safe intermediate byte code
	Transparent line-by-line compilation is invisible to the user
	Integer variable/array support
	String variable support
	Block-structured programming and subroutine support
Interactive BASIC debugger, supporting:	
	Breakpoints, assertions, and watchpoints
	Live variable (and pin) manipulation and examination
	Execution tracing and single-stepping
	Edit and continue!
BASIC performance profiler	
	Trivially see where your program spends its time!
BASIC file system	
	Load and store multiple BASIC programs in flash
2.4-GHz ZigFlea wireless transport	
	Remote control via a telnet/rlogin-like interface
	Remote variable access in BASIC
	Wireless BASIC program update
	Wireless StickOS firmware upgrade
USB Host mode (on selected MCUs)	
	Log StickOS "print" statements to external USB flash drive
External control of MCU I/O pins, implicit through "pin variables"	
	Digital input or output
	Analog input or output (PWM actually)
	Servo output
	Frequency output
	UART input or output
	I²C master input and output
	QSPI master input and output
	4-bit HD44780-compatible LCD output
	4 × 4 scanned keypad input
Internal peripheral control	
	Interrupts delivered to BASIC handlers!
	Interval timers, DMA timers, ADC, PWM, UARTs, I²C, QSPI, etc.
	Direct MCU register access from BASIC for low-level control through MCU register variables
Internal flash memory control	
	Save programs and parameters to flash for standalone operation
	Prolong flash lifetime by storing incremental updates in RAM
	Clone one MCU's flash directly to another
	Upgrade StickOS firmware via terminal emulator!
	No external flash programmers needed!

Table 1—The StickOS commands are quite extensive and include functions specific to the hardware (devices). Multiple programs can be saved and automatically run at power-up. While the FSLBOT has four AA batteries, you can write, download, program, and debug powered by the USB connection (except actually driving the servos). This saves the batteries until you want to run the servos or operate untethered.

Register	Protocol	Function	Default
0	Read byte	Input Port register	Unknown
1	Read/write byte	Output Port register	Logic 1
2	Read/write byte	Polarity Inversion register	Logic 0 (noninverted)
3	Read/write byte	Configuration register	Logic 1 (inputs)

Table 2—The I/O expander that controls the seven LEDs is a registered I²C device located at address 0x5A. There are four registers in this device: Register 0 can be read to sample the eight inputs of the port. Register 1 can be read or written and hold data affecting any I/O bit that has been defined as output (via the configuration register). Register 2 can be used to invert any I/O bit and make it look like the opposite logic state. Register 3 is used to define the function of each bit (1 = input, 0 = output).

outputs and the touch inputs are accessed via peripherals on the I²C bus. BASIC I²C commands make it easy to communicate with these peripherals.

Let's look at a simple program to turn on the (mouth) LEDs. Referring to Table 2, because the default values of this device's configuration register are all logic 1 (inputs), we need to write logic 0 to each bit that we want to be an output (bits 1 through 7). Note the output register default is already logic 1 (LEDs pulled up to logic 1 are turned ON), so we need only initialize Register 3 with the I²C commands, `i2C start <address>`, `i2c write <register>, <value>`, and finally `i2c stop`.

```
10      i2c start 0x5A
20      i2c write 0x03, 0x01
30      i2c stop
40      end
```

Note this sequence of I²C commands (start, read/write, stop) is required for each I²C communication. Without line numbers each command is executed as typed. When line numbers are used this becomes a program and can be executed with the command `<run>`. I've purposely not used any variables for the `i2c_address`, `i2c_register`, and `i2c_values`, to simplify the code as much as possible. You can find all of the demo code I've written for experimenting with the LEDs, touch inputs, servos, accelerometer, and compass on *Circuit Cellar*'s FTP site.

You might ask, why use so many peripherals when the processor can do all of this with its own I/O? Actually, the TWR-MECH board was designed to show off Freescale's peripherals that can be added to any design. In this case, this leaves the processor and its I/O unencumbered for an engineer to fully use as they see fit.

WALK THE WALK

Can the FSLBOT actually walk with only four servos? There are only two joints per leg on this robot. Each leg has a hip joint that enables the leg and foot to rotate slightly toe left and toe right of the normal toe forward position. Each foot has an ankle joint that enables the foot to roll sole outward and sole inward of the normal sole flat position. You can download the code I've experimented with from *Circuit Cellar*'s FTP site. Instead of explaining the actual code here, I want to give a bit of insight about what you need to think about when developing an application that enables the FSLBOT to walk.

To take a step, there are three basic moves, lean onto one foot, swing body, and return to both feet. When starting with both feet together, your first step is really only a half step. The second and subsequent steps, beginning with one foot ahead of the other, can be a full step to get the other leg in the lead or a half step to end up with both feet back together. So movement of the hip joints control whether you are making a half or full step, while the ankle joints control the leaning of the body onto one foot or the other.

One of two things can happen when you attempt to lean onto one foot. You want the robot to lean raising the opposite foot off of the ground. However, depending on the weight distribution, just pivoting the ankle might not lean the robot (lift the other leg). Instead, the foot may rotate the sole outward leaving both feet on the ground. To prevent this from happening, you must give the body a push with the opposite foot. By moving this foot (sole inward) the leg is lengthened helping the body to lean. Once it is leaning appropriately, you may return the ankle joint of the lifted leg to neutral so the body can move freely when the hip joints are rotated.

You may note that the ankles can move ±90° (to neutral) while the hip servos are mechanically restrained and can rotate only about 30° to –60°. As it turns out about 30° is all you want to rotate or the center of gravity becomes unstable and the robot goes down (and cannot get up).

Most servos have an operating range of 180°, the angle corresponds to the timing of a PWM output. A PWM on-time of 500 µs produces the minimum position and an on-time of 2,500 µs produces a maximum position. The servo command requires a number between 500 µs and 2,500 µs. To change a servo's position to what I call neutral or 90°, use the servo command with a value of 1,500 µs. You'll notice that the action of changing position with a single command will move the joint as fast as possible (limited only by the physical properties of the servo). To readily control the speed of the movement you will want to break the repositioning into multiple movements using a loop. The timing can be further slowed by including pauses inside the loop.

Refer to Photo1b for the steps I used to create a single pace, or double step (one left, one right). These photos show sequential movements of a single servo at a time. There are some minor issues with this. For instance to lean left (bring the right foot off the ground) you need to move both feet for reasons previously stated. As an ankle joint pivots (sole inward or outward) it will either slip as the opposite foot remains stationary or it will dig in and drag or push the opposite foot slightly. This can be minimized

BOARDS, BOOKS, DVDs AND MORE AT WWW.ELEKTOR.COM/SHOP

Elektor Shop

The world of electronics at your fingertips!



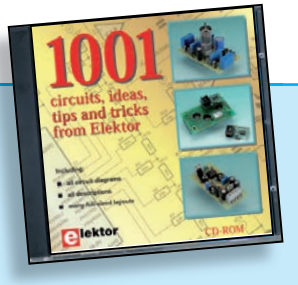
CD/DVD-ROMs

A whole year of Elektor magazine onto a single disk

DVD Elektor 2011

The year volume DVD/CD-ROMs are among the most popular items in Elektor's poduct range. This DVD-ROM contains all editorial articles published in Volume 2011 of the English, American, Spanish, Dutch, French and German editions of Elektor. Using the supplied Adobe Reader program, articles are presented in the same layout as originally found in the magazine. An extensive search machine is available to locate keywords in any article. With this DVD you can also produce hard copy of PCB layouts at printer resolution, adapt PCB layouts using your favourite graphics program, zoom in / out on selected PCB areas and export circuit diagrams and illustrations to other programs.

ISBN 978-90-5381-276-1 • $37.90



Circuits, ideas, tips and tricks from Elektor

CD 1001 Circuits

This CD-ROM contains more than 1000 circuits, ideas, tips and tricks from the Summer Circuits issues 2001-2010 of Elektor, supplemented with various other small projects, including all circuit diagrams, descriptions, component lists and full-sized layouts. The articles are grouped alphabetically in nine different sections: audio & video, computer & microcontroller, hobby & modelling, home & garden, high frequency, power supply, robotics, test & measurement and of course a section miscellaneous for everything that didn't fit in one of the other sections. Texts and component lists may be searched with the search function of Adobe Reader.

ISBN 978-1-907920-06-6 • $55.70



Elektor's Components Database

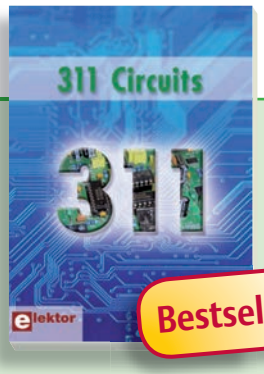
CD ECD 6

This CD-ROM gives you easy access to design data for over 70,000 components! The program package consists of eight databanks covering ICs, transistors, diodes and optocouplers. A further eleven applications cover the calculation of, for example, zener diode series resistors, voltage regulators, voltage dividers and AMV's. A colour band decoder is included for determining resistor and inductor values. All databank applications are fully interactive, allowing the user to add, edit and complete component data. This CD-ROM is a must-have for all electronics enthusiasts!

ISBN 978-90-5381-258-7 • $40.20

Prices and item descriptions subject to change. E. & O.E

Books



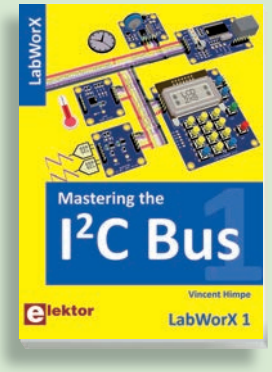
Bestseller!

Creative solutions for all areas of electronics

311 Circuits

This book contains circuits, design ideas, tips and tricks from all areas of electronics: audio & video, computers & microcontrollers, radio, hobby & modelling, home & garden, power supplies & batteries, test & measurement, software, not forgetting a section 'miscellaneous' for everything that doesn't fit in one of the other categories. 311 Circuits offers many complete solutions as well as useful starting points for your own projects. Both categories and anything in between represent a veritable fountain of inspiration for cultivating your own ideas and learning about electronics.

420 pages • ISBN 978-1-907920-08-0 • $47.60

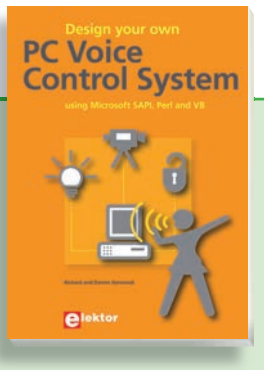


LabWorX: Straight from the Lab to your Brain

Mastering the I²C Bus

Mastering the I²C Bus is the first book in the LabWorX collection. It takes you on an exploratory journey of the I²C Bus and its applications. Besides the Bus protocol plenty of attention is given to the practical applications and designing a solid system. The most common I²C compatible chip classes are covered in detail. Two experimentation boards are available that allow for rapid prototype development. These are completed by a USB to I²C probe and a software framework to control I²C devices from your computer.

248 pages • ISBN 978-0-905705-98-9 • $47.60



Talk with your computer

Design your own PC Voice Control System

This book guides you through practical speech recognition, speech annunciation and control of really useful peripherals. It details a project which will enable you to instruct your computer using your voice and get it to control electrical devices, tell you the time, check your share values, get the weather forecast, etc. and speak it all back to you in a natural human voice. If you are interested in the practical technology of interfacing with machines using voice, then this book is your guide!

216 pages • ISBN 978-1-907920-07-3 • £29.50

Elektor is more than just your favorite electronics magazine. It's your one-stop shop for Elektor Books, CDs, DVDs, Kits & Modules and much more!
www.elektor.com/shop

Elektor US
4 Park Street
Vernon, CT 06066
USA
Phone: 860-875-2199
Fax: 860-871-0411
E-mail: order@elektor.com

Kits & Modules



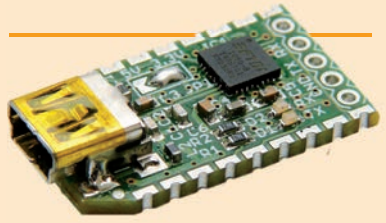
Bestseller!

Improved Radiation Meter

This device can be used with different sensors to measure gamma and alpha radiation. It is particularly suitable for long-term measurements and for examining weakly radioactive samples. The photodiode has a smaller sensitive area than a Geiger-Müller tube and so has a lower background count rate, which in turn means that the radiation from a small sample is easier to detect against the background. A further advantage of a semiconductor sensor is that is offers the possibility of measuring the energy of each particle, allowing a more detailed investigation of the characteristics of a sample. The optional PC-based software displays the energy spectrum, permitting a very detailed analysis to be carried out.

Kit of parts incl. display and programmed controller

Art.# 110538-71 • $57.30



FT232R USB/ Serial Bridge/BOB

You'll be surprised first and foremost by the size of this USB/serial converter—no larger than the moulded plug on a USB cable! And you're also bound to appreciate that fact that it's practical, quick to implement, reusable, and multi-platform —and yet for all that, not too expensive! Maybe you don't think much of the various commercially-available FT232R-based modules. Too expensive, too bulky, badly designed, ... That's why this project got designed in the form of a breakout board (BOB).

PCB, assembled and tested

Art.# 110553-91 • $20.90

Address	Name	Definition	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$0	XOUTL	10-b output value X LSB	XOUT[7]	XOUT[6]	XOUT[5]	XOUT[4]	XOUT[3]	XOUT[2]	XOUT[1]	XOUT[0]
$1	XOUTH	10-b output value X MSB	-	-	-	-	-	-	XOUT[9]	XOUT[8]
$2	YOUTL	10-b output value Y LSB	YOUT[7]	YOUT[6]	YOUT[5]	YOUT[4]	YOUT[3]	YOUT[2]	YOUT[1]	YOUT[0]
$3	YOUTH	10-b output value Y MSB	-	-	-	-	-	-	YOUT[9]	YOUT[8]
$4	ZOUTL	10-b output value Z LSB	ZOUT[7]	ZOUT[6]	ZOUT[5]	ZOUT[4]	ZOUT[3]	ZOUT[2]	ZOUT[1]	ZOUT[0]
$5	ZOUTH	10-b output value Z MSB	-	-	-	-	-	-	ZOUT[9]	ZOUT[8]
$6	XOUT8	8-b output value X	XOUT[7]	XOUT[6]	XOUT[5]	XOUT[4]	XOUT[3]	XOUT[2]	XOUT[1]	XOUT[0]
$7	YOUT8	8-b output value Y	YOUT[7]	YOUT[6]	YOUT[5]	YOUT[4]	YOUT[3]	YOUT[2]	YOUT[1]	YOUT[0]
$8	ZOUT8	8-b output value Z	ZOUT[7]	ZOUT[6]	ZOUT[5]	ZOUT[4]	ZOUT[3]	ZOUT[2]	ZOUT[1]	ZOUT[0]
$9	STATUS	Status registers	-	-	-	-	-	PERR	DOVR	DRDY
$0A	DETSRC	Detection source registers	LDX	LDY	LDZ	PDX	PDY	PDZ	INT2	INT1
$0B	TOUT	"Temperature output value" (Optional)	TMP[7]	TMP[6]	TMP[5]	TMP[4]	TMP[3]	TMP[2]	TMP[1]	TMP[0]
$0C		(Reserved)	-	-	-	-	-	-	-	-
$0D	I2CAD	I²C device address	I2CDIS	DAD[6]	DAD[5]	DAD[4]	DAD[3]	DAD[2]	DAD[1]	DAD[0]
$0E	USRINF	User information (Optional)	UI[7]	UI[6]	UI[5]	UI[4]	UI[3]	UI[2]	UI[1]	UI[0]
$0F	WHOAMI	"Who am I" value (Optional)	ID[7]	ID[6]	ID[5]	ID[4]	ID[3]	ID[2]	ID[1]	ID[0]
$10	XOFFL	Offset drift X value (LSB)	XOFF[7]	XOFF[6]	XOFF[5]	XOFF[4]	XOFF[3]	XOFF[2]	XOFF[1]	XOFF[0]
$11	XOFFH	Offset drift X value (MSB)	-	-	-	-	-	XOFF[10]	XOFF[9]	XOFF[8]
$12	YOFFL	Offset drift Y value (LSB)	YOFF[7]	YOFF[6]	YOFF[5]	YOFF[4]	YOFF[3]	YOFF[2]	YOFF[1]	YOFF[0]
$13	YOFFH	Offset drift Y value (MSB)	-	-	-	-	-	YOFF[10]	YOFF[9]	YOFF[8]
$14	ZOFFL	Offset drift Z value (LSB)	ZOFF[7]	ZOFF[6]	ZOFF[5]	ZOFF[4]	ZOFF[3]	ZOFF[2]	ZOFF[1]	ZOFF[0]
$15	ZOFFH	Offset drift Z value (MSB)	ZOFF[7]	ZOFF[6]	ZOFF[5]	ZOFF[4]	ZOFF[3]	ZOFF[2]	ZOFF[1]	ZOFF[0]
$16	MCTL	Mode control	-	DRPD	SPI3W	STON	GLVL(1)	GLVL(0)	MOD(1)	MOD(0)
$17	INTRST	Interrupt latch reset	-	-	-	-	-	-	CLRINT2	CLRINT1
$18	CTL1	Control 1	DFBW	THOPT	ZDA	YDA	XDA	INTRG(1)	INTRG(0)	INTPIN
$19	CTL2	Control 2	-	-	-	-	-	DRVO	PDPL	LDPL
$1A	LDTH	Level detection threshold limit value	LDTH[7]	LDTH[6]	LDTH[5]	LDTH[4]	LDTH[3]	LDTH[2]	LDTH[1]	LDTH[0]
$1B	PDTH	Pulse detection threshold limit value	PDTH[7]	PDTH[6]	PDTH[5]	PDTH[4]	PDTH[3]	PDTH[2]	PDTH[1]	PDTH[0]
$1C	PW	Pulse duration value	PD[7]	PD[6]	PD[5]	PD[4]	PD[3]	PD[2]	PD[1]	PD[0]
$1D	LT	Latency time value	LT[7]	LT[6]	LT[5]	LT[4]	LT[3]	LT[2]	LT[1]	LT[0]
$1E	TW	Time window for second pulse value	TW[7]	TW[6]	TW[5]	TW[4]	TW[3]	TW[2]	TW[1]	TW[0]
$1F		(Reserved)	-	-	-	-	-	-	-	-

Table 3—The MMA7455 accelerometer is a registered I²C (or SPI) device. The X, Y, and Z axes can be read as 8- or 10-bit values via sequential read operations.

by moving both simultaneously. The act of simultaneously moving multiple servos at the same time is much trickier than sequential moves.

There are a few tricks you will need to play to get this to work correctly. Suppose you want to move both feet simultaneously. The left ankle wants to move from 1,500 (flat) to 1,333 (approximately 30°) and the right ankle wants to move from 1,500 flat to 500 (90°). Let's consider the neutral position (1,500) to be zero. This means the left ankle will move –333 and the right will move –1,000. The largest absolute movement is 1,000, so the loop will be x = 0 to 1,000 step 1. Each time through the loop, the left ankle with move to new position 1,500 – (x × 1,000/1,000) and the right ankle will move to new position 1,500 – (x × 333/1,000). I wrote these both with no simplification, so the relationship would be easier to see. The problem is, we have integer-only math and the fraction 333/1,000 is always = 0, so this servo won't move!

You can work around this issue with the use of scaling. The largest (possible) move is 2,000. Therefore, if all fractional computations are scaled (multiplied/divided) by 2,000, there will never be fractional errors. Let's just look at the first two steps of the right ankle to see how this will work.

When x = 0, the right ankle position is:

$$\frac{\left[(1{,}500 \times 2{,}000) - \left(\frac{0 \times 2{,}000 \times 333}{1{,}000}\right)\right]}{2{,}000}$$

$$= \frac{(3{,}000{,}000 - 0)}{2{,}000}$$

$$= \frac{3{,}000{,}000}{2{,}000}$$

$$= 1{,}500$$

When x = 1, the right ankle position is:

$$\frac{\left[(1{,}500 \times 2{,}000) - \frac{(1 \times 2{,}000 \times 333)}{1{,}000}\right]}{2{,}000}$$
$$= \frac{(3{,}000{,}000 - 666)}{2{,}000}$$
$$= \frac{2{,}999{,}334}{2{,}000}$$
$$= 1{,}499$$

This (temporary) scaling enables rounding to occur in the calculations overcoming the limitations that the truncation has on the integer-only result. Note this is possible because variables can be declared as byte, word, or double word (signed). The double-word variable gives plenty of overhead to fit the largest computational value. Note that being "signed" also has the advantage of enabling the value to hold directional information. The result of adding this offset to the "neutral" value 1,500 is that the offset will add to or subtract from neutral, enabling movement to either side.

TURNING

If you paid attention to the photo sequence in Photo 1b, you will note that each step is completed with the feet parallel to one another. It is the sequence of movements that enables this to happen. If the sequence is altered slightly (i.e., only one hip is rotated), then the foot won't come down parallel to the opposite foot. If on the next step, when the opposite leg is up, you merely rotate that leg, the feet will become parallel once

Name	Type	Register address	Auto-increment address (Fast read)	Default value	Comment
DR_STATUS	R	0x00	0x01	0000 0000	Bit7 - ZYXOW - overwrite
					Bit6 - ZOW - overwrite
					Bit5 - YOW - overwrite
					Bit4 - XOW - overwrite
					Bit3 - ZYXOR - data ready
					Bit2 - ZOR - data ready
					Bit1 - YOR - data ready
					Bit0 - XOR - data ready
OUT_X_MSB	R	0x01	0x02 (0x03)	data	Bits [15:8] of X measurement
OUT_X_LSB	R	0x02	0x03	data	Bits [7:0] of X measurement
OUT_Y_MSB	R	0x03	0x04 (0x05)	data	Bits [15:8] of Y measurement
OUT_Y_LSB	R	0x04	0x05	data	Bits [7:0] of Y measurement
OUT_Z_MSB	R	0x05	0x06 (0x07)	data	Bits [15:8] of Z measurement
OUT_Z_LSB	R	0x06	0x07	data	Bits [7:0] of Z measurement
WHO_AM_I	R	0x07	0x08	0xC4	Device ID Number
SYSMOD	R	0x08	0x09	data	Current System Mode
OFF_X_MSB	R/W	0x09	0x0A	0000 0000	Bits [14:7] of user X offset
OFF_X_LSB	R/W	0x0A	0x0B	0000 0000	Bits [6:0] of user X offset
OFF_Y_MSB	R/W	0x0B	0x0C	0000 0000	Bits [14:7] of user Y offset
OFF_Y_LSB	R/W	0x0C	0x0D	0000 0000	Bits [6:0] of user Y offset
OFF_Z_MSB	R/W	0x0D	0x0E	0000 0000	Bits [14:7] of user Z offset
OFF_Z_LSB	R/W	0x0E	0x0F	0000 0000	Bits [6:0] of user Z offset
DIE_TEMP	R/W	0x0F	0x10	data	Temperature, signed 8 bits in °C
CTRL_REG1	R/W	0x10	0x11	0000 0000	Bit7 - DR2 - data rate 2
					Bit6 - DR1 - data rate 1
					Bit5 - DR0 - data rate 0
					Bit4 - OS1 - over sampling ratio 1
					Bit3 - OS0 - over sampling ratio 0
					Bit2 - FR - fast read
					Bit1 - TM - triggered measurement
					Bit0 - AC - operating mode
CTRL_REG2	R/W	0X11	0x12	0000 0000	Bit7 - AUTO_MRST_EN - auto reset
					Bit6 -
					Bit5 - RAW - no offset (0x09-0x0E)
					Bit4 - MAG_RST - reset cycle
					Bit3 -
					Bit2 -
					Bit1 -
					Bit0 -

Table 4—The MAG3110 magnetometer is a registered I²C (or SPI) device. The X, Y, and Z axes values are signed data, which can be read as byte or word values via sequential read operations according to the FR bit in CTRL_REG1.

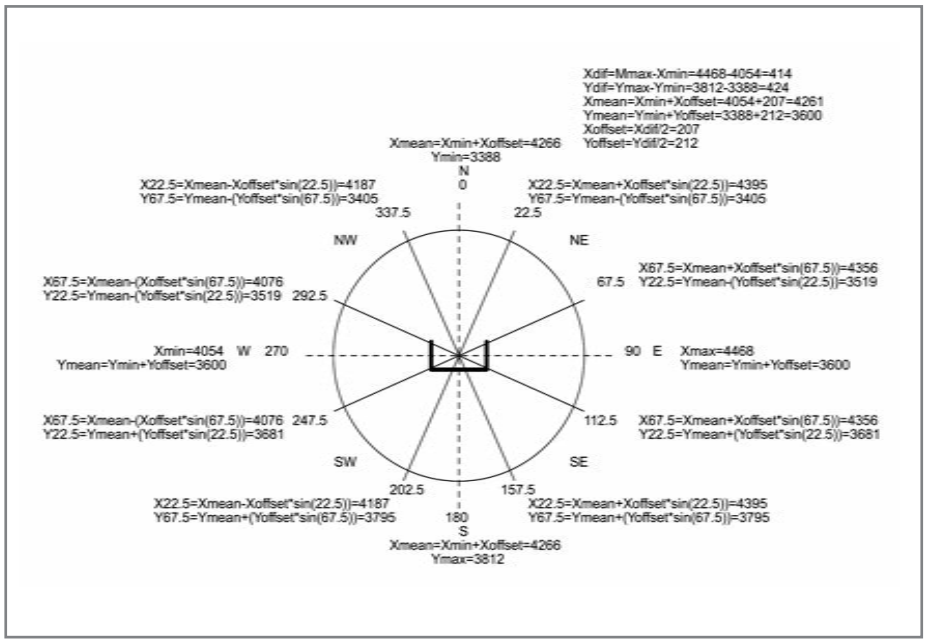


Figure 2—This diagram shows the FSLBOT in the center facing south. The dotted lines show the X-axis east/west and Y-axis south/north. The 360° circle is divided into eight boundary areas, each 45° wide. These mark the boundaries for the eight primary compass directions. Located around the outer circle are the X and Y measurements associated with the FSLBOT facing that direction. The maximum and minimums are collected while the FSLBOT is rotated 360°. The remaining values are calculated.

measurements. As I tilted the FSLBOT left, the X-axis went negative in value. As I tilted to the right, X increased. When tilted forward, the Y-axis decreased. It increased when tilted backward. The Z-axis changed little. It was clear that the values could be used to indicate an amount of tilt. I took reference samples of the robot while it was standing flat on both feet and at the balance point standing on each leg. The balance point was a difference of seven counts from the reference. I monitored this value in a loop to adjust the ankle positioning for proper balance. I know this may not be the best way to detect the balance point (I should be using a change in acceleration to determine this), but this will suffice for the time being. Future experiments may also include monitoring acceleration while swinging the hip servos to prevent tipping from moving the body mass too quickly.

again, and when the leg is lowered, the robot will have turned slightly (an offset of ±333 equals approximately 30° increments) about the center of one of the feet. I developed a collection of routines that will step/turn the FSLBOT starting with either foot and move/turn forward or backward. This should cover most required movements; however I'm sure there is room for improvements.

BALANCE

One of the sensor modules offered for the FSLBOT is a three-axis accelerometer using Freescale's MMA7455L digital output accelerometer. This has three low "g" ranges ±2 g/4 g/8 g and the value of each axis can be read as either 8 or 10 bits. Since gravity will look like a 1-g acceleration toward the center of the earth, an accelerometer can be used to determine tilt (whenever there is no other acceleration). Acceleration values can be measured at a 125-Hz (or 250-Hz) rate with conversion ready signaled via an interrupt pin or through the status register. Four modes are available to the user: power down, measurement, level detection, and pulse detection. While the first two modes are obvious, level detection can be used to trigger an interrupt when any axis exceeds predefined thresholds or when all axes are less than predetermined thresholds (as in freefall). Pulse detection can be used to interrupt-on-a-tap with programmable durations in the millisecond range. Double taps can also be detected with programmable latency between taps. Table 3 shows the registers that make up the MMA7455L.

I used acceleration due to gravity as an indicator of balance. I ran a simple program that output the X, Y, and Z

DIRECTION

During my evaluation, a second sensor module became available for use in one of the two sensor sockets on the FSLBOT, a three-axis magnetometer using Freescale's MAG3110 magnetic sensor. This device is capable of measuring magnetic fields with an output data rate (ODR) up to 80 Hz; these output data rates correspond to sample intervals from 12 ms to several seconds. Most people know that the balancing magnetized needle of a compass will be drawn toward the magnetic north and south poles by the earth's magnetic lines of flux. The MAG3110 measures the directional strength of this flux that penetrates the device's X, Y, and Z sensors (see Table 4). I²C communication is used to access its 18 registers (refer to Table 3), which hold status, measurement data, offset correction data, and configuration values.

Metal and ferrous materials can interfere with the natural pattern of the flux and distort these measurements. When necessary, a calibration routine can be used to identify this distortion and offset correction data can be applied to every reading to cancel out the effects of local fixed distortions. Note that there are separate data-ready (DR) bits associated with each axis as well as a global bit (ZYXDR) that is mirrored by the interrupt pin. The MAG3110 can be set for continuous or triggered measurements. The maximum ODR is based on the ADC rate (set by the DR2:0 bits) divided by the oversampling rate (set by the OS1:0 bits).

As the MAG3110 is positioned on my FSLBOT, the X-axis has a maximum output when the robot is facing south and the Y-axis has maximum output when the robot is facing east. The outputs are dynamic and quite sensitive. The

magnetometer's output will vary sinusoidally if rotated 360°. During this time, each axis will output some maximum amount (when it is aligned with its referenced direction) and some minimum amount (when it is aligned in the opposite direction). To determine a compass heading, you must gather the maximum and minimum values for each axis. When properly calibrated by adding offsets, each axis would output a positive value (maximum), a negative value (minimum), and a null or zero value when 90° to the flux. To avoid having to deal with signed values, you can add an offset that will never enable the value to be negative. Referring to Figure 2, you can see how typical values I've measured for each the X- and Y-axis relate to one another. After rotating the sensor to gather this data, I calculate a number of constants that will be used in determining direction. Starting with the maximum and minimum values:

$X_{MAX} = 4{,}468$, $X_{MIN} = 4{,}054$, $Y_{MAX} = 3{,}812$, $Y_{MIN} = 3{,}388$

We can calculate:

$$X_{DIF} = X_{MAX} - X_{MIN} = 414$$
$$X_{OFFSET} = \frac{X_{DIF}}{2} = 206$$
$$X_{MEAN} = X_{MIN} + X_{OFFSET} = 4{,}260$$
$$Y_{DIF} = Y_{MAX} - Y_{MIN} = 424$$
$$Y_{OFFSET} = \frac{Y_{DIF}}{2} = 212$$
$$Y_{MEAN} = Y_{MIN} + Y_{OFFSET} = 3{,}600$$

Boundary points can be calculated at 22.5° and 67.5° points in the sine curves, where 22.5° = 0.38 and 67.5° = 0.92. Referring to Figure 2, the FSLBOT is facing south. The circle is labeled in degrees with north being 0°. The compass is divided into eight pie-shaped boundaries that are offset 22.5° from north. The X and Y values are calculated for these boundaries. The eight principal directions fall within these boundaries. You can see that if the X sensor's value is less than 4,356 and greater than 4,395, then we may be either NE or SE. If the Y sensor's value is less than 3,519 and greater than 3,405, then we must be either NE or NW. Using both values, we can conclude that the true direction must be NE.

Once all this is proven using a static compass program—that is, manually rotating the FSLBOT and then printing out the present direction the FSLBOT is turned toward—we can begin to automate these functions. This basically means having the robot turn itself around 360° to capture the maximum and minimum values for each axis. At first, I made 12 30° movements pausing between each movement to record compass values. I found that if the feet slip too much, it won't get a full 360°. So, I used a slightly different method. I kept turning until I saw axis values change direction three times, ensuring that it has moved at least 360°.

The second part was to give the robot an objective and that is to face north. Once the calibration is done, the FSLBOT can determine its present direction and move toward the north. I turn on the smile LEDs when I check for north and turn them off if another turn is necessary. When the smile remains illuminated, the robot has achieved its objective. If you move it (rotate it), it will begin to move itself back toward north.

DONE PLAYING

At some point you will have become comfortable with using a particular device. You can choose to move on to another device or begin to work with that device in a language other than BASIC. The tools provided for use with the FSLBOT are there to gently encourage you further. For whatever reason, I never considered using Freescale's ColdFire 32-bit processor. Now I think I would like to experiment with Freescale's MQX real-time operating system (RTOS) that is available for use with the FSLBOT's TWR-MECH board. There are so many directions you can go with this great offering from Freescale. The guys who have put together this package have devoted many hours of personal time to see their new vision succeed. "Provide a complete set of tools that work in harmony with a wide portfolio of 32-bit MCU solutions. Interface that hardware with real-world peripherals in a way that invites entertainment with a side of subconscious education. Make it simple to program and it will become addicting."

During the freak snow storm that hit the northeast at the end of October 2011 (that's another story), I had a couple of grandkids overnight while there was no school. When they saw the FSLBOT they immediately wanted to play with it. While I was putting on little show to make the robot "dance" (walk), my grandkids sat down on the floor mesmerized. Then eight-year-old Josh jumped up and began pressing keys on my laptop as if to make Pinocchio come alive. "Sorry, Geppetto," I said to him, "maybe next time."

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

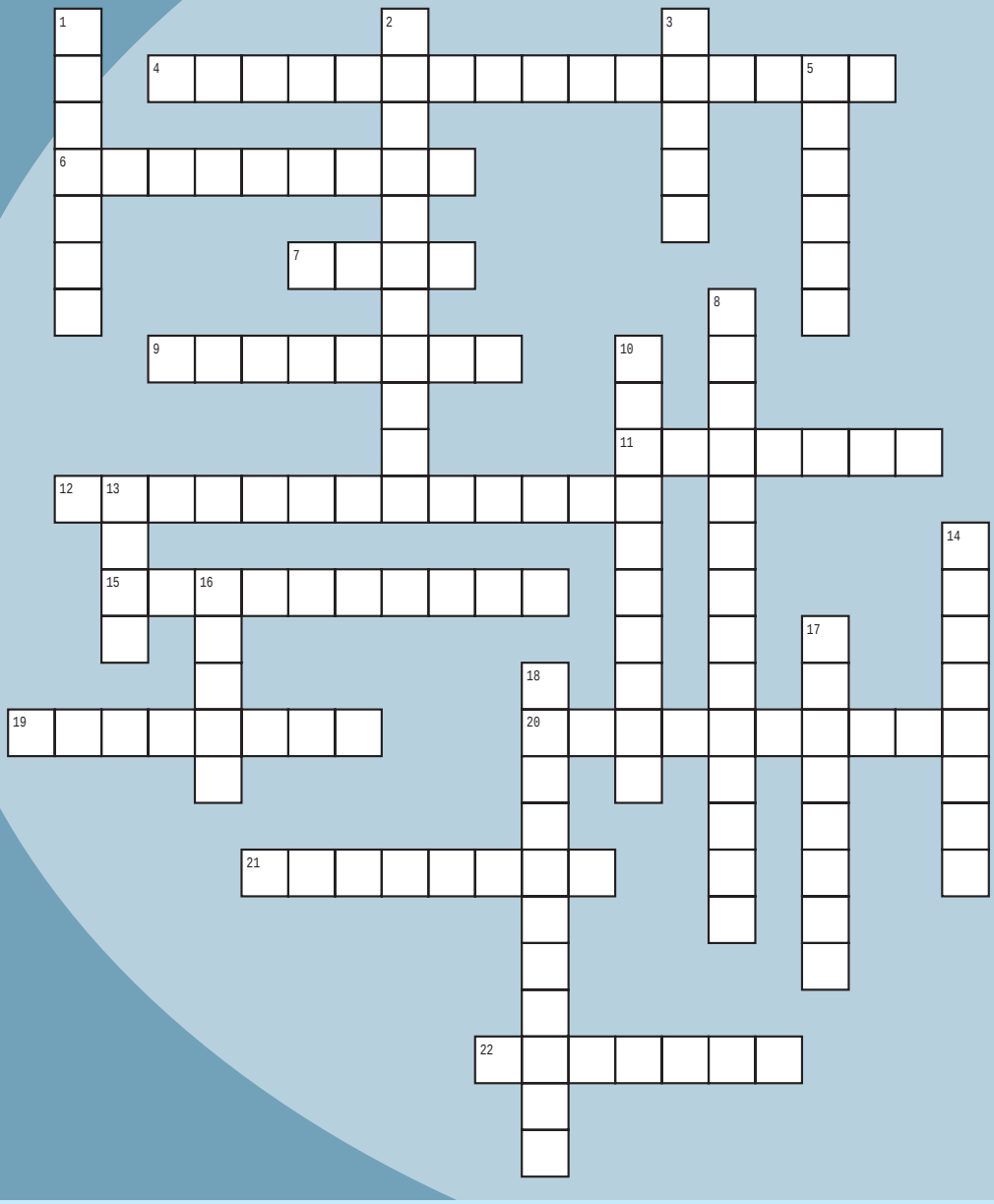
To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/260.

SOURCES

StickOS BASIC Interactive programming environment
CPUStick.com | www.cpustick.com

FSLBOT: Freescale Mechatronics Robot, Tower System, TWR-MECH board, MCF52259 ColdFire 32-bit processor, MC9S08JM60 microcontroller, MMA7455L digital output accelerometer, MAG3110 magnetic sensor, and MQX RTOS
Freescale Semiconductor, Inc. | www.freescale.com

CROSSWORD



Across

4. Applied science of ergonomics [two words]
6. Output not proportional to input
7. It isn't aligned above the center of an oblique cone's base
9. A switch, control, or lever used for guidance
11. A robot with human-like qualities, or the opposite of iOS
12. Value estimation
15. An LED emitting coherent light [two words]
19. Switching from high to low or low to high
20. A beam with only one fixed end
21. Filter type associated with GRSK
22. Natural substance capable of producing voltage

Down

1. Operating below average
2. Base of 16
3. True, but not a theorem; helps prove a theorem
5. Used in math to indicate a quantity or amount
8. Enables only a narrow band of frequencies access to a circuit [two words]
10. Able to function without connection
13. Empty set
14. What happens when 0 is 1 and 1 is 0? [two words]
16. Screen lock
17. 1.602×10^{-19}
18. A 20-faced polyhedron

The answers will be available in the next issue.

Elektor Print

Classical reading: on paper



+

Elektor Digital

Contemporary & innovative reading: on PC, notebook or tablet

=

Elektor PLUS

Comprehensive reading: at home and on the road

Read Elektor with the premium PLUS subscription!

Subscribe now or upgrade: www.elektor.com/usa

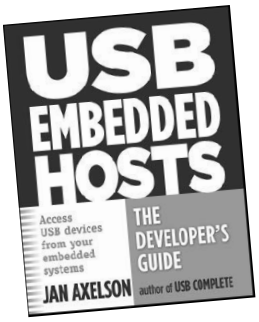
IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

AD FORMAT: Advertisers must furnish digital files that meet our specifications (www.circuitcellar.com/advertise). ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2" x 3" FORMAT. E-mail adcopy@circuitcellar.com with your file or send it to IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066. For current rates, deadlines, and more information contact Peter Wostrel at 800.454.3741, 978.281.7708 or peter@smmarketing.us.

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

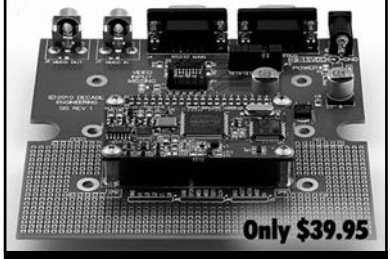
Access USB Devices
from your
Embedded Systems



USB Embedded Hosts
The Developer's Guide
Jan Axelson
$29.95 **LVR.COM**
From the author of *USB Complete*

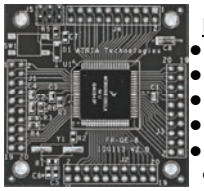
NEW **SIS Board**

Turbocharges the development of low-cost modular video text and graphics generators using **BOB-4**!



www.decadenet.com

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



Microcontrollers
- MCF51AC256
- MCF51CN128
- MCF51JM128
- MCF51QE128
- MC9S08AC128
- MC9S08JM64
- MC9S08QE128

Modules
- Bluetooth
- RS-232
- USB
- RF
- 2 x 20 LCD
- 4 x 4 Keypad
- Motor Control
- Real Time Clock
- Power Supply

GREAT For:
- Projects
- Breadboards
- Experiments
- Labs
- Prototypes

Kits Available

Learn about BASIC ON BOARD

ATRIA *Technologies Inc.*
www.AtriaTechnologies.com

i2cStick™



Discover I²C
Read, Write, Connect...

MCC
Micro Computer Control

www.mcc-us.com

microEngineering Labs, Inc.
www.melabs.com 888-316-1753

Programmers for Microchip PIC® Microcontrollers



Starting at $79.95

PC-Tethered USB Model (shown):
- •Standalone software
- •Command-line operation
- •Hide GUI for automated use
- •Override configuration with drop-downs

Stand-Alone Field Programmer:
- •Power from target device or adapter
- •Program file stored on SD-CARD
- •Programming options stored in file
- •Single-button operation

Program in-circuit or use adapters for unmounted chips.
Zero-Insertion-Force Adapters available for DIP, SOIC, SSOP, TQFP, and more.

PIC is a registered trademark of Microchip Technology Inc. in the USA and other countries.

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

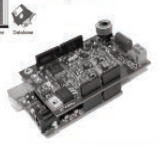
Free 96 page catalog

1-800-826-5432

Power-Line Communication, Street Lighting System, UHF EPC Gen2 RFID Reader, Arduino Shields (WiFi)



■Power-Line Communication Street Lighting System



■Power-Line Communication Modules

- Low-Speed Modem Pro
- Low-Speed Shield for Arduino – The first one in the world!!!
- High-Speed Modem
- High-Speed Modem for DMX512



■Arduino Shields: WiFi, GPRS, Charger, IR, Smart Outlet

■JPEG Serial Camera

■Cellular modules

■GPS, Bluetooth modules

■ LS6410 embedded system with Win CE 6.0, Linux, and Android.

www.linksprite.com

MaxSonar

Perfect for OEMs & Engineers



XL-MaxSonar-EZ

- High acoustic power
- Real-time calibration
- Tiny size
- $39.95 USD qty. 1

MaxSonar-WR (IP67)

- Industrial packaging
- Weather resistant
- Standard ¾" fitting
- $99.95 USD qty. 1





LV-MaxSonar-EZ

- Power-up calibration
- Smallest MaxSonar
- Low power, 2.5V-5.5V
- $29.95 USD qty. 1

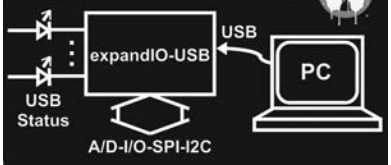
MaxSonar-WRC (IP67)

- Compact packaging
- Weather resistant
- Quality narrow beam
- $99.95 USD qty. 1



www.maxbotix.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

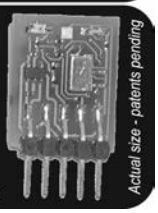
www.hexwax.com - Buy from Mouser & Farnell

Amazing PIC programmer

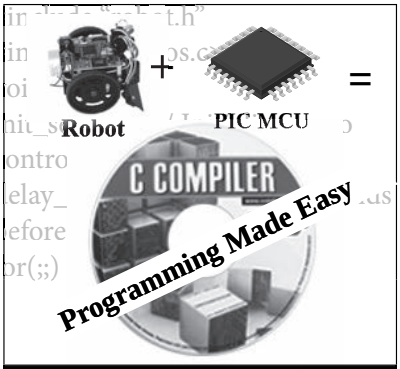
Most devices supported ICSP, SQTP, & copy limits

$32 at Digikey & Mouser

www.flexipanel.com



Actual size - patents pending



Compiler Includes:

- 160 Device/Peripheral Drivers
- 144 Example Programs
- PIC 16 Compilers start at $150

CCS Inc

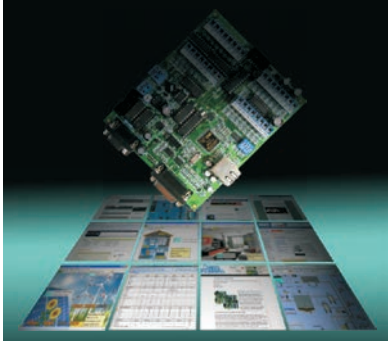
www.ccsinfo.com/ccad

sales@ccsinfo.com

262-533-6500 ext. 35

INTEGRATED

PLCs with Integrated Ethernet



Integrated Features :

- ETHERNET / Modbus TCP/IP
- 16 or 32 digital I/Os
- 10 analog I/Os
- RS232 and RS485
- LCD Display Port
- I/O Expansion Port
- Ladder + BASIC Programming

$229 and $295

before OEM Qty Discount

tel : 1 877 TRI-PLCS

web : www.tri-plc.com/cci.htm

TRI TRIANGLE RESEARCH INTERNATIONAL

JazzOS - Java for Makers

Get started with the JazzOS Explorer

Program in Java

32 Bit Native Code Performance

28 Multipurpose I/O Channels

8 Megabytes RAM
16 Megabytes FLASH

Plug-and-Play Development

$89.00*

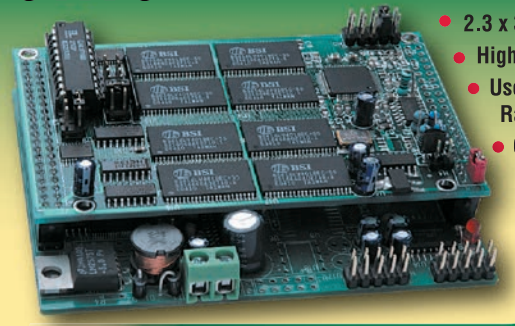
* Quantity 1 ~ Volume discounts available.



[PERNETICS] Practical Innovation

PERNETICS.COM

(800) 651-8698

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

RF Specialists
RF Modules
GSM/GPRS
Bluetooth
Data Loggers
GPS
Wi-Fi
ZigBee Pro
LEMOS INTERNATIONAL
www.lemosint.com

BGA & QFN Sockets
Quick-Turn Custom Designs
• Bandwidths to 40 GHz
• Industry's Smallest Footprint
• Five different contactor options
• Ideal for Prototype and Test
• Simulation Models Available
• Multi Level Stacked Sockets
• BGA and QFN
• Sockets for ALL Xilinx and Altera Chips
• Pitch 0.4mm to 1.27mm
• SMT Options
RoHS
Ironwood ELECTRONICS 1-800-404-0204
www.ironwoodelectronics.com

BPS BusBoard Prototype Systems
Prototyping PC Boards
Many Patterns
BusBoard StripBoard PadBoard ProtoBoard ProtoBoard-2H PowerBoard SMTBoard SMTpads PC BreadBoards
See our site
www.BusBoard.us
Available From
JAMECO ELECTRONICS
MOUSER ELECTRONICS a ttl company
amazon.com amazon.co.uk

Disk On Chip
Ready for Delivery
From 16MB to 128MB Available!
Call 530-297-6073 Sales@jkmicro.com
www.jkmicro.com
JK microsystems, Inc.
International Orders Welcome

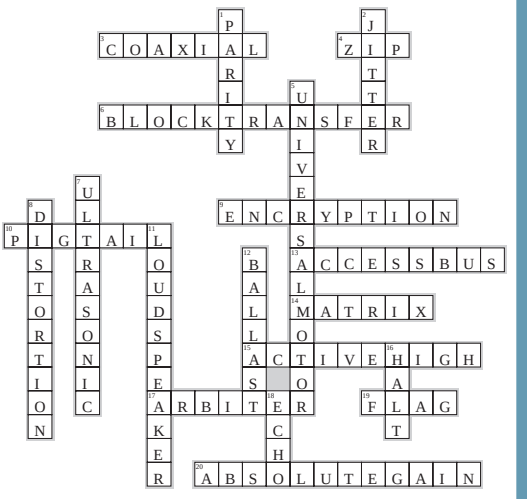
CROSSWORD ANSWERS from Issue 258

Across

3. **COAXIAL**—Having the same turning point
4. **ZIP**—Used to compress and archive data
6. **BLOCKTRANSFER**—Moves groups of data [two words]
9. **ENCRYPTION**—Unreadable without a key
10. **PIGTAIL**—A short length of wire used to connect components
13. **ACCESSBUS**—Uses an I²C bus as its electrical hardware interface [two words]
14. **MATRIX**—A group of numbers or symbols found in a rectangular
15. **ACTIVEHIGH**—A type of signal that represents a binary digit of 1 [two words]
17. **ARBITER**—Has the power to accept requests from and grant control of buses
19. **FLAG**—Can be a character, digit, signal, or symbol
20. **ABSOLUTEGAIN**—The relationship between the signal level at the output of a device to its input under specific conditions [two words]

Down

1. **PARITY**—A method of checking accuracy
2. **JITTER**—Fluctuate, vibrate
5. **UNIVERSALMOTOR**—Operates from single-phase AC or DC power [two words]
7. **ULTRASONIC**—Audible at more than 20,000 Hz
8. **DISTORTION**—An audio signal that isn't quite right
11. **LOUDSPEAKER**—Radiates acoustic energy
12. **BALLAST**—Used to balance current flow
16. **HALT**—Stop command
18. **ECHO**—Sound wave bounce



CLIENT INDEX

Page		Page		Page		Page	
77	All Electronics Corp.	51	DESIGN West Summit	31	Jeffrey Kerr, LLC	35	Pololu Corp.
23	AP Circuits	15	egnite GmbH	15, 78	JK microsystems, Inc.	1	Renesas RL78 Green Energy Challenge
47	ARM Ltd.	68, 69	Elektor	41	Labcenter Electronics	23	Robot Power
76	Atria Technologies, Inc.	75	Elektor	76	Lakeview Research	49	Saelig Co., Inc.
39	Bitscope Designs	57	EMAC, Inc.	19, 78	Lemos International Co.	19	SiliconRay
78	Busboard Prototype Systems	33	Express PCB	77	Linksprite	2, 3	Technologic Systems
63	CC Gold	77	FlexiPanel Ltd.	77	Maxbotix, Inc.	78	Tern, Inc.
5	Comfile Technology, Inc.	57	Grid Connect, Inc.	76	MCC, Micro Computer Control	77	Triangle
77	Custom Computer Services, Inc.	31	HannoWare	76	microEngineering Labs, Inc.	45	Trinity College Robotics Contest
76	Decade Engineering	34	Humandata Ltd.	7	Mouser Electronics, Inc.		
17	DesignSpark chipKIT™ Challenge	C4	Imagineering, Inc.	C2	NetBurner		
26, 27	DesignSpark chipKIT™ Challenge	78	Ironwood Electronics	77	Pernetics		

PREVIEW of April Issue 261

Theme: Embedded Programming

The menU Universal Menu System

The Rainbow Color Reader (Part 2): Color Distinction & Classification

Build a Capacitive-Touch Gain-Controlled Personal Amplifier

THE CONSUMMATE ENGINEER **Is Software Development an Art or a Science?**

FROM THE BENCH **SNAP to It (Part 1):** Mesh Networking Simplified

EMBEDDED IN THIN SLICES **Getting Started with Embedded Linux (Part 4):** Linux Software Development Tools

THE DARKER SIDE **Delta-Sigma Modulators 101**

ABOVE THE GROUND PLANE **RFI Bypassing (Part 2):** Parallel Capacitors

ATTENTION CLIENTS

May Issue 262 Deadlines

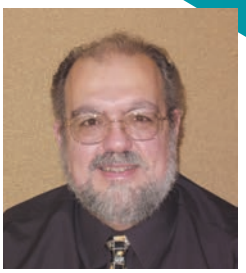
Space Close: March 12
Materials Close: March 19

Theme
Measurement & Sensors

Bonus Distribution
CTIA Wireless

Call Peter Wostrel
now to reserve your space!
800.454.3741 or 978.281.7708
e-mail: peter@smmarketing.us

PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

A Rose by Any Other Name...

Let's talk about languages—specifically, embedded control languages. Everyone has their favorite, typically whatever they learned first, but when you get right down to it, all languages offer the same basic features.

First of all, you need to be able to specify a sequence of steps and then select between two (or more) alternative sequences—the if-then-else construct. You also need to be able to repeat a sequence, or loop, and exit that loop when a condition is met. Finally, you want to be able to invoke a sequence from multiple places within other sequences—a call function.

Assembly language is the lowest-level language you can use on most machines. Its statements bear a one-to-one relationship with the instructions the hardware executes. If-then-else and loop-exit constructs are implemented using conditional and unconditional branch instructions, and there's usually a hardware stack that facilitates subroutine call and return. This is both a blessing and a curse—it enables you to become familiar with the hardware and use it most effectively, but it also forces you to deal with the hardware at all times.

Very early on in the development of computers, the concept of a high-level language (HLL) was developed to reduce this hardware focus. By creating a set of abstract computer operations that aren't necessarily tied to a particular processor, the HLL frees the programmer from a specific hardware architecture and enables him to focus on the actual algorithm development. The compiler and library writers took these abstractions and efficiently mapped them onto the hardware. HLL opened up programming to "non-hardware" people whose first interest was the application problem and its solution.

Today, there are literally hundreds of computer languages (see http://en.wikipedia.org/wiki/List_of_programming_languages). Some of them are completely general-purpose, while others are very domain-specific. Two languages have been implemented on virtually every microprocessor ever invented: C and BASIC. (There's no way I can mention them all, so I'll just touch on some popular embedded ones.) Of the two, C is by far the more popular for embedded apps, since it runs more efficiently on most hardware. Many people would argue that C isn't a "true" HLL; but even still, it's a huge step up from Assembly language in terms of productivity.

There have been some niche languages intended for small systems. For example, there's what you might call a family of reverse-Polish notation (RPN) languages: Forth, Postscript, and does anyone remember a tiny interpreted language called Mouse? These never caught on in any big way, except for Postscript, which is almost universally available these days on printers as a page-description language. But it's a full programming language in its own right—just ask Don Lancaster!

Along the way, there have been a few processor-specific languages invented. For example, there's JAL—just another language—which is optimized for 8-bit Microchip PIC processors, and Spin, which is designed to support the parallel-processing features of the Parallax Propeller chip.

Once you start getting into larger 16- and 32-bit chips, the set of available tools expands. Many of these processors have C/C++ toolchains based on the GNU Compiler Collection (GCC). However, this means you can really use any number of languages in the collection on these processors, including Fortran, Java, and Ada.

The designers of some embedded systems want to include the ability for the system to be programmed by their end users. To this end, the concept of an "extension language" was developed. Two notable examples are TCL and Lua. These provide a standard notation for control constructs (branching, looping and function calls) wrapped around application-specific primitive operations implemented by the system designer.

Once you start getting into systems that are large enough to require an operating system (real-time or otherwise), many of the available choices support the POSIX API. This means you can use any of the mainstream scripting languages—such as shell scripts, Perl, Python, Ruby, PHP, etc.—either internally or exposed to the end user.

And finally, there's the web-based user interface. Even relatively simple embedded applications can have sophisticated GUIs by pushing much of the programming out to the browser itself by means of ECMAscript (JavaScript) or Java. The embedded app just needs to implement a basic HTTP server with storage for the different resources needed by the user interface running on the browser. There are all kinds of toolkits and frameworks out there that help you build such a system.

I'll stop now. The point is, even in the world of embedded computer applications, there's a wide variety of tools available, and picking the right set of tools for the job at hand is part of the system design process. Taking a higher-level view, this brief survey might give you an idea of what kinds of tools you would want to put some effort into learning, based on where your interests lie or the application dictates.

steve.ciarcia@circuitcellar.com

CIRCUIT CELLAR

Buy it today!

ADuC841 Microcontroller Design Manual:
From Microcontroller Theory to Design Projects



$35 USD

Shlomo Engelberg provides a comprehensive guide to designing and programming with the Analog Devices, Inc. ADuC841 microcontroller as well as other microcontrollers in the 8051 family. This book provides valuable introductory and advanced labs which demonstrate the many ways you can use a microcontroller to solve practical problems.

Perfect for university classrooms and self study! If you are a professor interested in buying a bulk shipment for your students, e-mail custserv@ circuitcellar.com for more information.



www.cc-webshop.com


PC Boards, Components, Assembly
because no one should ever have to wait again...
... imagineering has Revolutionized the Online Turnkey Quoting Process
Join the Quoting Revolution Happening Now @
PCBnet.com