Q&A: Exploratory Designs & Programs
LOCATION: United States
PAGE: 18

INSIGHT: Sine Wave Synthesis
LOCATION: Canada
PAGE: 26

DESIGN: DIY Battery Control Unit
LOCATION: United States
PAGE: 44

TIPS: Schematic Prep
LOCATION: United States
PAGE: 64

CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

JULY 2011
ISSUE 252

INTERNET & CONNECTIVITY

Microcontroller-Based GSM Connection

Construct a Password Security System

Pre-CAN Protocol 101

Build a Six-Axis Mini Articulated Robot

PLUS
mbed Innovation
Rapid Prototyping Success
// Smart "Electronic Load"
// RTOS for mbed
// Network Traffic Meter
// And More

$7.50 U.S. ($8.50 Canada)
www.circuitcellar.com

NEW chipKIT™ Development Platforms from Microchip

The first 32-bit MCU platforms compatible with Arduino™ HW and SW

The chipKIT™ Uno32 and Max32 development boards are the first 32-bit microcontroller-based platforms that are compatible with existing Arduino™ hardware and software.

The chipKIT platform allows hobbyists and academics from many disciplines, such as mechanical engineering, computer science and artists to develop original embedded applications easily and quickly including: motor control, LCD display, wired and wireless communications, LED matrix control and sensor networks

Key Features:

- Application development using an environment based on the original Arduino IDE, modified to support PIC32 devices while still supporting the original Arduino line. Leverages existing code examples, tutorials and resources.
- Pin-out compatibility with many existing Arduino shields
- Higher performance at a lower price-point than existing solutions
- Advanced capabilities including:
  - Integrated USB (Device/Host, OTG)
  - Integrated Ethernet
  - CAN

Development Board Comparison

0 KI	- I	6 I IE F	6I AI E E IN 3,	3,	0CF D0 KI
BCH32 P F	0K	1		)	6 6 686 - -
BCH32 M	0K	1	(		8, M- 5 / KB IF K - -

IT'S EASY TO GET STARTED!

- Visit **www.microchip.com/chipkit**
- Purchase a chipKIT development board
- Download the free software
- Order free PIC32 samples

chipKIT Uno32 (TDGL002)

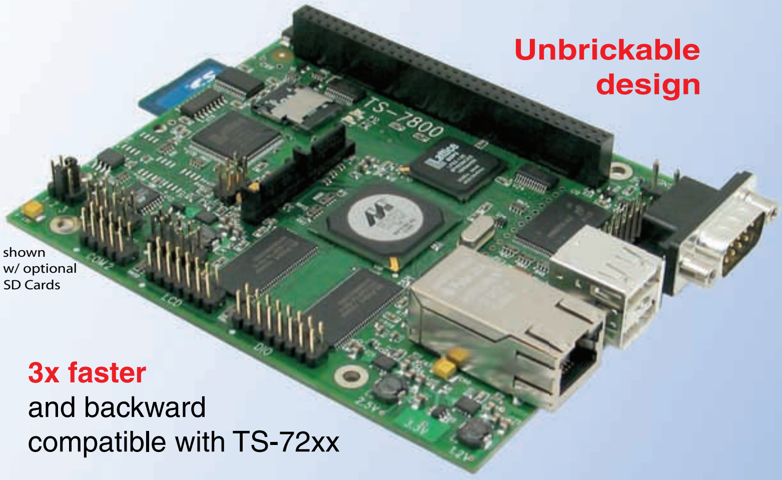
chipKIT Max32 (TDGL003)

www.microchip.com/chipkit

MICROCHIP

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
- 128MB DDR RAM
- 512MB high-speed (17MB/sec) onboard Flash
- 12K LUT customizable FPGA
- Internal PCI Bus, PC/104 connector
- 2 host USB 2.0 480 Mbps
- Gigabit ethernet
- 2 SD sockets
- 10 serial ports
- 110 GPIO
- 5 ADC (10-bit)
- 2 SATA ports
- Sleep mode uses 200 microamps
- Boots Linux 2.6 in 0.7 seconds
- Linux 2.6 and Debian by default

$229 qty 100

$269 qty 1

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

TASK MANAGER

Connection Achieved

The cover of the Internet & Connectivity issue is a great place to feature our new app, the Circuit Cellar Electronic Toolbox. The app—now available in Apple iTunes for the iPad and iPhone—is an excellent resource for designers, programmers, and electronics enthusiasts who work with embedded technologies and computer applications. It is intended to provide users with immediate, on-the-go access to need-to-know information and a variety of handy tools: a value calculator for resistors, capacitors, and filter circuits; an Ohm's Law calculator; unit conversions (pressure, energy, distance, speed, weight, etc.); schematic diagrams; and much more.

Note that *Circuit Cellar*'s app is based on the Elektor Electronic Toolbox, so you should choose the app that best suits your needs. The main difference between the two apps is that *Circuit Cellar*'s includes direct links to *Circuit Cellar* webpages—which connect you with us—as well as a technical writing tool that enables you to better communicate engineering information to your customers and coworkers. Use the Tech Writing tool when you need to write a technical document such as a datasheet, academic paper, or white paper.

We trust you'll find the app handy. Be sure to tell us what you think!

As for this issue, well, it's jam-packed with a lot of content. A good place to start is George Novacek's article. On page 26 he explains how to modify a circuit to build a robust sine wave synthesizer featuring a WWVB receiver module, an antenna, and a resonator.

Next, turn to page 30 for an article about Toby Baumgartner's innovative TROBOT six-axis robot design. The third version of the robot is built around a Texas Instruments LM3S9B96.

Interested in embedded intelligence (EI)? Check out Indranil Majumdar's MCU-based GSM cell phone design that features emergency auto-dial (p. 38).

Power-hungry users should try replicating Marty McLeod's li-ion battery control unit (BCU). On page 44 he explains how he built the BCU and implemented a safety-certified RTOS.

Long-time *Circuit Cellar* readers know that Cornell University's ECE Department has produced many amazing projects over the years. This month we feature another inventive design—a touchpad password security system (p. 54).

On page 64 George Martin presents the second installment in his series on product design. This time he covers schematic preparation.

Once you're up to speed on design development, check out Jeff Bachiochi's article on pre-CAN protocols (p. 68). It's the second article in his series about the technologies associated with vehicle diagnostics.

cj@circuitcellar.com

C. Abate

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
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Ken Davidson
David Tweed

STAFF ENGINEER
John Gorsky

PUBLISHER
Hugo Van haecke

ASSOCIATE PUBLISHER
Shannon Barraclough

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ADMINISTRATIVE COORDINATOR
Valerie Luster

ART DIRECTOR
KC Prescott

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

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

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

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

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Vernon, CT 06066. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $45, Canada/Mexico $60, all other countries $63. Two-year (24 issues) subscription rate USA and possessions $80, Canada/Mexico $110, all other countries $116.** All subscription orders payable in U.S. funds only via Visa, MasterCard, international postal money order, or check drawn on U.S. bank. **Direct subscription orders and subscription-related questions to Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650 or call 800.269.6301.**

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

INSIDE ISSUE 252

July 2011 • Internet & Connectivity

NXP mbed Design Challenge Results

p. 22

First Prize
(VI)sualizer
A Smart Electronic Load
Hexley Ball

Second Prize
mbos
An RTOS for mbed
Andrew Levido

Third Prize
CNC Panel Cutter
James Koehler

Build a Six-Axis Articulated Robot

p. 30

MCU-Based GSM Phone

p. 38

Battery Control Unit

p. 44

by Simon Ford

NXP mbed DESIGN CHALLENGE

mbed Challenge Success!

The mbed team set out to make rapid prototyping with microcontrollers a reality. Mission accomplished. Now it's up to the world's most innovative designers to take what they've learned and continue designing.

Well done Hexley Ball, the proud winner of the first ever mbed competition!

As you have probably seen by now, the winning entry was an intelligent electronic load—an unassuming name for an engineering feat. The finished product is an eclectic mix of digital and analog electronics, mechanical and web design, all packaged to provide a totally professional and innovative product (see Photo 1). Not to mention the comprehensive write-up submitted with it, which is very important to *Circuit Cellar* as a publisher. It is a real example of how a design challenge entry should be done.

I was lucky enough to meet Hexley at the Embedded Systems Conference 2011 in Silicon Valley, where the results were announced and NXP Semiconductors had invited him to show off his design. It was wonderful to have the creator there in person to take me through all the features of what really is an impressive bit of kit. It uses just about every interface on the mbed to provide amazing levels of functionality. With every feature, my appreciation of how much engineering had gone into the design grew, as did my surprise to how it could all be hidden behind the title of "electronic load."

This was apparently the biggest response *Circuit Cellar* has ever had to a design competition, so well done to everyone who took part! I thought we might see a few good projects, but I certainly wasn't expecting quite the volume or diversity. From CNC machine controllers to HAM radio tools, iPod controllers to iPhone-controlled homes, robot controllers to robotic injection systems, real-time audio filters to bat detectors, this was quite a mix. I look forward to seeing some of the write-ups appearing in future issues of *Circuit Cellar* and on the mbed website. I'm sure they'll inspire others and help them benefit from all the hard work the entrants put in!

During the last few months, I've also had the pleasure of bumping into some of you at shows who were exposed to mbed through this competition. Welcome to mbed! Whilst a fair few discussions started with "I was going to enter, but didn't get around to writing anything up," the conversations still highlighted how many new things were being built, new ideas being explored, and the general enthusiasm for using this modern microcontroller technology. It is wonderful to see such a skilled and innovative crowd building in the mbed community. We'll be working hard to make sure you can be even more productive.

After a long Embedded Systems Conference, I was offered a lift back to the airport by Hexley. It was a great opportunity to chat. I think in that 45-minute drive we may have planned some form of workshop tour of the United States! If we manage to put that together, maybe we'll see some of you there!

Congratulations again, and thank you to everyone who entered. I hope you enjoyed the challenge. ■

Photo 1a—Hexley Ball's winning mbed-based (VI)sualizer electronic load project. b—The design was on display at the Embedded Systems Conference 2011 in San Jose, CA. Attendees had a chance to see the design and chat with Hexley at the NXP Semiconductors booth.

Simon Ford, co-creator of mbed, is a lifelong electronics and computer engineer. He works at ARM, and before starting mbed was technical lead for the ARMv7/NEON architecture now found in most new smartphones.

To view the winning projects from the NXP mbed Design Challenge 2010, go to: www.circuitcellar.com/nxpmbeddesignchallenge/

NEW TOOL FOR EMBEDDED & DIGITAL ENGINEERS

The **MicroBench MB-500** is a compact, affordable unit that integrates with a PC to capture and source common microcontroller interfaces. Traditional oscilloscopes and logic analyzers were originally created to view analog signals or parallel processor buses. Since microcontrollers now keep processing inside, engineers can only access interface signals, often with serial protocols, such as I^2C or SPI. Compared with traditional tools, these signals can easily be seen with the MB-500.

The MB-500 is designed to view and control modern device interfaces. Since these interfaces typically have activity separated by idle periods, the MB-500 captures on activity, not periodic sampling. This enables engineers to see the details of a burst, such as a protocol transaction, along with the context. When engineers need to control device inputs, the MB-500 can source a mix of protocol, digital, and analog signals. This enables the MB-500 to simulate digital or electro-physical signals, or an I^2C or SPI master or slave device.

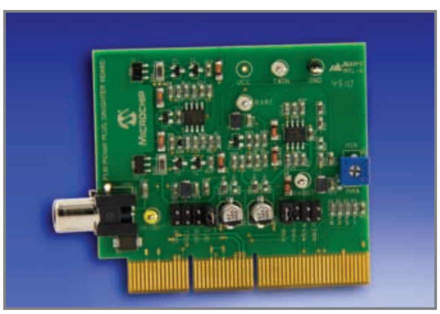
The MB-500 combines a protocol device simulator and monitor, digital pattern generator, logic analyzer, analog waveform generator and programmable power supply, and Anewin's Live Logic 500-MSps logic capture tool with voltmeter.

The MicroBench MB-500 costs **$1,325** to **$2,525**.

Anewin
www.anewin.com

POWER-LINE SOFT-MODEM DEVELOPMENT KIT

The **Power-Line Modem (PLM) PICtail Plus Daughter Board Development Kit** (part # AC164142) enables customers to easily incorporate power-line communication into their products. The power-line soft-modem technology, supported by the daughterboard, enables consumer and industrial products to communicate using the same wiring that provides power, which creates an instant network at a low system and deployment cost. Based upon the scalable dsPIC33F digital signal controller (DSC) architecture, the board uses a binary phase shift keying (BPSK) modulation scheme and operates in the consumer frequency band. The soft modem interfaces with the Explorer 16 Development Board and is supported with royalty-free schematics, demo code, and software in source-code format.



The development kit comes with two nodes, providing point-to-point communication out of the box. The BPSK scheme provides a robust modulation method that performs well in noisy environments. The data rate of the power-line soft modem can be selected in software. Because a large selection of dsPIC33F DSCs is available, customers can choose a device to meet their specific requirements, which optimizes performance and cost.

The PLM PICtail Plus Daughter Board Development Kit costs **$225** and includes two daughterboards and two sets of high-voltage adapter cables.

Microchip Technology, Inc.
www.microchip.com

SOLID-STATE STORAGE FOR THE EMBEDDED MARKET

The rugged **S7 Series** 2.5" SATA SSD provides extreme-environment performance and reliability with high-speed data processing and performance optimizing TRIM support. The devices are well suited for applications that require high transfer speed, continuous optimization, reliability, and uptime. They are designed for long-term exposure to extreme levels of humidity, wide temperature ranges, intense shock and vibration, or high altitude.

The series features eight channels and an SATA (3 Gb/s) bus interface. The devices have high-speed processing for industrial- and military-grade flash storage at 250 MBps read and 200 MBps write. The TRIM-enabled S7 SSD continually purges deleted files to keep performance at its peak for the life of the device. To prevent data loss, the series also features TransferSAFE which ensures complete data transfers even under inconsistent power conditions.

Pricing for the 16-GB series starts at **$295**.

Emphase
www.emphase.com

NEW PRODUCT NEWS

Edited by John Gorsky

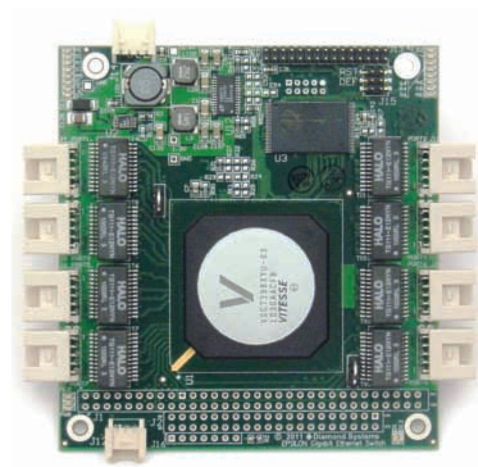
MANAGED EIGHT-PORT GIGABIT ETHERNET SWITCH

The **Epsilon** is a managed Layer 2 Ethernet switch module offering eight 10/100/1,000-Mbps copper twisted pair ports on a PC/104 form-factor board. The switch module can be standalone, without any connection to a single-board computer, or used in conjunction with a host CPU. Epsilon's built-in microcontroller handles configuration and management. The module's on-board memory holds dual-application images, boot code, MAC addresses, and other parameters, and can also be utilized for program execution.

An RS-232 serial port enables communication between Epsilon's on-board management microcontroller and a host processor. Its built-in 8051 microcontroller is also accessible via a web management interface over an Ethernet port.

Epsilon provides a full PC/104 stackthrough bus interface, so it can be integrated into any PC/104 stack. However, the module does not interface to the PC/104 bus and does not require it for its operation. While the PC/104 stackthrough connectors are not installed in the standard version, they are available as a special-order option.

Input power can be provided through the built-in, wide-range 7- to 36-VDC power supply, enabling operation through industrial power sources. Epsilon can also be powered from a 5-VDC source. The module supports fanless operation over –40° to 85°C to support temperature extremes of fixed and mobile applications in both indoor and outdoor environments.



Epsilon is supplied with all required firmware and drivers. The included web interface provides an intuitive GUI for use in configuring and managing switch functionality.

Single-unit pricing for the model EPS-8000-XT starts at **$450**.

Diamond Systems Corp.
www.diamondsystems.com

DIFFERENTIAL PRESSURE SENSOR EVALUATION KIT

The **EK-P3** evaluation kit is designed for testing the digital differential pressure sensors of Sensirion's SDP600 series of differential pressure sensors (DPS). The kit consists of a USB stick that is connected to the SDP610 sensor by an adapter cable. Using the software, which is available online for download, a DPS can be easily tested under realistic conditions with five installation steps. There is no need to program a microprocessor, since the evaluation kit can be connected directly to a PC. The included software displays measured values on the screen and provides the option of exporting the data to an Excel spreadsheet, so data can be easily saved and processed.

The DPSes in the SDP600 family feature I^2C output signals, long-term stability, and are accurate and sensitive, even at low differential pressure values. Their high performance is reached through the thermal flow through principle.

Contact Sensirion or a distributor for pricing.

Sensirion AG
www.sensirion.com

REMOTE STAMP PROGRAMMER

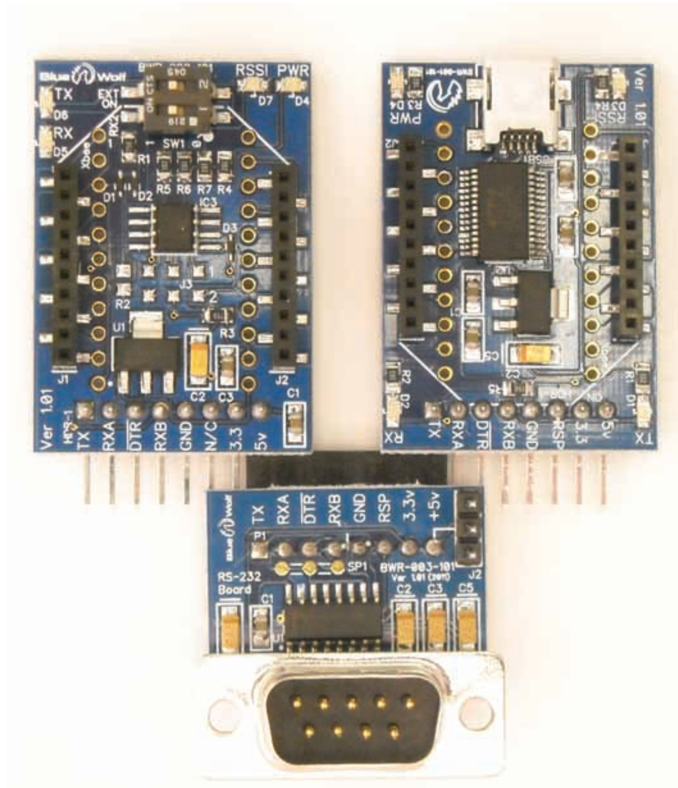
The **FlashFly** system enables users to remotely download BASIC Stamp programs to Parallax's Stamp modules or interpreter chips. FlashFly can also serve as a standalone wireless link for any other microcontroller. The system consists of three interdependent modules—the remote receiver board, an optional RS-232 adapter board, and the XBee USB base transmitter board.

FlashFly's wireless capabilities, in coordination with common Series 1 XBee modules, enable a user with a BASIC Stamp mobile robot or stationary platform to remotely modify a program. The user does not have to connect a robot to a computer before making any programming changes. The system also provides instant feedback data to the DEBUG terminal screen, which enables a user to remotely evaluate his or her program flow and I/O data. FlashFly can also be used in conjunction with any other microprocessor platform that needs to wirelessly transmit data.

The (1 × 8) 0.1″ inline header simplifies breadboard use. Additionally, FlashFly can be used for more advanced XBee experimentation by directly connecting additional wires or headers to the two rows of 0.1″-plated holes on the system. Because FlashFly is flexible and modular, any existing Series 1 XBee modules can be easily reconfigured to work with the system.

Kit prices start at **$44.95**.

Blue Wolf, Inc.
www.bluewolfinc.com



DIGITAL AC/DC PWM CONTROLLER

The **iW1700** zero-power AC/DC digital PWM controller is designed to enable low-cost, energy-efficient, 120/230-VAC offline adapters and chargers (up to 5 W) that consume zero no-load power for cell phones, audio players, digital cameras, and other low-power portable devices.

iWatt's patented adaptive digital PWM/PFM technology sends the controller into sleep mode when the load is disconnected, cutting no-load power consumption to less than 4 mW, or effectively zero. Utilizing digital techniques, the iW1700 features primary-side control to eliminate optocouplers and quasi-resonant switching for low EMI, cycle-by-cycle waveform analysis. It also has a high (up to 72 kHz) switching frequency to achieve no-load charger performance and meet manufacturers' power-supply requirements.

The controller includes an active start-up function that disconnects the start-up resistor after the IC powers up. This eliminates the standby power normally wasted by the resistor, and enables fast (1 s or less) start-up time. The iW1700 maintains better than ±3% output voltage and current regulation over the entire operating line, load, and temperature range, regardless of manufacturing variability or component tolerances.

The iW1700 comes in a standard six-pin SOT-23 package and costs **$0.25** in 10,000-piece quantities.

iWatt, Inc.
www.iwatt.com

NPN

LOW-POWER EMBEDDED COMPUTER & GSM/GPRS MODULE

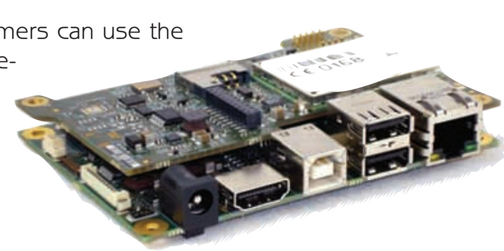
The **RE2** is a high-performance, low-power ARM Cortex A8 embedded computer with an OMAP 3 processing core that integrates a C64x with up to 520-MHz DSP. The RE2's power dissipation is typically only about 2 W at 720 MHz, yet it has the graphics capability to smoothly deliver 720p resolution video content. Target applications include high-definition multimedia, image and voice processing, and data compression/decompression. The board's small size and low power demand also make it well equipped for handheld and mobile applications.

Features include 256-MB LPDDR SDRAM and 512-MB NAND flash, both soldered onboard, plus an optional NAND flash µSD card. Video output is available through a 24-bit RGB TTL connector or a DVI port, and a four-wire touchscreen controller and audio codec interface are also provided. Communications capability comprises a 10/100 Ethernet port, quad USB 2.0 host ports, one USB 2.0 device port, dual RS-232 and a single RS-485 port. Bluetooth, Wi-Fi, and a camera interface are also provided. The board supports Windows CE 6.0, Linux, QNX, and VxWorks.

The versatile expansion port can accept an optional **CM1 Telematics Module** to add GPRS and GPS functionality for real-time vehicle-tracking and telematics applications. The CM1 module is an easy plug-in board that enhances the RE2's onboard ARM A8 Cortex Wi-Fi (802.11 b/g), Bluetooth, and 10/100-Mbit Ethernet capabilities. The two boards connect via a 20-pin header connector and three mounting holes, providing a robust, low-profile assembly, supporting 20-channel satellite GPS with wireless cellular communication over GSM/GPRS.

The RE2 is backward-compatible with the earlier RE1. This ensures that customers can use the latest available processing performance and functionality without incurring maintenance and upgrade problems during the life of their own equipment.

The RE2 costs approximately **$210** in 100-piece quantities while the CM1 costs approximately **$128**.



Blue Chip Technology Ltd.
www.bluechiptechnology.co.uk

DVD HELPS DESIGNERS WITH THE PATENT PROCESS

"Complete A to Z Guide to Performing a Patent Search" is a DVD designed to help you determine if your invention is truly unique and possibly eligible for patent protection. The DVD walks you through the patent search process step by step and allows you to follow along on your computer. It also helps you perform a thorough patent search without disclosing your idea. Information about three important databases, the United States Patent Office website, Google Patents, and FreePatentsOnline is also detailed.

The DVD costs less than **$250**.

Midwest Patent Services, LLC
www.midwestpatentservices.com

SECURITY MANAGER PROVIDES SUPERIOR DATA PROTECTION

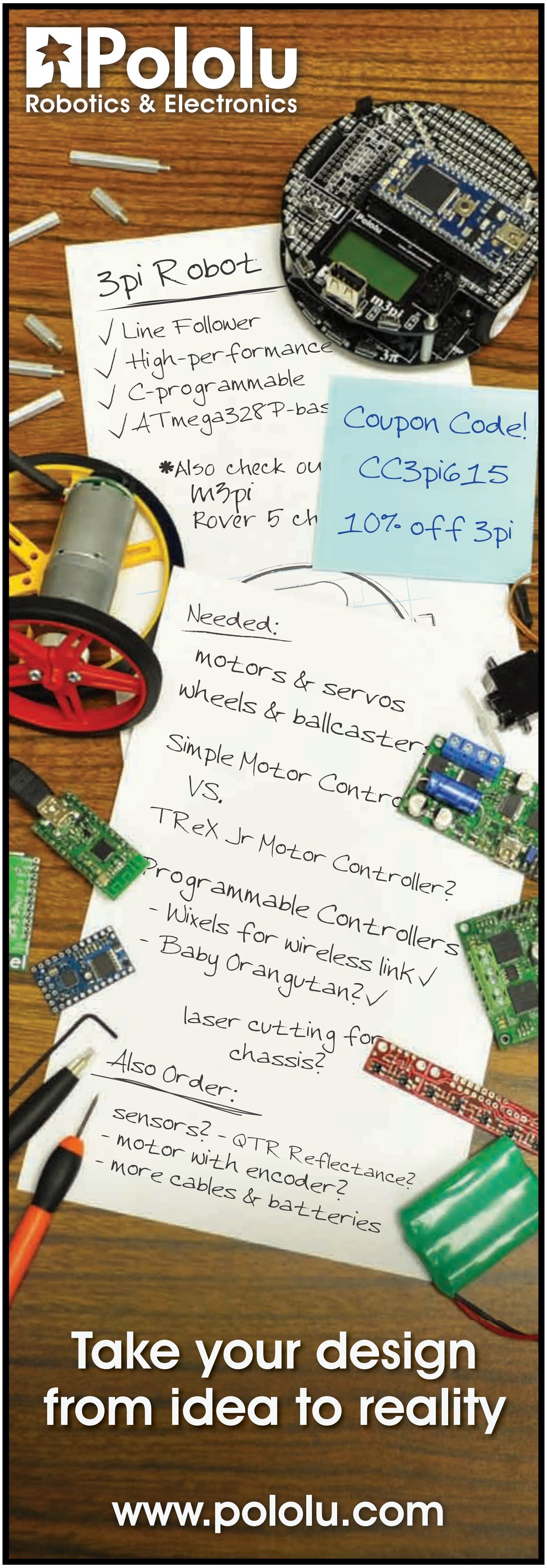
The **DS3660** is a security manager that features 1,024 bytes of nonimprinting memory to securely store sensitive data. With eight 128-byte banks, the on-chip nonimprinting memory can be selectively cleared by end users based on user-specified tamper events. The DS3660 also features an internal 1.8-V bias source to power low-voltage external SRAM. This can be used to store less critical data and can be configured to operate with a low-voltage microprocessor for battery-operated devices. The DS3660 offers a high-level of system security by combining internal tamper monitors with tamper-detection inputs that interface with external sensors. The security manager is well suited for ultra-secure applications, such as government and military systems, as well as applications where more than one tamper trigger needs to be monitored.

The ultra-low-power DS3660 provides tamper detection regardless of the power source. The on-chip battery back-up controller constantly monitors the main power and automatically switches to the battery when the main power is too low or not present. Drawing only 4 µA (typical) of battery current, the DS3660 can retain critical information and monitor tampers while running on battery power.

The DS3660 is fully specified over the –55° to 95°C extended temperature range. It is available in a 7 × 7 mm, lead-free, 49-ball CSBGA package.

Contact Maxim for pricing.

Maxim Integrated Products
www.maxim-ic.com



MULTIFUNCTION LOGIC ANALYZER

The **USBee RX Test Pod** is a PC-based programmable multifunction mixed-signal oscilloscope (MSO) logic analyzer. It features I²C, SPI, Async, SDIO, 1-Wire, USB, I²S, and CAN protocol decoders. In addition, the system features a digital signal generator, a frequency counter, and an integrated multilevel protocol analyzer.

The USBee RX also includes two analog and 18 digital input channels, 100-MHz sampling, 512-MB internal buffers, real-time sample compression, dual 10-bit ADCs, a simultaneous 100-Msps eight-channel digital signal generator, an 8-Msps analog signal generator, ±60-V protection on all inputs, variable logic thresholds, and USB over current protection.

With a single USB connection to your laptop or PC, the USBee RX gives engineers the power to design, prototype, test, and validate their mixed-signal electronic designs.

In combination with the USBee Suite software, which features the patented PacketPresenter, the USBee RX becomes a multilayer protocol decoder for click-and-drag instant decoding of embedded bus transactions, including I²C, SPI, Async, CAN, USB low and full speed, I²S, SMBus, PS/2, SDIO, and 1-Wire, enabling users to debug at the waveform, bus data, or packet level. In addition, users can create custom applications to decode proprietary protocols and control the USBee RX functions using the USBee Toolbuilder source code and library.

The USBee RX costs **$2,245** in single quantities and includes all necessary cables, clips, leads, software, and accessories.

CWAV, Inc.
www.usbee.com

NPN

CIRCUIT CELLAR

Test Your EQ

Edited by David Tweed

Problem 1—Some people will tell you that you can't use a bypass capacitor above its self-resonant frequency, because above that, it no longer behaves like a capacitor. Is this really true?

Problem 2—In a data communications system, what is a "killer packet"?

Problem 3—The C code below converts an unsigned 16-bit binary number into its decimal equivalent, in packed BCD format. Can you describe in simple terms how this works?

```
void adjust (unsigned char *p)
{
  unsigned char t = *p + 3;
  if (t & 0x08) *p = t;
  t = *p + 0x30;
  if (t & 0x80) *p = t;
}
```

```
unsigned long binary2bcd (unsigned int n)
{
  unsigned char bcd[3] = {0, 0, 0};
  int i;

  for (i=0; i<16; ++i) {
    adjust (&bcd[0]);
    adjust (&bcd[1]);
    adjust (&bcd[2]);

    bcd[2] <<= 1;
    if (bcd[1] & 0x80) ++bcd[2];
    bcd[1] <<= 1;
    if (bcd[0] & 0x80) ++bcd[1];
    bcd[0] <<= 1;
    if (n & 0x8000) ++bcd[0];
    n <<= 1;
  }

  return (unsigned long)
(bcd[2]<<16) | (bcd[1]<<8) | bcd[0];
}
```

Problem 4—Does the previous algorithm extend to other sizes of binary numbers?

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

Exploratory Design & Programming

An Interview with Chris Cantrell

Chris Cantrell (MSEE) is a Senior Software Engineer at Avocent in Huntsville, Alabama, who has been reading Circuit Cellar *since the late 1990s. Nine of his articles have appeared in* Circuit Cellar. *Chris's next article—about a Parallax Propeller-based gaming platform—is scheduled to appear in autumn 2011. In May 2011, I interviewed Chris about his engineering background, projects, and years of developing electronic systems and exploring the field of embedded programming. — Nan Price, Associate Editor*

NAN: Tell our readers about where you live and work.

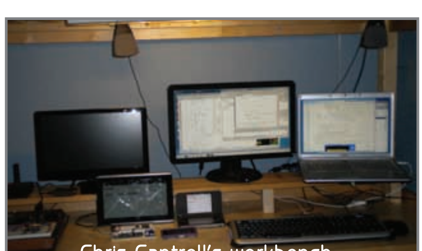
CHRIS: I live in Huntsville, Alabama, which is a high-tech town that grew up around the Marshall Space Flight Center and the Redstone Arsenal. I work at Avocent. We were acquired by Emerson Network Power, but we kept our name as the "Avocent" division. Emerson is a large global company. We formed a team of Java programmers at the Emerson office in Xi'an, China. The smaller team of senior engineers in Huntsville guides the work of the larger team of new graduates in China. One advantage of this new global market is that I can get out and see the world. I have been to Xi'an twice now to meet with the team.

Our product is an embedded Linux network appliance that monitors large servers within a data center. Our box provides KVM, serial sessions, and health monitoring. I lead the web interface team, and I provide general Java technology support to the Java newcomers. The product is a collection of C, C++, and Java processes communicating internally over an ActiveMQ message bus. I am also the Flex developer (a team of one) building the rich Internet application that runs in the user's web browser and talks to the appliance over an XML socket. I am fairly new to Flex, but its core language (ActionScript 3) is similar to Java.

NAN: When did you first become interested in electronics?

CHRIS: When I was a kid, I loved to take apart old broken radios and electronic gadgets. On her morning walks around the neighborhood, my mom would hide any dangerous devices set out for garbage pickup. Otherwise, I would bring them home and tear into them. Finally, she bought me an electronics learning lab from RadioShack. There were dozens of discrete components mounted to the cardboard box with little springs to hold connecting wires. The experiment manual showed you how to build blinking lights and sirens and even a radio. I was hooked.

I entered the "Mims Era" in the 1980s with the Forrest Mims engineering notebooks. In those days, I could ride my motorcycle to the local RadioShack and browse over pegboards of chips, resistors, and capacitors. I could take down a blister package with a 7468 decade counter chip inside. I could feel its weight in my hand, and I could turn the package over and look at the pinouts. The salesman punched my battery-of-the-month club card and I headed home to the breadboard. The LEDs were going to be blinking that night!


Chris Cantrell's workbench

NAN: How long have you been designing and programming embedded systems?

CHRIS: My first computer was a TRS-80 Color Computer with a cassette tape drive and a wonderful "How to Write BASIC" manual. Many, many nights I sat down in front of the TV, typed `10 CLS, 20`..., and waited for an idea to come.

In July of 1984, *Hot CoCo* magazine started a series of articles on hooking up external circuits to the Color Computer through the cartridge expansion port. The first article explained the 6821 PIA chip, which has two 8-bit ports whose pins could be configured for input or output. It was then that I first typed in a `POKE` command on the CoCo and watched LEDs change in the real world. That was the moment when my interest in electronics and programming became one: embedded programming.

NAN: *Circuit Cellar* has published nine of your articles. While they've ranged from robot design to LED movie design projects, it's clear you're interested in object-oriented programming and computer gaming. What draws you to these topics?

CHRIS: Game programming pushes the target hardware to the extremes, and it challenges the programmer to write slick, efficient code that must be creative, fast, and entertaining all at once. So, is it this extreme challenge that draws me? No. Honestly, I just love computer games. When I learned to program on the Color Computer, that's what programs were—games. There were no spreadsheet programs for the 4K Color Computer. I didn't need any office productivity tools. There was no Internet to surf. Writing programs meant writing games. Then, sometime during my formative early years, "making the game" became the game, and I've been playing ever since.

Object-oriented programming appeals to

me on an academic level. Assembly is the imperative language of the computer, and most computer/microcontroller assemblies are basically the same. Functional languages like C separate Assembly into functions and data. OO languages like C++ and Java put functions and data back together again showing that they are intimately connected. I find OO theory and literature, with its enchanting OO vocabulary, deeply fascinating.

NAN: Let's consider two of your articles: "The OO CarolBot" (*Circuit Cellar* 145, 2002) and "Embedded Object-Oriented Programming" (*Circuit Cellar* 187, 2006). Why did you focus so much time on OO programming techniques?

CHRIS: When I started at a previous job at ADTRAN, there was a lot of resistance to using C++ in embedded software. I spent a lot of time arguing with the veteran C programmers trying to convince them that OO languages were not inherently slow. In fact, most of the existing C programs at ADTRAN were written as objects that used polymorphism. They were already paying the cost of OO features built into a language like C++, except their cost was higher for their less efficient, home-grown solutions. I became an OO evangelist. I also taught Java and C++ continuing education night classes at the University of Alabama at Huntsville for several years. If you want to truly learn something, then teach it. The more I learned, the more excited I got, and that passion shows in my *Circuit Cellar* articles on object-oriented design.

NAN: In 2006, you wrote about Blend, a Java-based tool you developed that enables you to create an Assembly file by combining Assembly language and C-like program flow constructs ("Java Utility for Assembly Programmers," *Circuit Cellar* 193). What motivated you to create the tool?

CHRIS: In 2005 I pulled out my old TRS-80 Color Computer and Atari 2600 and began writing Assembly code for them again. I had forgotten how tedious program flow control code was in Assembly. I found myself getting up to speed coding a subroutine only to have to stop and create twisted condition tests and jump code for expressions like `if(A>30 && A<50)`. I really missed the structured flow constructs of Java and C/C++. So, I created an Assembly language preprocessor tool in Java to generate the tedious code for me. My Assembly programming has been much easier ever since.

NAN: Are any of your designs that have appeared in *Circuit Cellar* still in use today?

CHRIS: I still write Assembly programs for the Atari 2600, Nintendo NES, and the TRS-80 Color Computer. I still use the Blend tool to add `if/else/do/while` flow control, though it has definitely evolved over the years. Every Christmas, I break out the LED movie project. I add a new Christmas carol and movie each year and display it in the living room. I added a couple of GameCube controllers to the breadboard and made a PONG game (www.youtube.com/watch?v=pTqB7mghPiA).

NAN: Recently, you built an "all-in-one" joystick, which will be featured in an upcoming issue of *Circuit Cellar*. The module is built around a SparkFun breakout board for SD-MMC cards. Tell us about the design and how the Parallax Propeller chip factors in the project.

CHRIS: I wanted to make a programmable all-in-one joystick game system, which meant the circuit board would have to fit inside the packed joystick casing. The Propeller chip, with its built-in video and PWM (for sound), offered very low component count. Plus I had already built the core Propeller+SDCard circuit for several other projects. Three appeared in *Circuit Cellar* issues 205, 209, and 239.

The Propeller chip has great community following, and I was able to leverage a lot of shared code. Parallax hosts an "object exchange" website where developers can post code and ideas. I found the basis for an 8080 CPU emulator and a SID sound chip emulator in the Parallax forums. The multicore Propeller chip allows these third-party programs to run in isolation in their own dedicated CPUs. In a single core environment, third-party modules have to be blended together into a single executable. The Propeller is the perfect platform for mixing and matching these community modules.

NAN: What project are you currently working on?

CHRIS: I bought one of these Parallax Spinneret Web Servers, and I'm exploring embedded web technologies. The project itself is a universal remote control for my TV, stereo, and DVD player. It is kind of an Internet-to-infrared gateway that serves out webpages to control the entertainment center. The server will detect the kind of browser and deliver HTML that is appropriate. The Nintendo DS will use pure HTML with very large HTML buttons. The server will send a FLEX application to the Droid tablet. I'll create a custom application for the iPod Touch.

NAN: What useful new parts or promising technologies do you recommend to *Circuit Cellar* readers?

CHRIS: It's not new technology, but I believe embedded web servers are about to take off. Web browsers are now everywhere. They are in your phone, iPod Touch, iPad, and Droid tablets. The Internet is everywhere. It is in your home, restaurants, and bookstores.

Microcontrollers now have access to "all-in-one" serial-to-Internet plugs and integrated Ethernet chips. They also have access to cheap memory cards to host web document directories with large media files (pictures and sound). Marketable embedded devices are going to have web servers and accompanying Droid or Apple apps. Your portable browsers are going to be the input/output devices for your projects. ◼

Web project

ICbank
Gateway to Electronics

MP3P DIY KIT, Do it yourself

(Include Firmware Full source Code, Schematic)

· myPIC

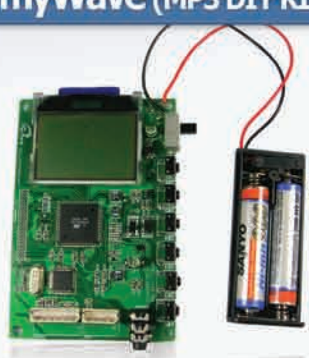
Only
$160 qty 100 **$220** qty 1

Powerful feature

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

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

· myWave (MP3 DIY KIT SD card Interface)



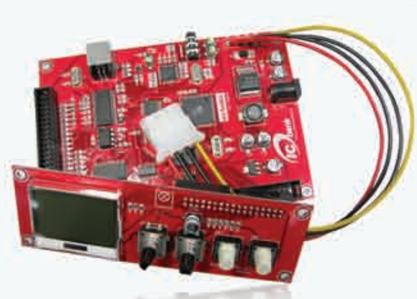
Only
$150 qty 100
$200 qty 1

Powerful feature

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

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

· myAudio (MP3 DIY KIT IDE)



Only
$180 qty 100
$220 qty 1

Powerful feature

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

ICbank
Gateway to Electronics

www.icbank.com

ICbank
Gateway to Electronics

3 PORT INTERFACE
RS-485 to Ethernet Converter

Only
$170

RS-485 to Ethernet Converter

Powerful feature

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

Specification

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

NXP mbed DESIGN CHALLENGE

Winners

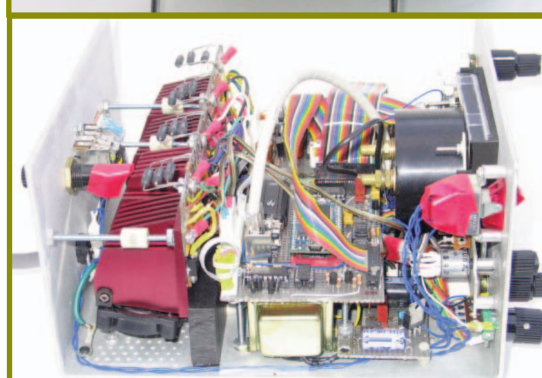
Many design challenges ask participants to toss together some parts, blink a few LEDs, and write a bit of code for a shot at a prize. The NXP mbed Design Challenge 2010 was different. It was administered for the higher purpose of encouraging designers around the world to help make embedded design a more accessible, productive, and community-oriented endeavor.

The rapid prototyping revolution began back in September 2010 when designers from such diverse locations as the United States, United Kingdom, Germany, Australia, India, and Taiwan started working with their mbed NXP LPC1768 prototyping boards. The challenge was to use the board, the mbed online “Cloud” compiler, and the mbed community resources to develop an innovative hardware- or software-based application.

The project submission period ended February 28, 2011. Soon thereafter, the judges began scoring the projects on their technical merit, originality, usefulness, cost-effectiveness, and design optimization. And now the judges’ results are final. Congratulations to all of the winners!

First Prize

(VI)sualizer: A Smart Electronic Load



The innovative mbed-based (VI)sualizer is a lab instrument for profiling solar, chemical, and grid-powered energy delivery devices. The design enables you to examine an energy source’s voltage (V) and current (I) delivery ability. You can use it for a variety of other reasons as well: to calculate the load resistance into which a solar cell delivers maximum power; to measure and capture a battery’s amp-hour capacity; or, using pulsatile loads, to test conventional power supply voltage regulation accuracy. The mbed module in the design accepts input data from local and remote interfaces, commands the desired load current, and monitors operating conditions.

Hexley Ball
United States
hexleyball@gmail.com

For complete projects visit:
www.circuitcellar.com/nxpmbeddesignchallenge

Second Prize

mbos: A Real-Time Operating System for mbed

The purpose of this interesting project was to enhance the mbed environment with a real-time operating system (RTOS). Designed especially for mbed, the "mbos" is a true preemptive multitasking RTOS that's scalable to support virtually any project developed on the LPC1768 platform. This fast RTOS enables you to streamline the development of both simple and complex applications. For example, it's simple enough for a one- or two-task application, but can also support as many as 100 tasks.

Andrew Levido
Australia
andrew.levido@gmail.com

Public Member Functions

	mbos (uint ntasks, uint ntimers=0, uint nresources=0) Create an mbos object.
void	Start (uint idlestacksize=32) Start mbos.
void	CreateTask (uint taskid, uint priority, uint stacksz, void(*fun)(void)) Create an mbos task.
uint	GetTask (void) Get the ID of the current task.
void	SetPriority (uint priority) Set the priority of the current task.
uint	GetPriority (void) Get the priority of the current task.
void	WaitEvent (uint event) Wait for an event or events.
void	SetEvent (uint event, uint task) Post an event or events to a task.
uint	GetEvent (void) Returns the event flag(s) which last caused the task to unblock.
void	CreateTimer (uint timerid, uint taskid, uint event) Create a mbos timer.
void	SetTimer (uint timerid, uint time, uint reload=0) Starts an mbos timer.
void	RedirectTimer (uint timerid, uint taskid, uint event) Redirects an mbos timer.
void	ClearTimer (uint timerid) Stops and clears an mbos timer.
void	CreateResource (uint resourceid, uint priority) Creates an mbos resource.
uint	LockResource (uint resourceid) Locks an mbos resource and temporarily allocates the resource's priority to the calling task.
uint	TestResource (uint resourceid) Tests whether a resource is locked or free, without changing its state.
uint	FreeResource (uint resource) Frees a resource Frees an mbos resource and restores the calling task's original priority.

Third Prize

CNC Panel Cutter

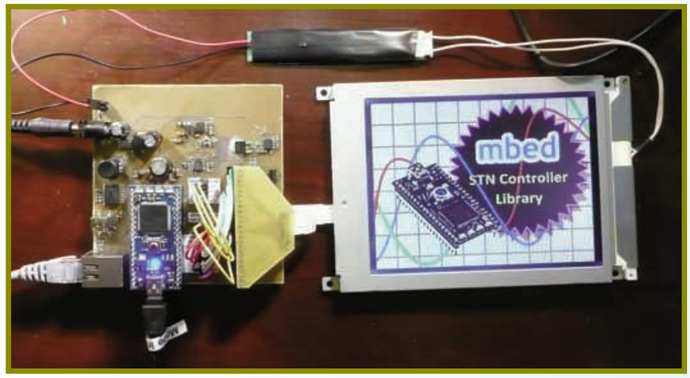
You can use an mbed-based system to control a three-axis milling machine to cut panels for electronic equipment. A smart alternative to a PC program, the self-contained controller enables you to run a milling machine either manually or automatically (following a script) without having to clutter your workspace with a computer. The design controls three stepping motors. Inside the controller are a power supply and a PCB, which carries the NXP mbed module plus the necessary interface circuitry and a socket for an SD card.

James Koehler
Canada
jark@shaw.ca

Honorable Mention

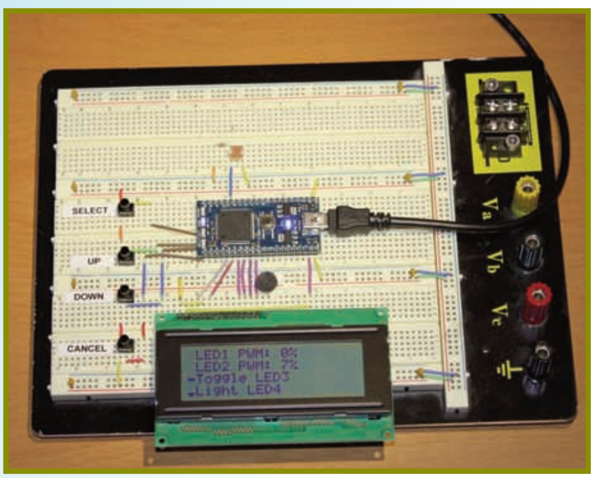
STN LCD Controller Library

When rapid prototyping is a requirement, integrating a graphic LCD in a design can be problematic. It can cost time and money, especially because quality software libraries are scarce. This helpful project solves the problem by eliminating the need for an external LCD controller by implementing the control routine in software. The mbed communicates directly with the LCD driver ICs and handles refreshing the image at 60 Hz or more. The library supports monochrome and color LCDs and also includes experimental support for dual-scan displays.



Matt Bommicino
United States
matt@cafelogic.com

menbed: A Universal Menu System Library



The "menbed" is a universal menu system library for the mbed prototyping board. It enables you to quickly develop an easy-to-use menu system to interact with any other application software executing on an mbed. Helpful interactions include observing internal state variables or analog values, changing program parameters, and calling arbitrary functions. The menu hierarchy is fully customizable and can even be changed dynamically.

Kyle Gilpin
United States
kwgilpin@gmail.com

mbed Net Meter: A Wall Clock Display of Real-Time Network Traffic

You can monitor network data with the mbed-based Net Meter. The clever wall clock-style design displays real-time Ethernet traffic data with three easy-to-read analog dials. The meter shows inbound and outbound data rates, as well as network bandwidth quota usage. The update rate for the "in" and "out" meters is twice per second. The "quota" meter's update rate is once every 8 seconds.

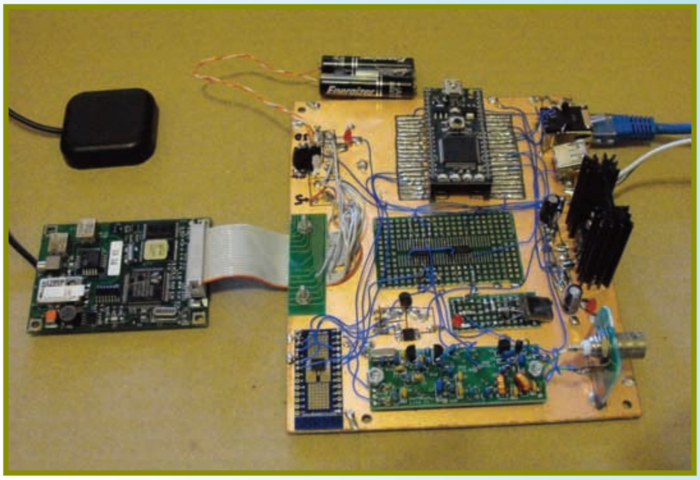
Bruce Lightner
United States
lightner@lightner.net

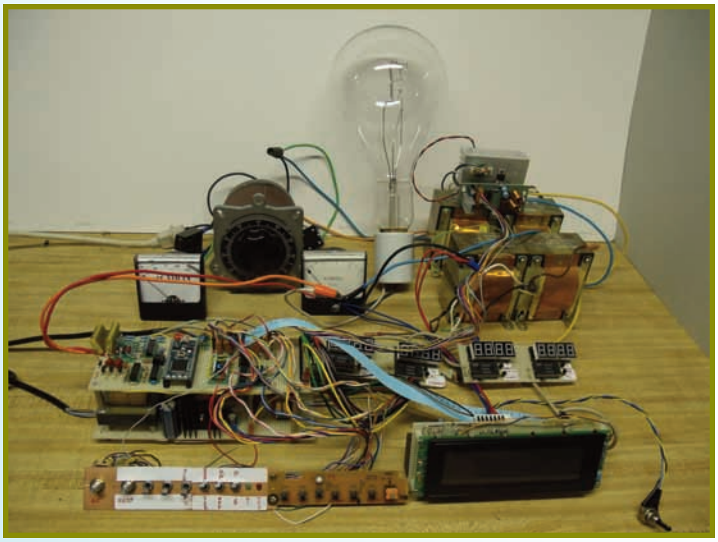
Honorable Mention

QRSS Grabber

QRSS is used by radio amateurs for transmitting slow Morse code from low-power transmitters and receiving them on special "grabbers." This creative project features an mbed at the core of a QRSS receiver, which is an RF receiver that can digitize a small bandwidth of RF signals and send them to a server for processing into a spectrum image for real-time display on a website.

Clayton Gumbrell
Australia
clayton@gumbrell.net





AC Tester

Safety is a top priority when working with electronics and circuits. The AC Tester design is an isolated variable voltage power source that includes an electronic circuit breaker for testing and debugging equipment. An mbed controller displays voltage and current, and it controls the breaker's trip point and response time. In addition, this inventive design can display power factor, VA, and VAR.

Kevin Gorga
United States
kgorga@stny.rr.com

For complete projects visit:
www.circuitcellar.com/
nxpmbeddesignchallenge

NXP mbed DESIGN CHALLENGE

elektor CIRCUIT CELLAR® mbed NXP

by George Novacek (Canada)

Sine Wave Synthesizer

If seasonal time changes cause your WWVB radio-controlled clocks to lose synchronization, building your own WWVB simulator is the first and necessary step for devising a remedy. With a few inexpensive parts, you can build a robust sine wave synthesizer as the heart of a WWVB simulator.

I have several WWVB radio-controlled "atomic" clocks. I never paid much attention to their inner workings except twice a year, during the seasonal time changes, when I noticed that some clocks would take several days to synchronize and others wouldn't synchronize at all. If I adjusted them manually, some would jump an hour forward or back (weeks or even months later), depending on the season. It drove me crazy. I blamed the clock design, but I didn't do anything about it. Until now.

After reading Ed Nisley's articles on building a WWVB clock in *Circuit Cellar* issues 235, 237, and 239, it finally dawned on me that the cause of the sync problem had to be poor reception from the 60-kHz radio sync signal transmitted from Colorado. A few tests confirmed the carrier was buried in noise during the day and barely above noise at night. Consequently, successful reception was infrequent, only during the wee hours of the morning. I realized I needed a WWVB signal simulator before I could do any more work to rectify the situation.

TRANSMISSION & RECEPTION

I purchased a WWVB receiver module, a 100-mm long 60-kHz loopstick antenna, and a 60-kHz resonator from Digi-Key, all for less than $20. The first step to building the simulator was to generate a clean 60-kHz sine wave from the square wave output of the resonator-stabilized oscillator. I measured the Q of the loopstick antenna to be more than 120, so I didn't really need to generate a sine wave. Tests confirmed that when driving the loopstick antenna with a 60-kHz square wave, higher harmonics were sufficiently attenuated to transmit an acceptably low-distortion sine wave. I planned to experiment with different transmission antennas—some with potentially low Q—at a later date, so it made sense to avoid later hardware changes and to design the modulator producing a sine wave carrier now. Since I was going to breadboard the circuit on a Vector board, I wanted a robust, digital design with high immunity to stray capacitances and only a few inexpensive components.

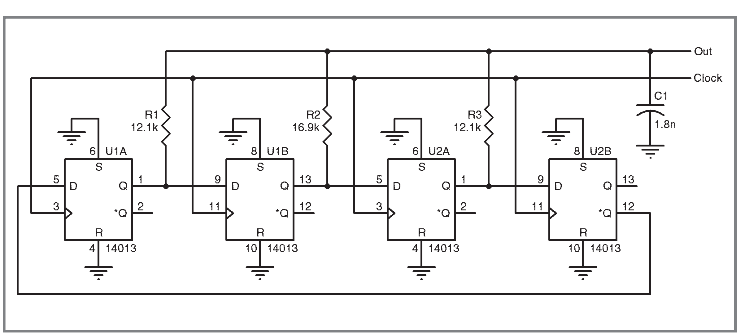


Figure 1—The digital sine wave generator

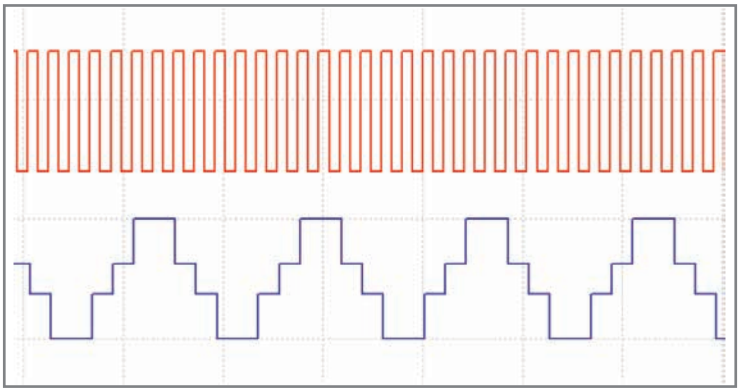


Figure 2—The input and raw output of the synthesizer

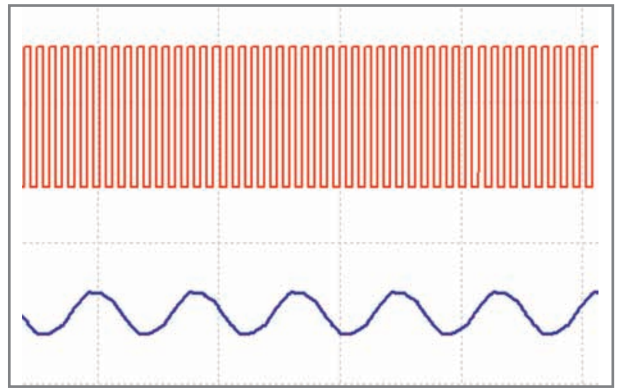


Figure 3—A 60-kHz sine wave

CIRCUITRY

Years ago, after CMOS integrated circuits made their debut, I saw an interesting sine wave synthesizer that I often used at a later date. Other than a few flip-flops, it only needed several 1% resistors. I used that circuit in several projects to generate a linear variable differential transformer (LVDT) excitation signal of 3,200 Hz, low total harmonic distortion (THD) sine wave with excellent frequency and amplitude stability. It has worked on aircraft at temperatures from –40°C to 85°C with no additional compensation. I dug out my old design notes and Figure 1 is the circuit modified for 60-kHz output.

The circuit uses two 4013 dual flip-flops, three standard value 1% resistors, and one capacitor. It took me a few minutes to breadboard it and the result was impressive. With capacitor C1 removed, the input—that is, the clock (red trace)—and the output (blue trace), are shown in Figure 2.

The THD of the output staircase signal is about 20%. By adding C1, the staircase waveform turns into an approximately 500-mV$_{RMS}$ (V_{CC} = 5 V) sine wave with 4.9% THD, as shown in Figure 3. At this point, I hasten to add that when it comes to embedded controllers, audiophiles' standards usually do not apply. I am not aware of any embedded control application

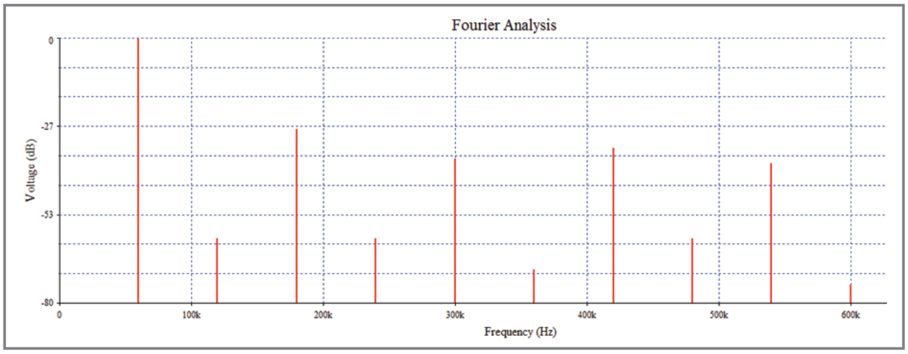


Figure 4—Fourier analysis of the output 60-kHz sine wave

where a 5% THD sine wave would not be considered a sufficiently low distortion.

Frequency stability of the 60-kHz sine wave depends on only the resonator stability, while the amplitude depends mainly on the power supply, provided a zero temperature coefficient capacitor C1 is used. A sharper low-pass filter could reduce the 4.9% THD; but, in my case, I couldn't see a need for it. Other than the stable power supply, the temperature stability of the filter is the critical part of the design to ensure amplitude stability over a wide temperature range. Because I don't plan on using the circuit outside of my workshop, the single zero temperature coefficient C1 works well. Figure 4 shows the spectrum of the sine wave with C1 in place. Table 1 lists the actual decibel levels of the first 10 harmonics. This is more than sufficient to drive the RF transmitter with a low Q antenna.

It is possible to design the synthesizer with more stages to reduce the level of the harmonic distortion. If you want to build more stages, remember that the last stage is always inverting the signal and has no resistor (see Figure 1). If we normalize the resistor values (in Figure 1 the normalized value "1" corresponds to 12.1 k Ω), a six-stage synthesizer, for example, will have normalized values of 2.000, 1.155, 1.000, 1.155, and 2.000. The harmonic content will be significantly reduced compared with the four-stage synthesizer. With just one filter capacitor (e.g., C1 in its four-stage sibling in Figure 1), the 60-kHz sine wave exhibits a mere 0.14% THD. An eight-stage synthesizer will have 2.613, 1.413, 1.083, 1.000, 1.083, 1.413, and 2.613 normalized values.

A small price to pay for this performance and simplicity is that the clock frequency must be the desired sine wave frequency multiplied by the number of stages multiplied by two. To synthesize the 60-kHz sine wave in four stages, a 480-kHz clock (i.e., 60 × 4 × 2) is needed. Rather than looking for a 480-kHz resonator, I simply used a 4046 phase-locked

Table 1—Levels of harmonic frequencies for the four-stage synthesizer

Frequency	Harmonic frequency	Voltage	Decibels
60,000 Hz	1	537.53 mV_{RMS}	0
120,000 Hz	2	0.504 mV_{RMS}	–60.56
180,000 Hz	3	22.408 mV_{RMS}	–27.6
240,000 Hz	4	0.509 mV_{RMS}	–60.47
300,000 Hz	5	8.094 mV_{RMS}	–36.44
360,000 Hz	6	0.1704 mV_{RMS}	–69.99
420,000 Hz	7	11.504 mV_{RMS}	–33.39
480,000 Hz	8	0.509 mV_{RMS}	–60.46
540,000 Hz	9	6.959 mV_{RMS}	–37.76
600,000 Hz	10	0.102 mV_{RMS}	–74.48

loop (PLL) with a 60-kHz resonator and a divide-by-8 counter.

SIMPLE YET ROBUST

I am quite happy with the performance, simplicity, and robustness of the circuit. It shows that even old technology can produce quick and satisfactory results. The synthesizer is only the first step, but it is also necessary for the repeater, which is the end product. I built the modulator and a pulse sequencer for the simulator using an ATtiny85. The time keeping and pulse generation for the repeater is currently performed by an ARM mBed controller. I am considering adding the synthesizing and modulation functions to it as well. At a later date I plan to revisit the topic, discussing the rest of my WWVB inconsistent sync solution. ■

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

National Institute of Standards and Technology (NIST), "WWV and WWVH Time Code Format," http://tf.nist.gov/stations/wwvtimecode.htm.

———, "WWVB Radio Controlled Clocks: Recommended Practices for Manufacturers and Consumers," 2009, http://tf.nist.gov/general/pdf/2422.pdf.

SOURCES

ATtiny85 Microcontroller
Atmel Corp. | www.atmel.com

Loopstick antenna and resonator
Digi-Key Corp. | www.digikey.com

AFET Loopstick antenna
HKW-Elektronik | www.hkw-elektronik.de

by Toby Baumgartner (USA)

The TROBOT

A Miniature Articulated Robot

The TROBOT is a compact six-axis robot powered by RC-style servo motors. An MCU acts as a servo controller interface between the robot and a PC running robot programming software.

The TROBOT is a miniature six-axis articulated robot driven by small RC-style servo motors (see Photo 1). It is modeled after a much larger ABB Robotics Products industrial robot, the IRB 6640, which weighs in at about 3,000 lbs, can lift up to 500 lbs, and can move at speeds up to 23′ per second. At about one-ninth the size of the IRB 6640, my TROBOT would be squashed in the blink of an eye if the two were to meet unexpectedly.

Like most projects, this one involved various stages of development. The second version of the design, TROBOT 2.0, featured a WIZnet W7100 Internet MCU (see Photo 1a). The third version, TROBOT 3.0, was built with a Texas Instruments (Luminary Micro) LM3S9B96 microcontroller (see Photo 1b). I'll focus on the latter in this article. A useful schematic is posted on the *Circuit Cellar* FTP site.

DESIGN SIMULATION

As an experienced robotic controls engineer, I've used ABB's RobotStudio software to develop large industrial robot applications in the automotive, forging, die-casting, and foundry industries. As an electronics tinkerer, I've been using small RC servo motors for projects since I was a kid. Recently, I gained access to a laser engraver and cutting machine to further support my acrylic and plastics fabricating projects. The TROBOT is the end result of combining all my interests.

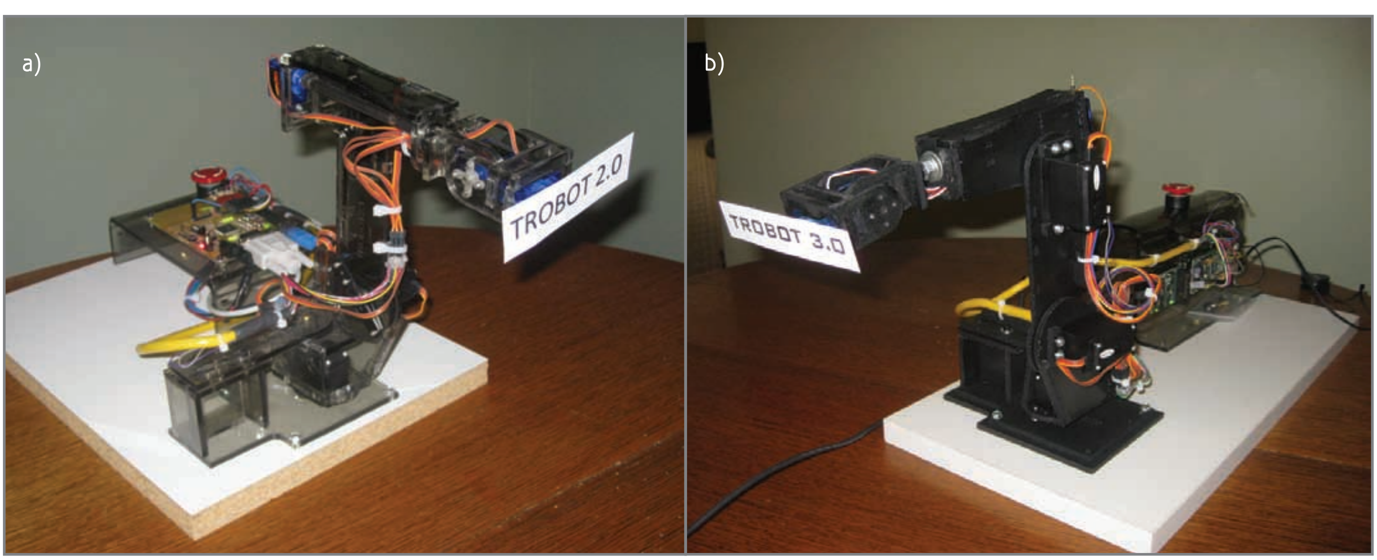


Photo 1a—The TROBOT 2.0 featured the WIZnet W7100. **b**—The TROBOT 3.0 features a Texas Instruments LM3S9B96.

Photo 2—A simulation of the full functionality and robot control of the IRC5 on a PC-based application

As I was learning how to use microprocessors for Ethernet applications, a robot controller seemed like a great device to "talk" to. Also knowing the power of ABB's RAPID robot programming language, it only seemed fitting that I create a robot.

I have years of experience working with acrylic and have used the material for hundreds of projects. Recently, I figured out how to use a Trotec Speedy 100 Laser Engraver, and I am now able to cut acrylic with greater detail and precision than I ever thought possible. So, I sat down with AutoCAD and started designing parts. After a lot of trial and error, the first TROBOT prototype was born.

RobotStudio is a powerful software tool used for simulation and offline programming of the full line of ABB industrial robots. Its 3-D virtual environment makes it possible to set up, simulate, and program complex robot cells in a virtual world. ABB's virtual robot controller technology can simulate almost all of the features of RAPID on ABB's IRC5 industrial robot controller. The IRC5 handles the motion control and logic to run ABB's entire line of industrial robots. Another advantage is that ABB "opened" the Ethernet port access of RobotStudio. This enabled the virtual robot to communicate as if it were a real robot controller, which created several new possibilities.

RobotStudio can simulate the full functionality and robot control of the IRC5 on a PC-based application (see Photo 2). Included with the project files is a RobotStudio executable that demonstrates the robot cell. The ABB "virtual" robot controller can also use the Ethernet port on the host PC, which made this project possible.

TROBOT 3.0 CONSTRUCTION

The TROBOT 3.0 utilizes an LM3S9B96 evaluation kit. I wrote a custom RAPID application in RobotStudio to perform the communication between the TROBOT and the evaluation kit. Programming in RAPID is similar to programming in C. It enables you to create routines and functions and use specialized robot motion commands.

Although RobotStudio makes it possible to model custom mechanisms and robots, the TROBOT was modeled after an IRB 6640 2.55-m reach robot for simplicity. It was scaled down to about one-ninth the original size (see Figure 1).

Keeping the scaled relative distances between the joints of the TROBOT the same as those of the ABB enabled me to use the IRB 6640 model directly in RobotStudio without having to create a custom kinematic model for the mechanism of the TROBOT.

After the parts for the TROBOT 3.0 were designed in AutoCAD, they were laser cut from 3/16″ black ABS sheet. The entire design was cut from an 11″ × 16″ section (see Photo 3). I chose ABS because it is a durable plastic and can be chemically welded (bonded) to form tough glue joints. Previous TROBOT versions—such as the W7100-based TROBOT 2.0—were made from acrylic, which is very brittle and can break if it becomes overstressed.

Upon powering up the TROBOT 3.0 application, the LM3S9B96 board first configures the SafeRTOS tasks, which include a web server, locator service, UART communications, a heartbeat LED, and PWMs for motion control. It then uses DHCP to obtain an IP address and begins socket listening at port 5,000.

The virtual robot is capable of socket communication through RobotStudio. The RobotStudio application opens up a socket and establishes socket communication to the LM3S9B96 processor running the TROBOT (see Photo 4).

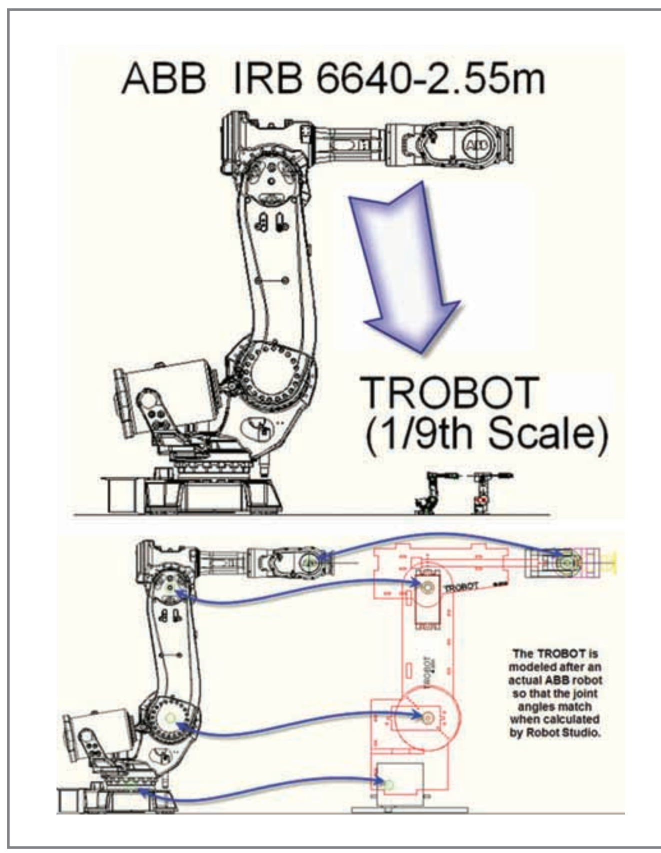


Figure 1—The TROBOT was modeled after an IRB 6640 2.55-m reach robot. It was scaled down to about one-ninth the original size.

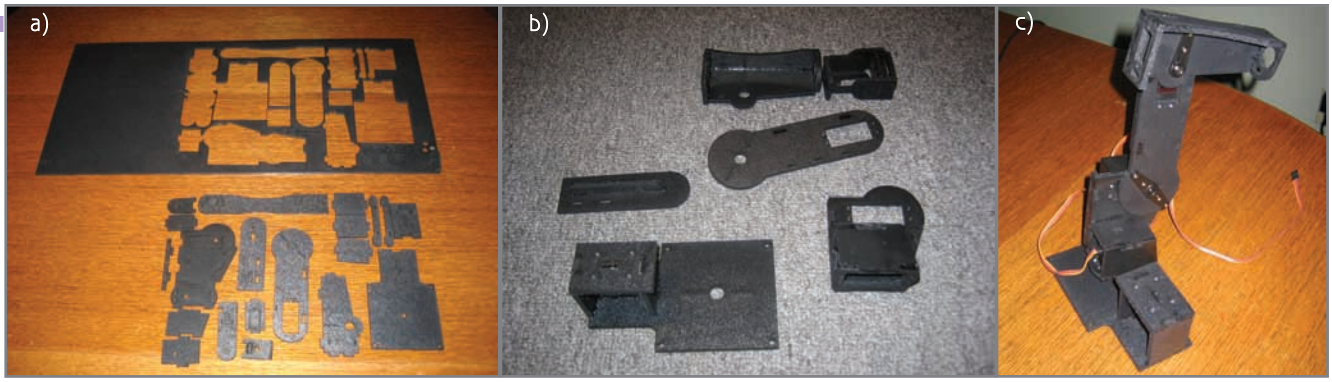


Photo 3—The parts (a and b) for the TROBOT 3.0 were laser cut from a 3/16" black ABS sheet. c—The assembled parts

I created a sample test application (Motion/Comm Test #1) to demonstrate whether the TROBOT's motion matches that of the ABB RobotStudio model. In the application, the virtual robot moves to a position and then sends data to the TROBOT to move to the same position. A video of the TROBOT is available on the *Circuit Cellar* FTP site.

THE SERVOS

An RC servo is controlled by a PWM signal—typically controlled by a 1- to 2-ms pulse that repeats every 20 ms. The 1.5-ms signal tells the servo to center, the 1-ms signal tells it to full clockwise travel, and the 2-ms signal tells it to full counter-clockwise travel. Servos are available in many different configurations for the amount of travel possible. Some standard servos can move ±30°, some up to ±90°, and some have full or even continuous rotation. More advanced digital servos are custom programmable and can be configured as necessary for the task.

I used two servos for this project. Axes 1, 2, and 3 use a MG946R 55-G high-torque digital servo. Axes 4, 5, and 6 use miniature SG-90 9-G servos. I found them at hobbypartz.com.

TEACH PENDANT

ABB robots can be programmed in several ways. They may be programmed "offline," meaning you can write your program using a computer and any text editor and then load it (via USB flash drive or FTP) into the robot for testing and debugging. RobotStudio makes it possible

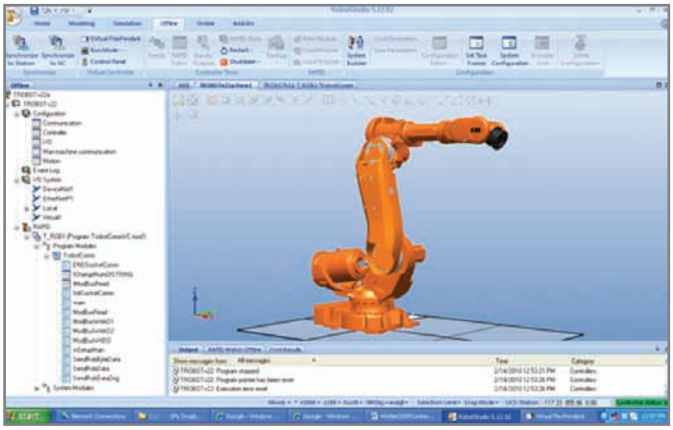


Photo 4—The RobotStudio application opens up a socket and establishes socket communication to the LM3S9B96 running the TROBOT.

to work "online," where you can make/edit your program right in RobotStudio, so all of your changes are live and immediately go into effect. You can also write and edit your program through the Teach Pendant.

Standard ABB robots using an IRC5 controller come with a Teach Pendant to jog and program (teach) the robot. ABB describes theirs as a "Flex Pendant." It is actually a Windows CE-based touchscreen device that uses Ethernet communication to the IRC5. RobotStudio uses a "Virtual Flex Pendant" to interact with the operator. The Flex Pendant and the TROBOT setup screens are used to change network settings, open and close socket communication to the TROBOT, and initiate program and test sequences (see Photo 5).

PROGRAM CODE

I started with the `qs-safertos` sample application provided with the LM3S9B96 evaluation kit. This sample code enabled me to get the basic communication portion talking to RobotStudio. The `lwIP` module provided all of the resources needed to establish basic socket communication and pass data over the Ethernet from the computer running RobotStudio to the LM3S9B96.

Modifying the SafeRTOS made it easy to create my own PWM task which handles configuring and updating the onboard PWMs. The LED task was left intact but is now used as a basic "heartbeat" signal indicator to let me know the board is on and functioning.

The PWM task continuously runs and updates the servo values. The servos' velocity are handled in a group of routines that compares the current servo position with the target position and then adjusts the PWM values according to the rate value.

The values that are valid to send to each servo are numerical, 5,000 to 10,000. The 5,000 value is full counter-clockwise, the 10,000 value is full clockwise, and a value of 7,500 centers the servo.

The LM3S9B96 TROBOT RTOS application is set up to receive a socket message simply containing a string of data. It is formatted as follows:

```
7500:7500:7500:7500:7500:7500;
```

The message contains six colon-delimited (:) 16-bit numbers followed by a semicolon (;). These numbers represent the values for the PWM outputs for each servo. Listing 1 is sample code that is used to extract the numbers from the string data and assign them to the variables used to set the PWM outputs for each servo motor.

The LM3S9B96 SafeRTOS receives data over the Ethernet. The PWM values are controlled to move the servos at the programmed speed. The code and logic shown in Listing 2 are used to control the robot's velocity and set the PWM values based on the speed value in the program or through the web-enabled interface.

RAPID CODE

An application was written in RobotStudio to set up and establish socket communication on the robot

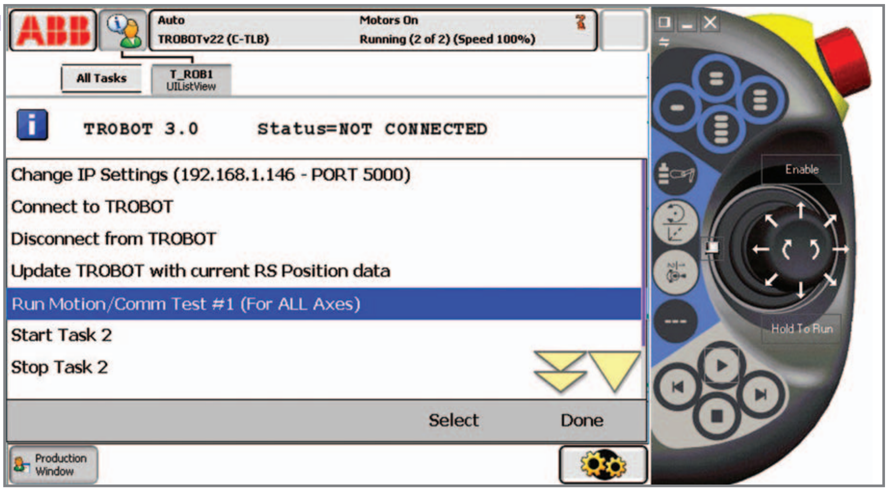


Photo 5—An example of the FlexPendant and the TROBOT set-up screens. These screens are used to change network settings, open and close socket communication to the TROBOT, and initiate program and test sequences.

side. Once basic communication was established, the LM3S9B96 board was able to echo back whatever data I sent to it from the RobotStudio application. An example of the RAPID code for RobotStudio is posted on the *Circuit Cellar* FTP site. The routine is used to initialize the socket communication in the virtual robot.

Once communication is established, the code is used to read the joint angles from the virtual robot, format them into a string, and send them as a socket message. From that point, it is just a matter of sending the correct data to the LM3S9B96 board and using it as a PWM driver.

WEB SERVER & LOCATOR

The LM3S9B96 board also functions as a web server. I added a TROBOT 3.0 webpage for adjusting the robot's velocity and sending direct position command data to the robot (see Photo 6).

I modified and revised the original HTML files supplied with the SafeRTOS application to work with the TROBOT. I updated the HTML in a text editor and used a simple batch file to run the MakeFS utility to update the HTML

Listing 1—Sample code used to extract the numbers from the string data and assign them to the variables used to set the PWM outputs for each servo motor

```
/* extract first string from string sequence */
str1 = strtok(str, ":");
printf("%i: %s\n", x, str1);
x++;
str2 = strtok(NULL, ":");
printf("%i: %s\n", x, str2);
x++;
str3 = strtok(NULL, ":");
printf("%i: %s\n", x, str3);
x++;
str4 = strtok(NULL, ":");
printf("%i: %s\n", x, str4);
x++;
str5 = strtok(NULL, ":");
printf("%i: %s\n", x, str5);
x++;
str6 = strtok(NULL, ";");
printf("%i: %s\n", x, str6);
/*  Set variables equal to the token values  */
axis1 = atoi(str1);
axis2 = atoi(str2);
axis3 = atoi(str3);
axis4 = atoi(str4);
axis5 = atoi(str5);
axis6 = atoi(str6);
```

Listing 2—The code and logic used to control the velocity of the robot and set the PWM values based on the speed value in the program or through the web-enabled interface

```
static void
PWMSpd1(unsigned long g_ulRate)
{
        if (g_cur_PWM1 < g_ulPWM1){
        g_cur_PWM1++;
        bInPosAxis1 = 0;}
        else if (g_cur_PWM1 > g_ulPWM1){
        g_cur_PWM1--;
        bInPosAxis1 = 0;}
        else {
        g_cur_PWM1 = g_ulPWM1;
        bInPosAxis1 = 1;
        }
}
        while(1)
        {
        while (g_cur_PWM1 != g_ulPWM1 ||  g_cur_PWM2
        != g_ulPWM2 ||  g_cur_PWM3 != g_ulPWM3
        ||  g_cur_PWM4 != g_ulPWM4 ||  g_cur_PWM5
        != g_ulPWM5 ||  g_cur_PWM6 != g_ulPWM6)
        {
        PWMSpd1(1);
        PWMSpd2(1);
        PWMSpd3(1);
        PWMSpd4(1);
        PWMSpd5(1);
        PWMSpd6(1);
        //
        ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_0,
        g_cur_PWM1 + Trim_PWM1);
        ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_1,
        g_cur_PWM2 + Trim_PWM2);
        ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_2,
        g_cur_PWM3 + Trim_PWM3);
        ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_3,
        g_cur_PWM4 + Trim_PWM4);
        ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_4,
        g_cur_PWM5 + Trim_PWM5);
        ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_5,
        g_cur_PWM6 + Trim_PWM6);
        //
        TBdelay(Spd_PWM1);
        }
        //
        xTaskDelay( 1 );
        //
    }
```

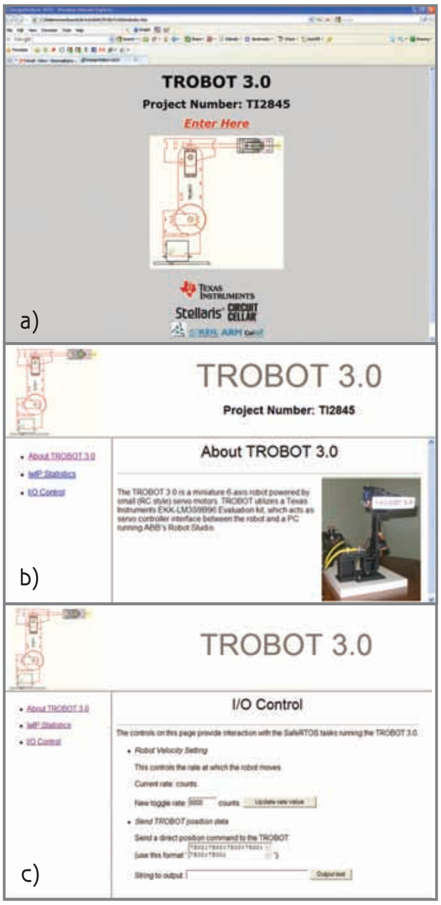


Photo 6—Images **a**, **b**, and **c** are screen shots from the TROBOT 3.0 webpage.

code each time I made changes. I simply type "`MakeFSbat`" and press Enter.

The I/O control screens are now used to set the robot's velocity and to send a command to directly move the servos to any position. For example, to immediately center all of the servos, set the output string to:

7500:7500:7500:7500:7500:7500;

The "Locator Service" originally supplied with the `qs_safertos` sample code was left intact because it is a useful tool for locating the IP address of a TROBOT on a network. The Stellaris Board Finder utility can be used to locate a specific LM3S9B96. If many different TROBOTs were running on one network, it would be useful for determining which is the correct one to communicate with.

DESIGN ITERATIONS

The TROBOT 3.0 design is the result of several months of work. As its name implies, the project is the third iteration of the design and the most advanced version yet. Over time, by leveraging the power of parts such as the W7100 and then the LM3S9B96 processor, I was able to incorporate new features and evolve the system.

Toby Baumgartner (tbaumg@gmail.com) is a Robotic Controls Engineer for Rimrock Corp. in Columbus, Ohio. He has more than 15 years of experience designing complex automation and robotic solutions for a variety of industrial environments and applications. Toby earned a BSEET degree from DeVry University and spends his free time designing embedded projects and robots as well as programming in a variety of languages. Go to www.youtube.com/tbaumg to watch Toby's project videos.

PROJECT FILES

To download the project files, go to ftp://ftp:circuitcellar.com/pub/Circuit_Cellar/2011/252.

RESOURCES

ABB Robotics Products, "RobotStudio Overview," www.robotstudio.com.

Texas Instruments DesignStellaris 2010 Design Contest, www.circuitcellar.com/designstellaris2010.

WIZnet iMCU Design Contest 2010, www.circuitcellar.com/iMCU.

SOURCES

IRB 6640/IRB 6640ID Robots, RobotStudio software, and IRC5 Robot controller
ABB Robotics Products AB | www.abb.com

T-Pro MG946R 55G High-torque digital servo and T-Pro Mini servo SG-90 9G
www.hobbypartz.com

LM3S9B96 Microcontroller
Texas Instruments, Inc. | www.ti.com

Speedy 100 Laser engraver
Trotec Laser, Inc. | www.troteclaser.com

W7100 Microcontroller
WIZnet | www.wiznet.co.kr

Elektor Shop

The world of electronics at your fingertips!

Kits & Modules

A versatile platform for learning and experimenting

Elektor Proton Robot

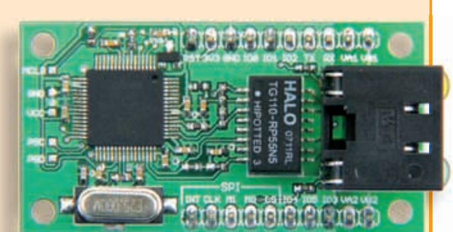
The Proton robot from Elektor is a versatile platform that's suitable for students, enthusiasts and professionals alike. The robot supports a broad spectrum of sensors and actuators. The most important feature is undoubtedly the fact that you, as the user, can personally decide which microcontroller to use to control the robot and which language and/or programmer you want to use for this purpose. When you buy the robot, you can choose from an add-on PCB for the PIC15F887 or a PCB for the AVR ATmega32, but with a piece of prototyping board and a minimum of soldering work you can also control the robot with virtually any microcontroller, as long as it supports a bit of digital I/O, a couple of analog inputs, I2C and RS232. You can purchase the robot as a complete kit or ready-made and it's easy to adapt the system to your own wishes and needs, as well as the available budget. This is an ideal opportunity to invest in a unique combination of knowledge and fun!

Pico C Meter

Even upmarket digital multimeters boasting a built-in capacitance meter are useless if you want to check out tiny capacitances like 2.7 pF or 5.6 pF. Usually, you're tied to a lowest measuring range of 2000 pF, which is a good laugh to RF designers and radio amateurs. Although at 3.5 digits the DMM's resolution is 1 pF, any measurement below 200 pF or so will produce coarse if not ridiculous results. Pico C does a far better job. Beating many DMMs hands down, this little instrument easily measures capacitances down to fractions of a picofarad.

Kit of parts incl. Elektor Project Case, programmed microcontroller, LCD and PCB

Art.# 100823-71 • $118.40



NetWorker

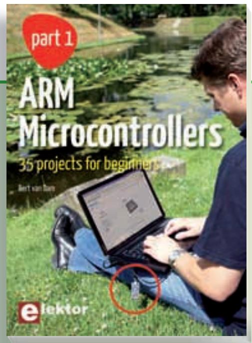
An Internet connection would be a valuable addition to many projects, but often designers are put off by the complexities involved. The 'NetWorker', which consists of a small printed circuit board, a free software library and a ready-to-use microcontroller-based web server, solves these problems and allows beginners to add Internet connectivity to their projects. More experienced users will benefit from features such as SPI communications, power over Ethernet (PoE) and more.

Module, ready assembled and tested

Art.# 100552-91 • $85.50

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

Books

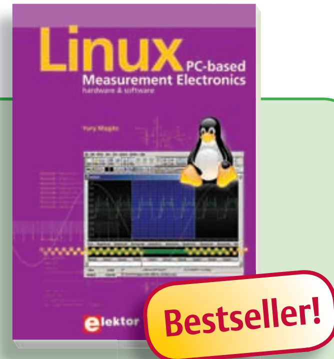


Associated starter kit available

ARM Microcontrollers

This is the perfect book for people who want to learn C and who want to use an mbed ARM microcontroller in an easy and fun way. The mbed NXP LPC1768 uses cloud technology. This means you do not need to install software on your PC in order to program the mbed! The only thing you need is a browser and a USB port on your PC. No previous experience or knowledge required. You can get access to your project from any PC anywhere in the world and continue working on it. When you are done a few mouse clicks transfer the program to your mbed hardware.

250 pages • ISBN 978-0-905705-94-1 • $47.60

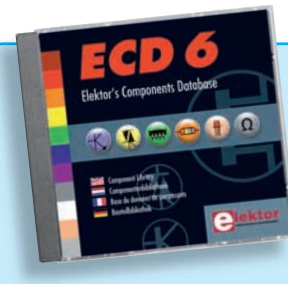


Linux-based applications

Linux – PC -based Measurement Electronics

If you want to learn how to quickly build Linux-based applications able to collect, process and display data on a PC from various analog and digital sensors, how to control circuitry attached to a computer, then even how to pass data via a network or control your embedded system wirelessly and more – then this is the book for you! Both hardware and software aspects of designing typical embedded systems are covered in detail with schematics, code listings and full descriptions.

264 pages • ISBN 978-1-907920-03-5 • $47.60

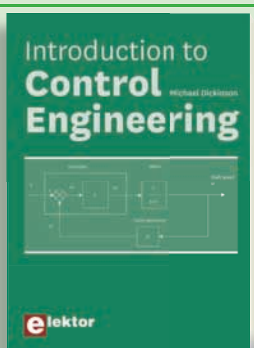


Elektor's Components Database

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.

ISBN 978-90-5381-258-7 • $40.20



Solutions for control system applications

Introduction to Control Engineering

This book is intended as a source of reference for hardware and software associated with instrumentation and control engineering. Examples are presented from a range of industries and applications. Throughout the book, circuit diagrams and software listings are described, typical of many measurement and control applications. The hardware and software designs may be used as a basis for application by the reader. The book contains examples of PIC, PLC, PAC and PC programming.

164 pages • ISBN 978-0-905705-99-6 • $44.40

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

Elektor US
PO Box 180
Vernon, CT 06066
USA
Phone: 860-875-2199
Fax: 860-871-0411
E-mail: order@elektor.com

All articles in Elektor Volume 2010

DVD Elektor 2010

This DVD-ROM contains all editorial articles published in Volume 2010 of the English, 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 978-90-5381-267-9 • $37.90

CD/DVD-ROMs

by Indranil Majumdar (India)

MCU-Based GSM Connection

A Fixed Cell Phone with Emergency Auto-Dial

A little embedded intelligence (EI) goes a long way. This microcontroller-based design features a GSM platform, a controller, a keypad, and a handset for easy dialing and wireless calling.

Don't call me crazy for trying to design things that are already on the market. There's a method to my madness. I'm interested in making existing technologies better. Most electronics—from satellites to dishwashers—have some form of low-level artificial intelligence or embedded intelligence (EI). But engineers are never complacent. We're always trying to build in additional intelligence. And so I assure you that we aren't far from a day when ubiquitous consumer gadgets (e.g., cellular phones, MP3 players, and home appliances) will have high-level EI. (Get ready for the "gadgetoids.") Our products will work in unison to protect the interests and the welfare of their users.

Photo 1—Don't let the retro casing fool you (**a**). Inside is a Microchip Technology PIC24FJ64GA002 microcontroller, a GSM board with an antenna, and a connected keypad (**b**).

The aforementioned ideas are the guiding principles for my global system for mobile communications (GSM) wireless phone design. The low-cost system has the look of a "retro" phone with a hook switch and cradle (see Photo 1a). But that's where its old-fashioned characteristics end. The phone's MCU-driven EI—along with its GSM platform, touchpad, and concealed antenna—proves it is truly a 21st-century design (see Photo 1b).

The system also features emergency autodialing functionality and a fire alarm that can transmit precanned SMS messages to preprogrammed telephone and mobile phone numbers. For instance, consider an elderly person who needs to call a pharmacist, a physician, or a family member. Memorizing all of the numbers or looking for the frequently called numbers before dialing is a common problem. My wireless phone design uses EI to address the problem. A user can simply press the "A" key for 911 alerts or press the "B" key for a medical emergency. Table 1 shows the phone's various functions.

DESIGN ESSENTIALS

The phone is built around a Microchip Technology PIC24FJ64GA002 16-bit microcontroller that performs the GSM module initialization, dials, and sends the commands for GSM module dial-up. The phone's Microchip Technology MCP9700 thermistor can send a fire

Sl Number	Keypad button	Functional description
1	Numeric keys (0 to 9)	Dial in a telephone number
2	D	<Enter> Make a call
3	C	Cancel the call
4	A	911 Emergency alert SMS
5	B	Medical alert SMS
6	Temperature sensor	Fire alert SMS
7	* and #	Unused

Table 1—Keypad functions

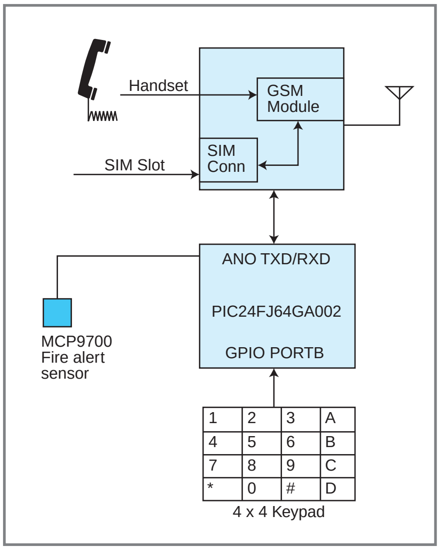


Figure 1—An iWOW GSM evaluation board with a TR-800 Triband GSM/GPRS module and a SIM cardholder with an SMA antenna port/GSM antenna. The board features a Microchip Technology PIC24FJ64GA002 16-bit microcontroller and Microchip MCP9700 thermistor, a 16 × 16 membrane-type keypad, and miscellaneous components bought at a local market.

alert to a preprogrammed number (see Figure 1).

GPIO Port pins	Alternate function mapping
GPIO-RB0/RB1	ICD 2 Debugger program pins, PGC, and PGD, respectively
GPIO-RB2/RB3	UART0 TXD/RXD pins, respectively
GPIO-RB5	Hook switch state sensor (active low)
GPIO-RB8	Output – GSM module reset line (active low)
GPIO-AN0	AN0 – ADC input for the temperature sensor
GPIO-RB6/7/10/11	Keypad 4 × 4 matrix row inputs (pulled high)
GPIO-RB12/13/14/15	Keypad 4 × 4 matrix column drive outputs (active low)

Table 2—The GPIO port pins/alternate function mapping

You can power the portable phone with either a back-up battery or through an AC mains wall outlet adapter.

The phone's anatomy is simple. Flashy LCDs, speakerphone features, and voicemail aren't included. It's just a 16-bit digital engine driving a GSM module with a preloaded SIM card. No special activation is necessary. You can choose any SIM module from a service provider to run this phone. The design accommodates the use of the typical GSM module working with standard attention (AT) commands and such a module having the GSM forum certification and homologated to the GSM carrier network.

The antenna is a high-gain concealed antenna that does not stick out of the telephone. A distinct advantage of this design is that it looks like a key telephone, so you could have a dummy RJ11 socket for an intruder to think it's a phone that's not plugged into the wall socket, and at the same time it wouldn't give the slightest hint that it's wireless.

Although it's not incorporated here, you could include a small security system that detects when doors and windows are opened, and then sends an SMS alert about a possible

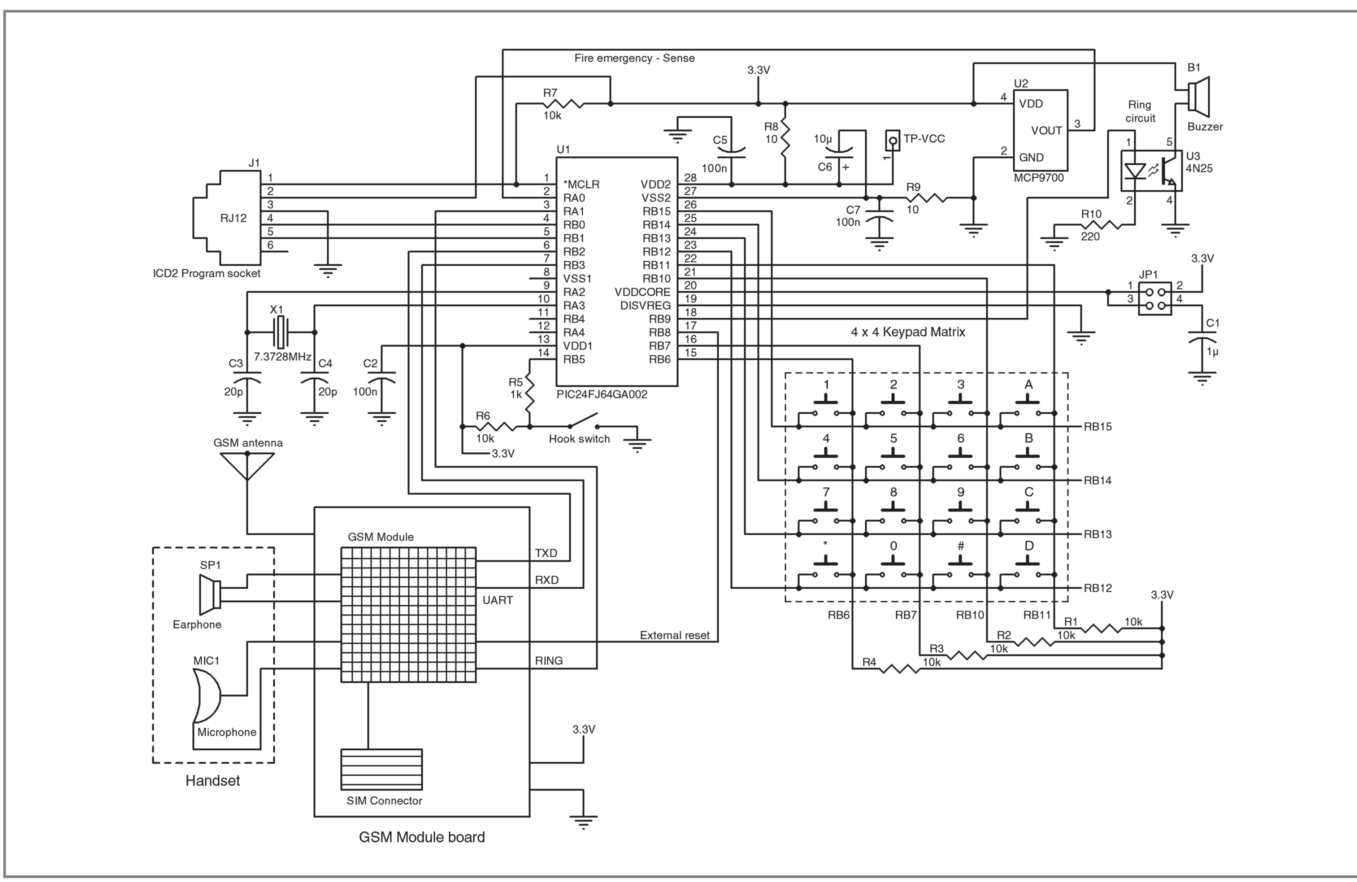


Figure 2—The complete circuitry

intrusion. You could activate this alarm from your GSM mobile through SMS or DTMF signaling over a voice channel. To build a security system, such as an intruder alert system, you'd need door/window sensors that aren't featured in this design. Activation of the alarm system over SMS is possible through control characters that could be passed by the GSM to the MCU UART. Such characters would be parsed to obtain the control code. DTMF would require an additional decoder (e.g., Mitel MT8870) over the voice channel. I already mentioned that this feature is not incorporated in the design, and it's a design that would include an intruder alert system (one that would dial 911 during an intrusion). The idea is to have your phone perform functions ranging from autodialing to building monitoring to appliance control. It can be a communication gateway with added security features. Now, here's how to design this "gadgetoid" with EI.

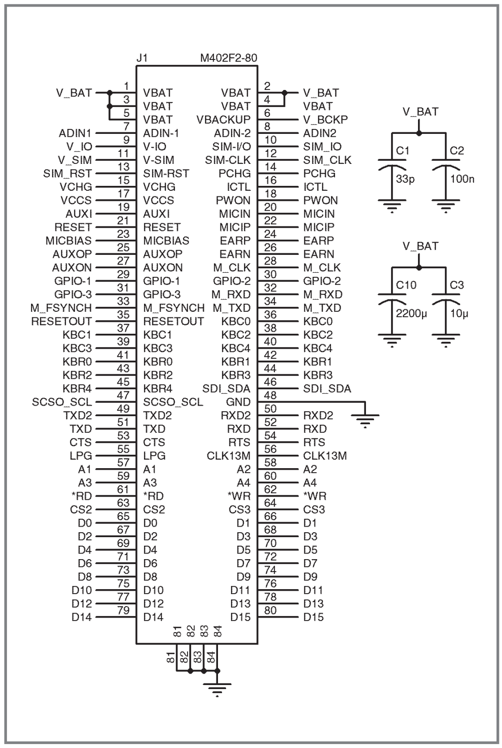


Figure 3—The GSM board, breakout header pinout diagram

CONTROL, POWER, & MONITOR

Refer back to Figure 1. The first module is the GSM radio, interfaced to the PIC24FJ64GA002 microcontroller with a keypad and a temperature sensor interfaced to the microcontroller. That's all.

I assembled the board with a Microchip Technology PIC starter board reference design (see Figure 2). The PIC24FJ64GA002 uses a 28-pin base and the GPIOs are available on the I/O header (see Table 2). The 7.3728-MHz crystal is intended to achieve a 0% error on the UART at the data rate of 115.2 kbps required for this application. The programming port is available on the default PGC/PGD pins on pins 4 and 5, and MCLR is available on pin 1. The six-pin RJ12 port for programming the PIC has the PGC, PGD, MCLR, and GND terminated at this port that interfaces with a Microchip Technology MPLAB ICD 2 debugger. The power supply to the PIC board is 3.3 VDC through the DC connector onboard. The 3.3 VDC is generated from the LDOs on the GSM module board.

The GSM module board runs from a 5-V source with the LDO onboard generating 3.3 VDC for the module and the SIM card (see Figure 3). The same supply of 3.3 VDC is routed to the PIC24FJ64GA002 section. The RB8 port pin from the PIC24FJ64GA002 microcontroller is used to reset (active low) the GSM module after power-up. The GSM audio I/O port is interfaced to the handset and an internal 3-dBi whip antenna is used. The AT command interface to this module with the PIC24FJ64GA002 is through UART0 pins, as mapped on the microcontroller.

The 4 × 4 keypad has rows as inputs (pulled high) on port pins RB6/7/10/11 and columns as drive outputs on port pins RB12/13/14/15. The keypad subroutine loops on making each column low and scanning the rows for a switch press that drives that particular pin low. The state is read with a delay subroutine to debounce the key press and the dialed digit information is sent as an ASCII character over the UART to the GSM module. The keypad scan routine is a standard routine that's explained in the source code with comments. Refer to the code posted on the *Circuit Cellar* FTP site.

The MCP9700 voltage output temperature sensor is used to determine fire alert conditions from a preset temperature value of 60°C and higher. It supports a wide temperature range of –40° to 125°C with an operating voltage of 2.3 to 5.5 V with a 10-mV/°C output. The sensor is biased from the VDD of 3.3 V, which serves as the maximum voltage value and is interfaced to AN0 input on the PIC24FJ64GA002. The firmware scans for the ADC value, compares it with a preset hex value corresponding to 60°C, and sends an SMS alert for all values of 60°C and above.

The PIC24FJ64GA002, the GSM module, and the SIM operate from a 3.3-VDC source (see Figure 4). The GSM module board accepts a 5-VDC input and generates 3.3 VDC, which is used for the GSM module, the SIM, and is routed to the PIC board to power the PIC24FJ64GA002 (see Figure 5).

CODE & PROGRAMMING

I used the MPLAB ICD 2 debugger and programmer to flash/debug the PIC board (see Photo 2). The PIC board has PGC/PGD pins available for the same thing. Programming

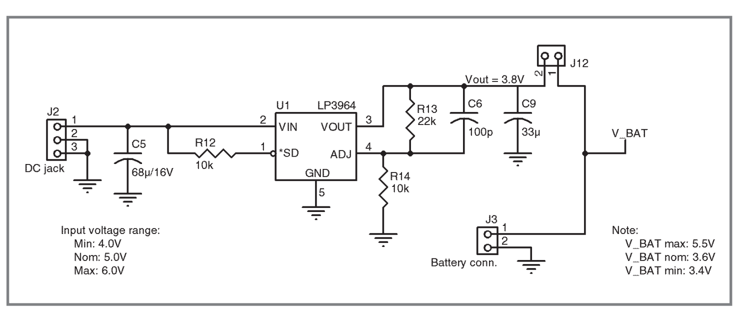


Figure 4—Board power supply

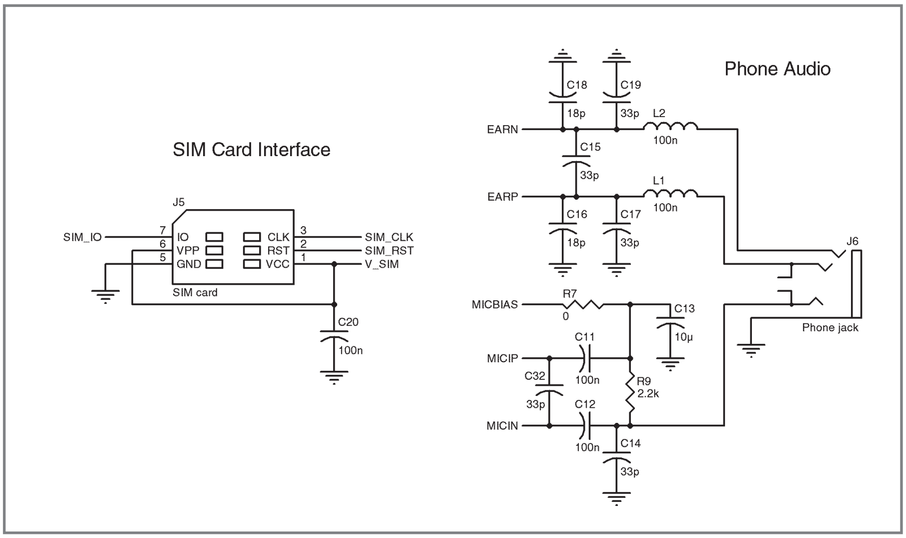


Figure 5—The SIM card and handset connection diagram

is straightforward and simple. Plug the debugger into a PC's USB port, plug the RJ12 into the RJ12 port on the application power, and apply power through the 5-VDC adapter to the PIC board. Next, open the MPLAB IDE v7.62, select ICD 2 under the Debug option, and the application board is instantly detected. Write the code, build flash, and run the same.

The Assembly language code for this application was written with the MPLAB IDE v7.62 (all higher versions are backward-compatible). Lower versions have a bug for the PIC24FJ64GA002 part and only versions v7.62 or higher are recommended to build the code. (Refer to the PIC24FJ64GA002 instruction set and the code listing in the project software file on the *Circuit Cellar* FTP site.)

The code is relatively simple (see Figure 6). The main loop initializes the microcontroller, sets up UART and input/output port pins on port RAx and RBx, and initializes the ADC function for fire alert sensing through AN0 (temperature sensor). `GSM_RESET` initializes an iWOW Technology TR-800 GSM/GPRS module. This block is optional and may be deleted for GSM modules that don't need a start-up reset.

SMS has the subroutines `SMS` and `Receive`. It scans RA1 (GSM ring indicator) and RB11 (keypad buttons "A" and "B") for an incoming call and performs call receive or state alerts by sending a precanned SMS.

To dial any number, the AT commands are used (sent over the UART to the GSM module). The subroutine is called while dialing a number or sending an SMS.

`SCAN123A` is the keypad scan subroutine preceded by the `Initiate` subroutine. If an off-hook state is detected, an `ATD` command is sent to the GSM module followed by a `Goto` instruction to jump to the subroutines. The subroutine scans the keypad and dialing entries sent to the GSM module over the UART.

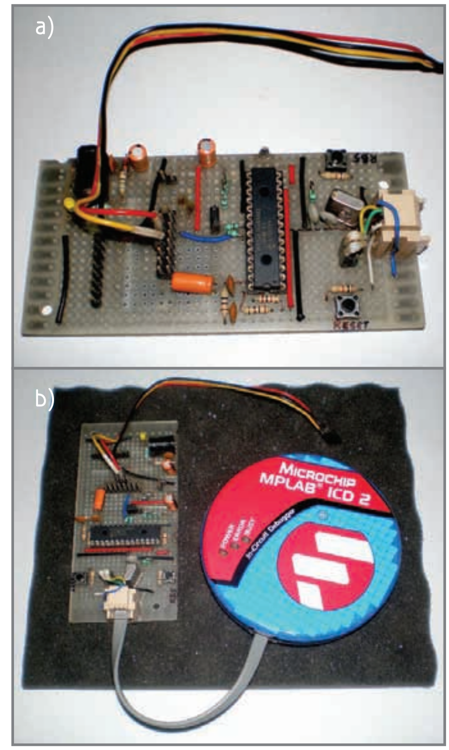
`Dial` checks the hook switch status and initiates a call; `Endcall` transmits `ATH` to the GSM to end the call. `Check` checks the hook switch status after every number is keyed. `Transmit` sends data to U1TXREG.

The precanned SMS messages are sent during an alert condition. Here is an example:

```
AT+CMGS="<Number to be
dialed>" <Enter>
<Message>
CTRL+Z
```

INTEGRATION & TESTING

I built the phone system on an existing TR-800 GSM evaluation board with a SIM cardholder and the power supply circuitry, antenna port, Photo 2a—This is a Microchip Technology PIC24FJ64GA002 microcontroller board. b—This is the Microchip Technology MPLAB ICD 2 debugger and programmer that I used for the project.



and other peripherals required to operate the GSM module. However, any standard GSM module (it must have an IMEI number and have a homologation approval from the GSM service providers in the country of use) may be used with minor changes if required. AT commands for dialing/SMS are the same and the modules are compatible with this design to operate in voice call/SMS mode.

Photo 1b shows the system rigged up in the simple push button telephone mold. The results of my dialing and emergency SMS tests were satisfactory. The call/receive function worked well. There was no appreciable difference in GSM signal quality with the onboard stealth antenna used, and it worked error-free like a fixed cell phone terminal. The handset had standard voice quality and I didn't have to increase audio volume, although it may be achieved by using the standard AT commands.

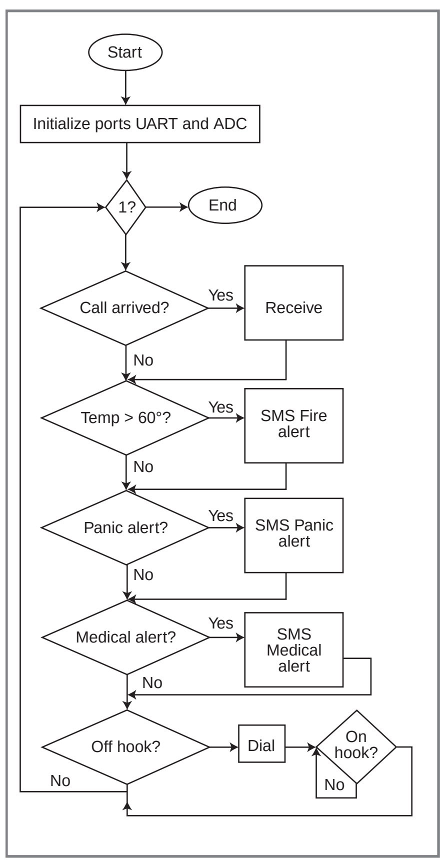


Figure 6—The program flow chart developed using iWOW Technology's "AT Commands Guide." (Refer to the Resources section at the end of the article.)

I built the emergency SMS numbers into the program. In this design, I didn't keep the provision to input numbers dynamically from a keyboard. Many features can be added by integrating a 1 × 16 or 2 × 16 LCD. The current system is fairly basic because it's intended to be affordable yet effective.

HANGING UP

I was motivated to design my phone primarily for geriatric use where elderly people are left to fend for themselves. The idea was to create a simple, cost-effective wireless phone that has the added features of emergency dialing, a fire alert, and other such services. And now I have a platform for various EI companion gadgets.

The possibilities for this platform are endless. I'm confident you could easily add building surveillance and monitoring features. Perhaps I'll read about your design in a future issue of *Circuit Cellar*. ■

Indranil Majumdar (majumdar@ieee.org) is an electronic communications engineer who has been working on RF and wireless systems for the last 20 years. He has worked on embedded designs since 1998 and has been extensively involved in the design of wireless embedded systems for vehicle tracking (GPS-GSM/GPRS), GSM POS terminals, smart energy meters using GSM and 2.4-GHz LPR, and 2.4-GHz ZigBee wireless systems for industrial and home automation. Indranil is an amateur astronomer and an avid ham radio operator (VU2KFR, licensed since 1984). He is also a member of the American Radio Relay League and IEEE (both U.S.).

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar/pub/Circuit_Cellar/2011/252.

RESOURCES

V. Garg and J. Wilkes, *Principles and Applications of GSM*, Prentice Hall, 1999.

iWOW Technology, "TR800 AT Commands Guide," 2006, www.iwow.com.sg/images/TR-800%20GSMGPRS%20Module(081007).pdf.

Microchip Technology, Inc., "MPLAB, Integrated Development Environment User's Guide," ww1.microchip.com/downloads/en/devicedoc/51519a.pdf.

ZigBee Alliance, ZigBee Specifications, www.zigbee.org.

SOURCES

TR-800 GSM/GPRS Module
iWOW Technology | www.iwow.com

PIC24FJ64GA002 Microcontroller, MCP9700 Thermistor IC, and MPLAB ICD 2 Debugger
Microchip Technology, Inc. | www.microchip.com

MT8870 DTMF Receiver
Mitel Networks Corp. | www.mitel.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:

—

GPS-GSM Mobile Navigator
by Ma Chao & Lin Ming
***Circuit Cellar* 151, 2003**

What's the more laudable engineering feat, designing a navigation system capable of tracking ships in Shanghai Port or placing at the top of a competitive design contest? With the award-winning GPS-GSM Mobile Navigator, Ma and Lin accomplished both. Topics: GSM, Cellular, Mobile, GPS, Navigation, Communications, Transmission, SMS

—

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

by Marty McLeod (USA)

Li-ion Battery System Solution

Implement a BCU and Safety-Certified RTOS

Current lithium-ion (li-ion) systems can be problematic. Here you learn how to implement a battery control unit (BCU) with a safety-certified RTOS. The design enables battery monitoring, data reporting, and circuit protection.

While working for a provider of large-capacity lithium-ion (li-ion) battery cells and battery systems, I became familiar with the major requirements for the maintenance, charging, and circuit protection used in the design of such energy systems. Along the way, I discovered several areas in the designs that needed improvement. In this article I'll describe some of my findings, as well as a useful battery module project that you can build at your own workbench.

My motivation for this project was to find solutions to several design issues found in typical li-ion battery systems. I designed an eight-cell, 32-V nominal voltage li-ion battery pack prototype complete with battery management electronics (see Photo 1).

Photo 1—This is a working li-ion battery cell module with simulated battery cells, proudly powered by the SafeRTOS system, thanks to the Texas Instruments Stellaris LM3S9B96 microcontroller evaluation kit.

STORAGE SYSTEMS

Large capacity energy storage systems are finding increased usage in applications such as electric/hybrid vehicles, solar and off-peak energy storage, and lab and development testing for replacing older energy sources and lower energy-density batteries. Most, if not all, li-ion battery systems share some common features including circuit protection (typically a circuit contactor), real-time cell voltage monitoring and cell balancing, and operational status data transmission to the user/outside world.

Some systems can be quite expensive. Some are also based on proprietary hardware or less-popular microcontrollers, and upgrades can be extremely difficult, if at all possible. Some systems have moderate software performance, at best, and may suffer from software response latency. They do not typically operate with a safety-guarantee or high-reliability real time operating system (RTOS), and they may not provide a fail-safe condition in the loss of external power. Some of the systems are inflexible and short on general-purpose input/outputs (GPIO). Others rely heavily on external integrated circuits (ICs) on the battery control unit (BCU) board, sometimes for basic functionality rather than implementing those in software.

To clarify a potential safety scenario, let's consider one of the weaknesses just mentioned. Figure 1 shows a comparison of an actual battery management system in use and my proposed alternative design. Figure 1a is a system powered by an external supply, consuming energy 100% of the time due to the contactor design (using a normally closed con-

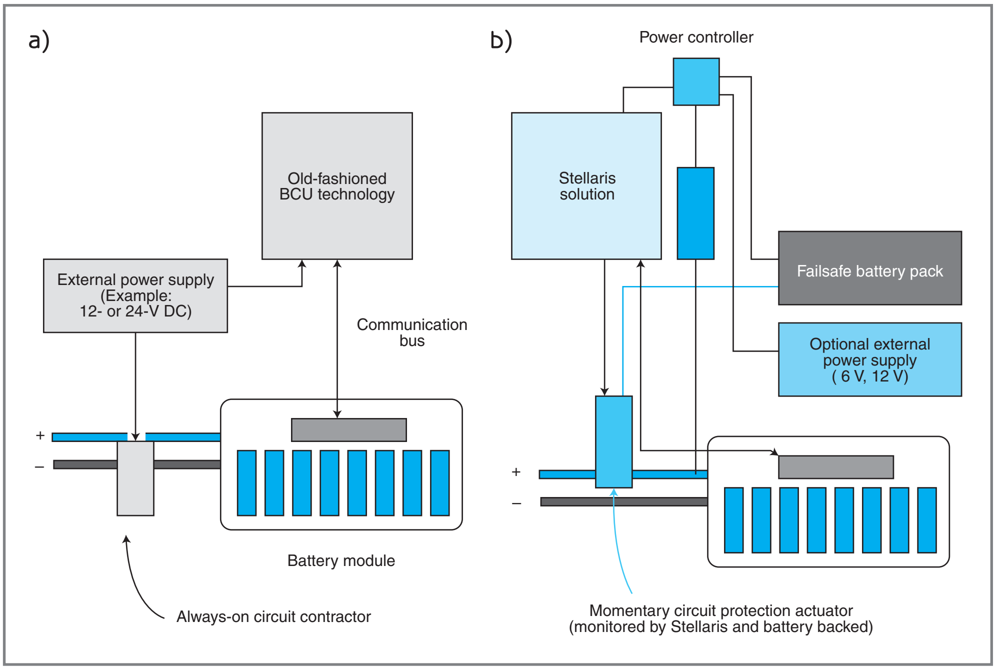


Figure 1—A conventional design versus the Stellaris-based approach. I encourage you to consider the results in a failure in the first system. How will it handle an emergency shutdown?

tactor). Figure 1b is an alternative in which the system can ensure safe operation regardless of power loss and the contactor is powered only when needed.

As you can see in Figure 1, if the conventional system loses its external power source, the contactor may open, but the entire system will lose all operation and not fail gracefully (safely)! To make matters worse, status data won't be reported to the outside world if the system shuts down. Therefore, I came up with my own approach to the design and decided to create a working proof of concept.

DESIGN CRITERIA

I built a Texas Instruments Stellaris LM3S9B96-based working battery module. (The microcontroller was part of an EKK-LM3S9B96 evaluation kit from the Texas Instruments DesignStellaris 2010 design challenge.) I designed the software design using the SafeRTOS in the ROM on the Stellaris microcontroller. Li-ion cells could be replaced by safer or low-cost miniature cells, but the voltage range must be identical in order to provide real-world voltage operating conditions.

My design provides precise battery cell monitoring and cell balancing under control of the BCU, and it accurately measures battery module current (which must be bipolar) and module temperature. The module prototype closely mimics a real battery module. It provides three data reporting methods: controller area networking (CAN), RS-232, and a custom LCD monitor. A circuit protection module provides an efficient means of operation. Power is consumed *only* when actuator operation is needed. The BCU detects contactor operation and circuit protection is guaranteed even in the event of external power loss or inability of the battery module cells to provide a sufficient voltage (if applicable). In case of power loss from the battery module or external power, the system can safely shut down and guarantee the contactor will open. Lastly, the software provides adequate timing, minimized

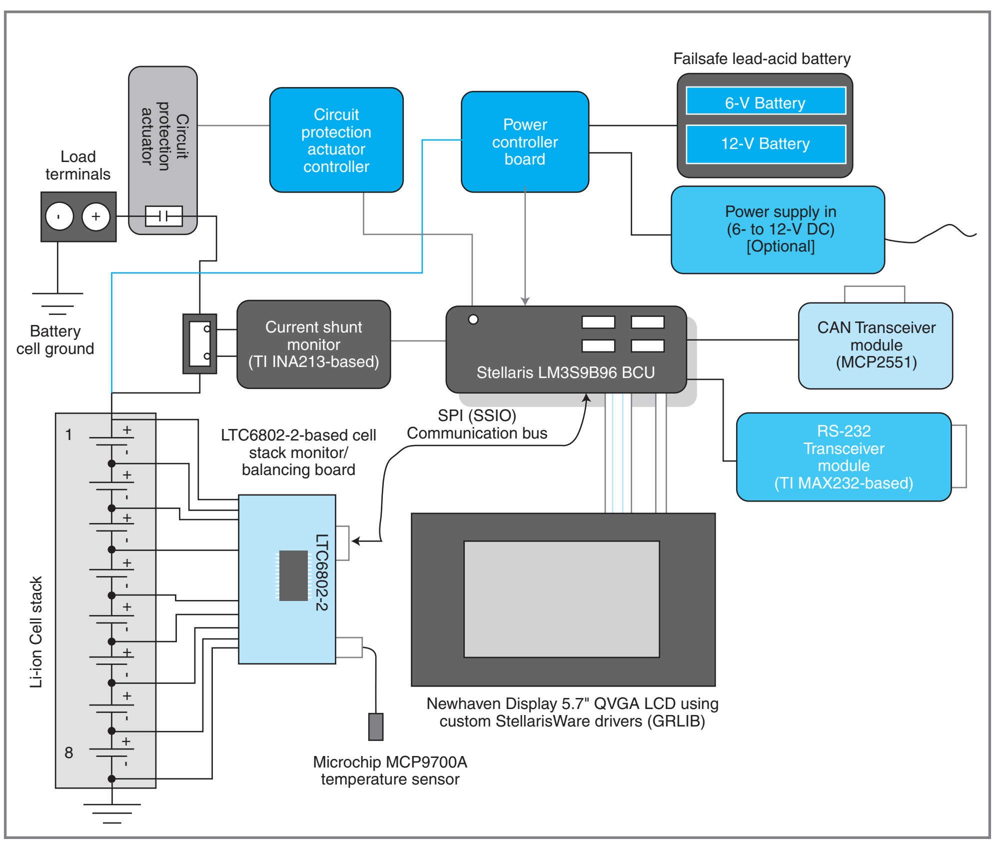


Figure 2—My design criteria resulted in the battery module system design shown here. Quite a few hardware subsystems ("modules") are required to support the operation functions listed in the criteria requirements.

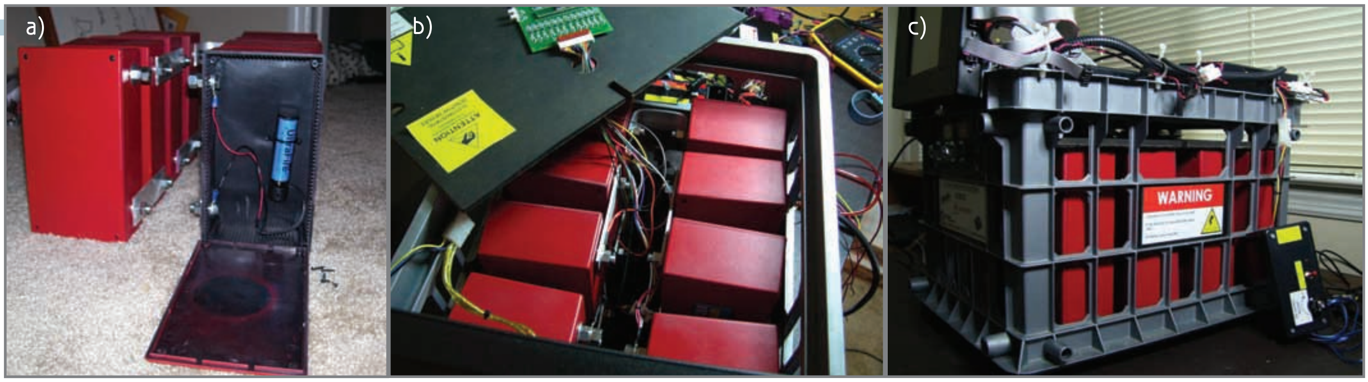


Photo 2a—A simulated large-capacity li-ion cell using a hidden live 18650 cell. The 3/8" bolts, washers, and nuts used to form battery cell terminals are shown. The "cell" assembly approximates the size and design of a larger single-cell high-capacity li-ion battery. b—This is the module cell housing with bus bars and a cell stack board harness installed. c—This is a side view of the battery module housing (a storage crate) with custom-made warning labels and vertical side-mounted simulated cells.

latency for operating external hardware, and uses the RTOS functional approach.

HARDWARE DESIGN

Figure 2 shows the overall system. For the module housing, I used a hanging folder storage crate from an office supply store. This affordable housing provides the appearance of a commercial battery module assembly.

Photo 2 shows the battery cell stack with simulated large-capacity cells that use much safer (and much less expensive) 18650 3.7-V nominal 2,500-mAh li-ion cells. Each simulated large-capacity cell consists of an ABS plastic box housing an 18650 cell along with 3/8" bolts and washers to serve as cell terminal posts. The result is a battery cell modeled after the approximate size of a large-capacity li-ion cell. My options seemed limited in simulating the physical size of a 150-Ah cell, so this idea seemed like a simple but effective solution.

Bus bars were created using custom-cut aluminum strips. The cell stack monitor board wiring harness is terminated with 3/8" ring terminals. A 2-A current shunt was mounted inline with the positive load terminal wire.

Photo 2c shows the module early on, complete with custom-made warning labels. Remember that li-ion battery modules are often used in multiple series-connected groups, achieving high-voltage output (easily 300 V and higher), and can present a lethal shock hazard.

The supporting hardware that was designed and built into the system included a cell stack monitoring and balancing board built around a Linear Technology LTC6802-2 IC battery stack monitor. I had originally planned to use the 10-bit resolution ADC peripherals in an LM3S9B96 microcontroller. However, typical li-ion cell voltage ranges (2.175 to approximately 4.2 V, measured within a 0 to 6 V maximum range in our case) would provide an accuracy of only 0.00586 V per bit (i.e., 6 V per 1,024 bits), which would be too low for the design.

My design needed measurement and balancing windows in the range of 10 to 20 mV. For optimal cell balancing, a better cell voltage ADC resolution was needed. Additionally, utilizing all of the ADC channels in the Stellaris microcontroller would reduce the number of GPIO pins, which rapidly became less frequent once the system design was in place. Therefore, I decided to design a custom li-ion battery cell monitoring and balancing board based on a Linear Technology LTC6802-2 monitor. The LTC6802-2 provides up to 12 cell voltage measurement channels with 12-bit ADC resolution, integrated cell-balancing output control pins, and a measurement resolution of 1.5 mV per bit. It also provides temperature measurement ADC inputs. The custom-designed, assembled, and debugged board is shown in Photo 3. Next in importance to the Stellaris BCU, this is the "heart" of the design, where the Stellaris BCU is the "brain." Note that I designed in a dual inline package (DIP) through-hole mounting approach for the SOP44 IC package, anticipating "worst-case" hardware problems. As the surface mount device (SMD) IC is mounted to an SOP44-DIP adapter, it can be replaced relatively easily should a repair be needed.

The cell-balancing board is a serial peripheral interface (SPI) communication slave, addressable from 0x00 to 0x0F in hexadecimal. The module design enables an expandable

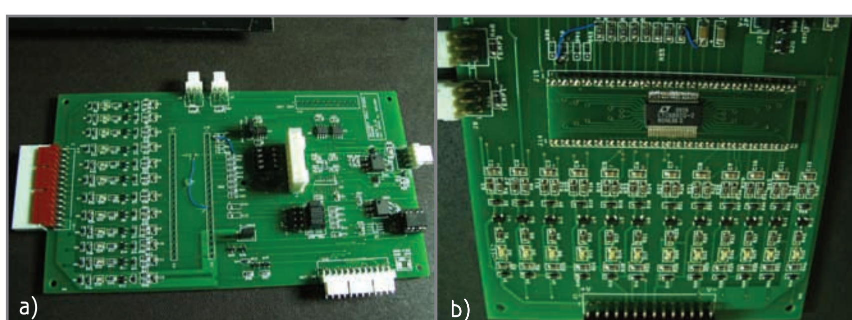


Photo 3a—The assembled battery cell stack monitor and balancing board. b—The board has a Linear Technology LTC6802-2 li-ion cell IC. Note the cell sensing and balancing circuitry with LEDs.

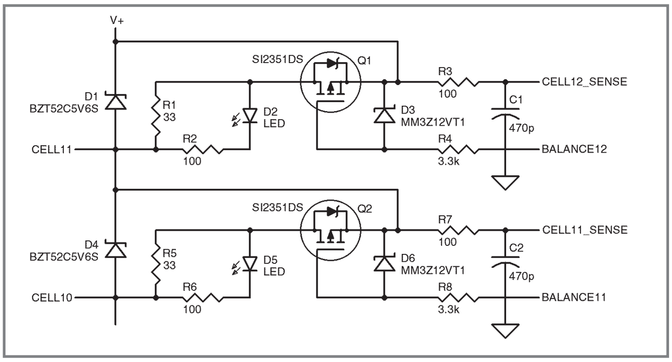


Figure 3—A portion of the cell-balancing circuitry (driven by the LTC6802-2 upon commands from the BCU)

system, up to a total of 16 slaves, with minor modifications to the software code without requiring any additional BCU boards. The LTC6802-2 differs from the LTC6802-1 in that it uses a parallel addressable SPI as opposed to the daisy-chained interface of the latter model.

Currently, the design communicates at 250 Kbps and the Stellaris BCU acts as a master. The LTC6802-2 must receive a message command periodically or, after 2.5 s of inactivity, the built-in watchdog will be triggered, the chip will enter low-power mode, and configuration register settings will be lost.

The central SafeRTOS software task, `LTC6802_MODULE_task`, which drives the cell stack board as a SPI "slave," never idles for a long enough period of time to enable this to happen. Time between communication transactions is typically less than 100 ms.

Messages received from the LTC6802-2 are terminated with a packet error code (PEC) as it serves as a more reliable method of error handling than a simple checksum computation. The PEC is implemented in the function `compute_PEC_CRC`, and is decoded and checked by the BCU in software (see Figure 3).

The system hardware also includes a current shunt monitor board. For both charging and discharging cycles, you must monitor not only positive (discharging) current, but also negative (charging) current produced or received by the cells. Ideally, the solution shouldn't require the overhead of generating a negative voltage for the current shunt monitor IC.

My solution was a Texas Instruments INA213 current shunt monitor, which is reasonably priced (less than $4) and well-documented. While it has a built-in gain of 50, I used external 0.1% precision resistors to reduce the gain to a value of 20. My reasoning was that the ADC inputs on the microcontroller are limited to 3 V using the internal voltage reference. To avoid negative voltages, for zero-current computational purposes, I used a DC bias of half the maximum allowed input voltage (1.5 V serves as the 0 current reference voltage). Using the selected current shunt (limited to 2 A for our scaled-down battery cell purposes), the gain needed can provide, at maximum current, an output of 1 V (i.e., 20 × 0.050 V). This must be added to the top of the DC bias value, for a maximum of 2.5 V at maximum positive current, which fulfills our ADC input limit of 3 V.

My system also features a CAN transceiver and RS-232 breakout boards. Photo 4 shows the two custom designed and built communication boards that were intended to be used for data reporting by the battery module. The CAN transceiver board can connect to the Stellaris BCU in a similar fashion to the RS-232 breakout board. Both provide a means of transmitting and receiving data when powered by a 5-V supply. In the case of the RS-232 board, the MAX232-based design was hand built due to design time constraints. The CAN breakout board provides both a DE9 pinout connector and an internal 0.1" header for custom wiring. The transceiver IC is a Microchip Technology MCP2551 DIP-8 package CAN transceiver.

I also designed a circuit protection actuator motor driver. It provides optically isolated control inputs for the BCU and powers the circuit protection actuator momentarily for minimal power consumption, thereby minimizing contactor power consumption. The design utilizes an automotive power door lock actuator to serve as a strong solenoid motor, moving a custom mechanical "contactor" assembly with auxiliary contacts. The contactor is modeled using a double-pole, double-throw (DPDT) On/Off toggle switch connected using custom linkage. A 12-V battery (or an external 12-V power, when present) provides power for driving the actuator motor.

The actuator controller/driver board drives the solenoid using a two-wire, reverse-polarity output. The BCU simply sets one input pin high for "open" or the other low for "closed" using momentary inputs. A second switch circuit is wired to provide auxiliary contacts to enable the BCU to monitor the current contactor position at any given time. Note that when at rest, the solenoid sees a negative (0 V) polarity on both power wires.

Inputs are isolated to prevent any voltage spikes from the collapsing magnetic field of the solenoid's coil from destroying the Stellaris BCU microcontroller. Additionally, inputs are routed to a manually controllable contactor push button box on the left side of the module.

The hardware design includes a battery-backed power

Photo 4—The CAN and RS-232 custom breakout transceiver boards. Both were designed, built, and debugged. (They were not implemented before the DesignStellaris challenge deadline submission date.)

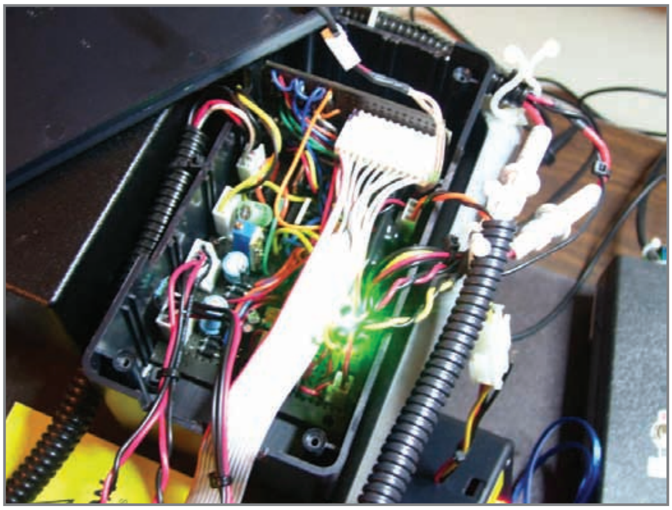


Photo 5—The custom-designed power controller board, was hand-built to guarantee emergency back-up power to the BCU the circuit protection contactor, and the LCD. Note the BCU interface cable carrying both BCU control signals and back-up power-sensing voltages for the ADC channels. This enables the Stellaris BCU to monitor power conditions for the back-up batteries and external supply, if present, in real time.

controller board and battery back-up module. When in a sufficient state of charge, the battery cell stack can power the BCU from a step-down converter (typically a cell stack total voltage range of 22 to 32 V). When needed, an external power supply will be able to override all other power sources to avoid consumption of battery cell stack power or back-up battery power.

If the cell stack is unable to power the BCU (normally due to one or more fully discharged cells), or if the external power cannot be used or fails, the BCU will be able to open the circuit protection contactor, notify you of the shutdown condition, and turn off its own power source until you reset it for the intention of remedying the fault. The power controller uses diode logic and interfaces with the BCU for the active-high enabling of the various power sources, including backlighting and power to the LCD (see Photo 5).

When in an "off" or "shutdown" state, the board consumes no additional current from the back-up batteries or external supply, if present. The design currently uses a small 12-V lead-acid battery for solenoid and LCD backlighting power. A 6-V, 1.2-Ah lead-acid back-up battery is for the BCU, accessory, and LCD supplies (see Photo 6). These are easy to find and reasonably priced.

Voltage-sensing ADC inputs to the BCU from the board provide a 1:4 scaled sensing input set, which is continuously monitored by the BCU to ensure the back-up batteries are in good health to detect a missing external power supply. A scaled voltage input is easy to design. Essentially, the board enables the module to switch seamlessly between the two primary power sources and the third source, the back-up batteries, to ensure a safe shutdown of the module system, if required. Using the same active-low Darlington octal driver IC on the board, the BCU also drives an audible buzzer to indicate start-up, alarm, and shutdown warnings.

The hardware includes an LCD assembly. A Newhaven Display 5.7" QVGA (320 × 240 pixel resolution) model NHD-5.7-320240WFB-CTXI#-T-1 color display monitor is used—via the StellarisWare graphics library and custom display drivers—to provide real-time display data for module parameters such as cell voltage, current, and temperature. The design features custom plastic housing (Serpac). The LCD with a 20-position 0.1" male shrouded header, cables, and another header for LCD backlighting and signals to the touchscreen controller board are mounted inside.

The BCU interfaces to the LCD using a ribbon cable for an 8-bit-wide data bus along with several other control lines (e.g., read, write, and data/control). The data width is fixed by Newhaven due to its design and receives 18-bit color data in the form of 6-6-6-bit data writes (6 bits per byte are used, the least significant bits are not used). Aside from the most basic initialization functions, hardware-level drivers are not supplied by Newhaven, so they had to be created as well.

A manual push button box enables you to open or close the contactor only when it's safe (only when enabled by the BCU). As the contactor position is monitored by the BCU, if you attempt to close the contacts during an unacceptable operating condition, the BCU will reopen the contacts or simply disable the logic circuitry power in the actuator controller board, thereby rendering the contactor unusable until alarm-free operating status returns.

The supporting hardware also includes the BCU assembly, which is housed in an enclosure containing both the JTAG debugger board and the Stellaris LM3S9B96 evaluation board. Note that the BCU can be powered by the battery module. It does not require connection to a host development or other computer for

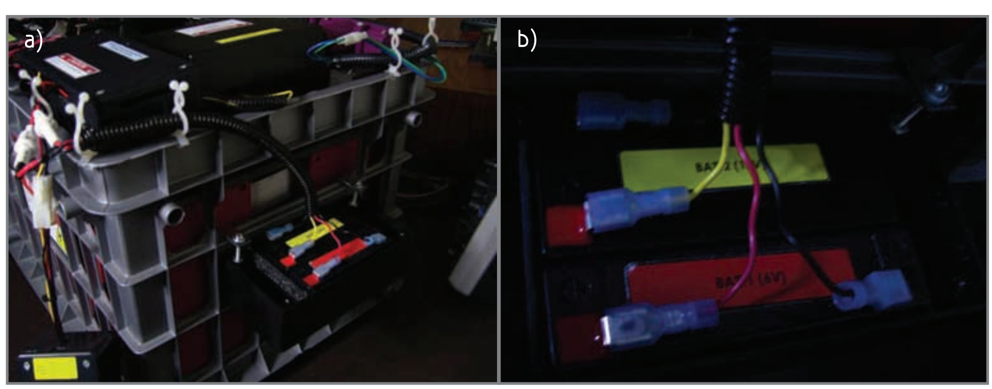


Photo 6—The back-up battery assembly is located in the rear of the module (**a** and **b**).

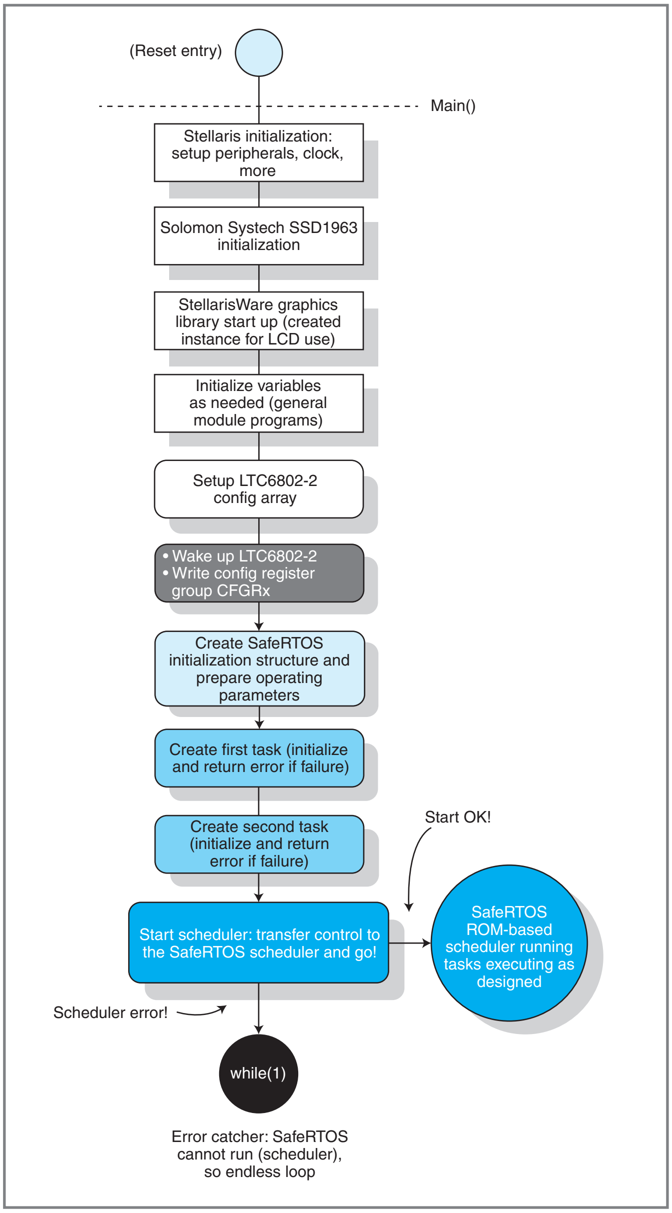


Figure 4—The software design architecture start-up code flow. This illustrates the initial steps executed before starting the RTOS, then the RTOS start up itself.

power. The power controller board provides a dedicated port for the BCU to receive 4.25- to 5-V power via the USB connector.

SOFTWARE DESIGN & ARCHITECTURE

The software design portion of the project was developed using ARM's RealView integrated development environment (IDE). The software was implemented through the development of fundamental procedural programming on a modular function basis, testing each function (e.g., write drivers for the LCD, LTC6802-2 SPI board, etc.). When developing the software, I focused on the essential functions first and completion of any modification second. Any "icing on the cake" items (e.g., nonessential, those that would improve appearance, and appeal) remained on the back burner until time allowed.

In addition, I planned to build and test a running battery module prototype with debugging, as needed. And, I also planned to add any obviously needed features that provide functionality that were not previously considered or found to be missing during the design process.

The logical steps in the process were to begin by converting the original program flow into a SafeRTOS-based format. If this went successfully, with adequate free time remaining, I planned to "clean up" and optimize the SafeRTOS-implemented software program, further divide the system into more tasks, and adjust the SafeRTOS priority levels and scheduler calls accordingly. Although I ran out of time to implement additional features, I completed the primary goal of writing the essential critical drivers and modular code, and I had the software successfully running within the SafeRTOS system. Due to limited time and resources, I decided against attempting to complete the CAN, RS-232, and enhanced graphics/touchscreen software portions.

Figure 4 shows the basic elements executed when the program begins. Obviously, the software program must complete a number of initialization and other "housekeeping" items that are essential to the battery module design before any RTOS elements come into play.

Note that a failure to properly start the SafeRTOS scheduler means a total program failure: the system cannot execute as intended and will go into an endless loop. In the original procedural design, the program itself was just an endless loop executing modular functions.

Once initialization tasks are complete, you create (initialize and provide the starting structures for) your RTOS tasks, which are the heart and soul of the battery module software. SafeRTOS actually executes the tasks (which *literally* carry out software tasks when running) according to task setup and priorities.

BATTERY MODULE TASKS

The battery module design runs with two RTOS tasks

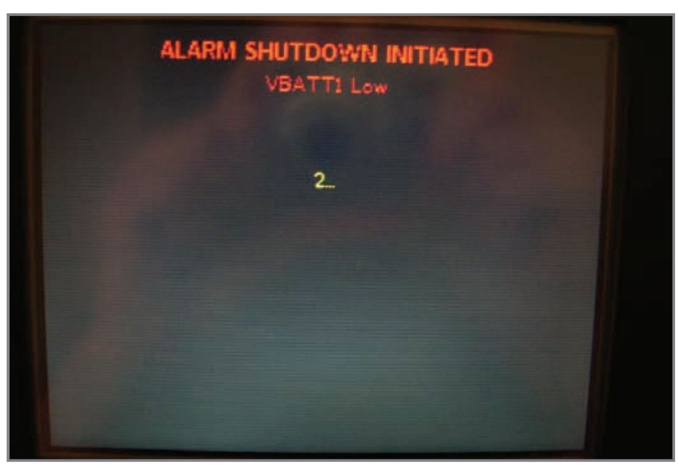


Photo 7—`MODULE_alarm_handler()` is continuously called during operation. When a power fault ("alarm") occurs, the BCU executes a safe shutdown process. The module will not enable unsafe operating conditions to occur. VBATT1 (battery backup 1) is sensed as failing. This triggers the shutdown sequence, which triggers the circuit protection counter.

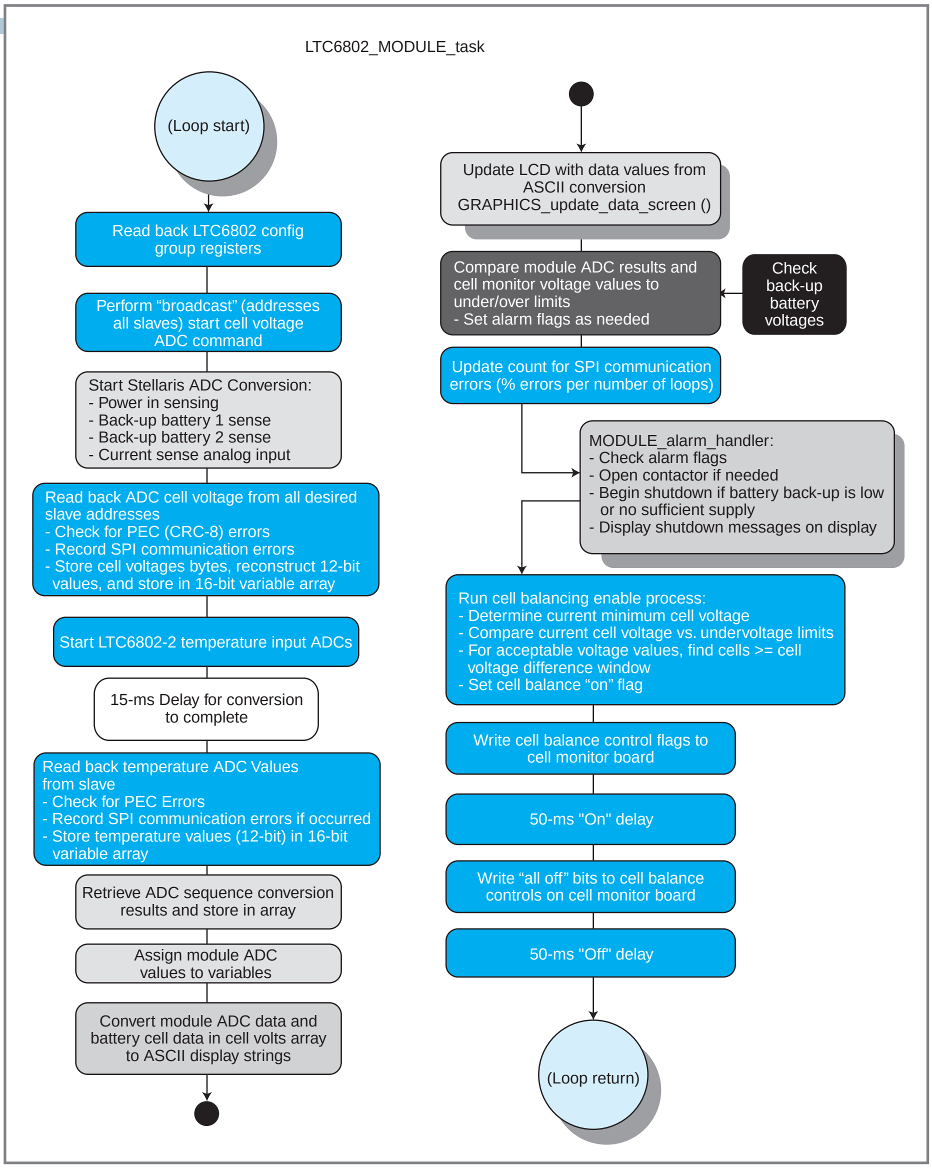


Figure 5—The central task's functional operation. This task executes both battery cell monitor board and basic module functions. Originating as a procedural-based design, the execution constantly loops and performs a series of sequential functions to measure cell voltages, update user information, balance cells if needed, and check the current operational conditions.

a continuous loop is used and SPI communication is constantly active to keep the LTC6802-2 awake.

The task continuously monitors battery cell parameters, temperature, current, and the contactor position status and reacts accordingly. The LCD screen is continuously updated as this is done. Photo 7 illustrates the shutdown sequence user display that is executed when a back-up battery failure is detected with or without external power applied.

In an RTOS, each task consists of a never-ending loop. In my case, I never need to halt execution of the endless loop by blocking or waiting on a queue event, but simply enable the task to run forever.

The `LED_task` executes constantly, as I desire, because it shares the same priority as the central task. However, the central module task loses very little execution time when running as the LED task occurs based on a "wait until" basis for a regular blinking interval. Also, the LED blink task is quite short.

The central module task is detailed in Figure 5. This is the sequential functional operation, which is a requirement when using the LTC6802 (in terms of cell balancing, initiating the ADC conversion processes, and more). It is carried out, and then the module operations are carried out when the LTC6802 cell data has been retrieved.

with equal priorities (I didn't have time to break the central task into assorted multiple tasks): the heart of the software programming, the `LTC6802_MODULE_task`, and the `LED_task`, which is a simple task to continuously blink an LED to indicate a valid running state. The `LTC6802_MODULE_task` was given its name as it combines the essential battery module functions along with those specific to the heart of the design, the LTC6802-2 cell monitor board.

The central task is perhaps best understood by looking at its procedural predecessor, which was a `while(1)` loop carried out in `main()` during the initial phase of testing and development. The BCU must not allow the cell monitor board to rest idle for an extended period of time (specified as 2.5 s) as the IC will execute its watchdog function and lose its volatile configuration register values. Hence,

A SUCCESSFUL SOLUTION

This project successfully demonstrates an alternative battery module design solution. Working with the Stellaris microcontroller, supporting software libraries, and application programming interfaces (APIs) was also productive.

I tested the finished module. It shuts down safely during a critical alarm condition such as back-up battery loss, communication loss from the cell stack board, or other conditions. This triggers the opening of the circuit protection contactor (with minimal power use) when the BCU detects an alarm condition.

In hindsight, I would have preferred to implement the CAN, RS-232, and touchscreen software features for which I designed hardware. Unfortunately, numerous problems arose that caused setbacks. These included late

circuit boards (up to two weeks late!), the Stellaris debugger interface in Keil µVision3 frequently crashed, and I only purchased one 18650 battery cell charger, which severely limited my test iteration speed. Therefore, my engineering decisions were based on what I could complete in an acceptable amount of time.

I recommend that you limit the scope of larger-scale projects in order to minimize setback risks. I would also recommend ensuring that no one item in a project will cause all progress to halt should a temporary obstacle occur. When possible, design items in parallel and then return to the problematic item when the issue is resolved. If possible and reasonable, plan a quick assessment of estimated time for a design scope *before* beginning the parts purchasing and actual construction. And allow time for things to go wrong—because they will!

I also recommend approaching PCB design for initial projects in the following fashion. In the first revision of a board design, allow for larger, more easily assembled components and provide additional parts spacing and copper trace availability should modifications be necessary. Don't spend an excessive amount of time trying to make a perfect board the first time. Additionally, use reliable PCB CAD software that you're comfortable with, which will enable expedient work. Lastly, add breakout and test points and headers on draft version circuit boards for test equipment, especially on digital logic lines or busses for digital debugging.

Marty McLeod (martymcleod@yahoo.com) is an embedded software engineer. He holds a degree in Electrical Engineering from Auburn University, as well as an AA in Electronics Technology. Marty enjoys listening to satellite radio and working with hardware/software designs and electronics projects, especially embedded systems.

PROJECT FILES

To download the code and additional photos, go to ftp://ftp:circuitcellar.com/pub/Circuit_Cellar/2011/252.

RESOURCES

Texas Instruments, Inc., DesignStellaris 2010 Design Contest, www.circuitcellar.com/designstellaris2010.

———, StellarisWare libraries, http://focus.ti.com/docs/toolsw/folders/print/spmu020j.html.

SOURCES

GC-123 and GC-612 Batteries
All Electronics Corp. | www.allelectronics.com

RealView Microcontroller development kit and µVision3 IDE debugger
Keil—An ARM Company | www.keil.com

LTC6802-2 Battery stack monitor
Linear Technology Corp. | www.linear.com

MCP2551 Transceiver and MCP9700A Temperature sensor
Microchip Technology, Inc. | www.microchip.com

NHD-5.7-320240WFB-CTXI#-T-1 Controller
Newhaven Display International, Inc. | www.newhavendisplay.com

BP5220A DC/DC Converter
ROHM Co., Ltd. | www.rohm.com

SSD1963 Display controller
Solomon Systech, Ltd. | www.solomon-systech.com

Stellaris LM3S9B96 microcontroller, INA213 current shunt monitor, and SafeRTOS
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:

—

BatMon to the Rescue
A Battery Monitor for RC Applications
by Thomas Black
***Circuit Cellar* 143, 2002**

For years, hobbyists have relied on voltmeters and guesswork to monitor the storage capacity of battery packs for RC models. Now Thomas introduces a more precise high-tech battery monitor that is small enough to be mounted in the cockpit of an RC model helicopter. Topics: Battery, Monitor, NiCd, NiMH, Capacity, LED, PIC16C63, DS2438

—

Portable Power
A Power Supply for Embedded Applications
by Jason Wu, Kiran Kanukurthy, & David Andersen
***Circuit Cellar* 193, 2006**

This team of designers built an inductively charged power supply for embedded applications. The portable system provides 100-mA, 3.3-V continuous power. Topics: Portable Power, LTC1325, Wireless, Inductive Charging, ATmega8, LM2621, Coil, Inductive

—

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

by Daniel Mejía & Jayce Doose (USA)

Passwords Through a Touchpad

A Gesture-Based Security System

Password usage is ubiquitous, which makes password theft and reproduction a big concern. While many password-entry techniques have been created to increase security, there are disadvantages associated with most of them. Using a gesture-based password security system helps to successfully create, store, and match passwords, making them safer and more difficult to reproduce.

Security has been a permanent concern in a variety of environments ranging from access restriction in homes and industrial settings to information security in digital systems. Numeric passwords, fingerprint recognition, and many other techniques have been extensively implemented in the past, but they all have drawbacks. Recently, security has acquired special importance due to the vast amount of digital information and the high value that is frequently attached to it. Thus, we decided to use our design skills to build a security system of our own. Our gesture-based password security system enables you to create, secure, and access important passwords (see Photo 1).

PASSWORD PROTECTION

Passwords are now ubiquitous. You may need to remember different passwords to obtain access to a building, to withdraw money from an ATM, to log into a computer, to check an e-mail account, and so on. The main problem is that the nature of these passwords makes them easily identifiable. A typical password consists of a finite set of alphanumeric characters, and anyone can easily steal it by hearing it, seeing it written, or simply watching while it is being typed. Moreover, without the appropriate encryption, automated systems can iterate through many common combinations and decipher the passwords.

We believe that passwords should not be based on combinations that can be written or pronounced, but should instead be abstract elements that are not easily definable but can still be understood and remembered on a personal basis. With this idea in mind, we propose a new type of password that consists of time-based symbols. Using a touch-sensitive device, a user enters a combination of symbols using his or her finger. The dimensions used to represent the password are the shape of each symbol and the time taken when drawing each of the traces. Using this representation, the user is forced to think of and memorize passwords in a more qualitative manner, thus replication by others becomes a very difficult task. For demonstration purposes, we implemented a prototype that presents the new security approach and exhibits one direct application in access-restriction systems.

Gesture-based systems are becoming increasingly common with the spread of touch-enabled devices. Many new

Photo 1—The password security system prototype consists of a Synaptics touchpad, two push buttons, two LEDs, and a USB connection that provides a simple interface to enter and set passwords.

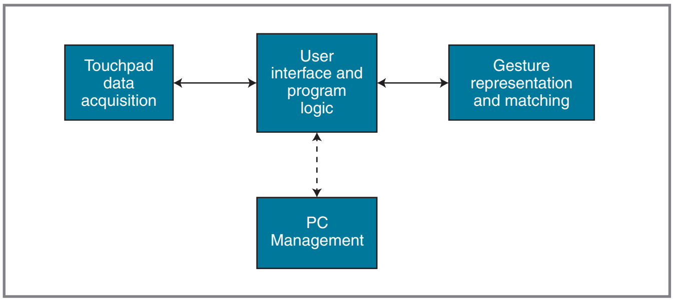


Figure 1—The high-level organization for the device is divided into four self-contained entities: program logic, touchpad data acquisition, gesture representation and matching, and PC management.

and exciting applications are constantly coming out and exhibiting the great potential that gesture systems have in different areas. However, little work has been done in the area of security. A few implementations exist, but they still have pitfalls. The most popular example is the lock-screen gesture password implemented for mobile devices that use the Android operating system. The lock-screen functionality requires the user to enter patterns into a grid using his or her finger. While this approach is innovative and similar to the system we propose, it has a serious problem: after entering the password, the finger leaves marks on the phone screen that are easy to identify and reproduce. Our solution to this problem is the addition of a time dimension with which the recognition of the shapes is not sufficient to copy the password.

As a proof of concept, we implemented a prototype that shows how the new security approach can be easily used in restricted access environments. In this article, we'll describe the different design decisions we made for the device. We'll also outline how the proposed gesture security can be achieved.

HIGH-LEVEL DESIGN

The prototype is simply a black box that receives passwords entered on a touchpad, validates them, and then provides the user with feedback on the accuracy of the password. The box could be put next to a door or any other resource that requires limited access. The feedback signal can be easily used to enable access or produce some other desired effect. Furthermore, we also implemented the administrative capability of accessing the device remotely for managing and monitoring purposes.

In order to obtain the desired capabilities, we designed a system that is divided into four self-contained entities (see Figure 1). The program logic entity is in charge of keeping track of the state of the system, managing the interaction with the user, instantiating the other entities, and controlling the interaction between them. The touchpad data acquisition unit is in charge of retrieving the finger position and pressure data in a raw format and transmitting it to the main program logic. The gesture representation and matching entity has the purpose of converting the raw data obtained from the touchpad into a logic representation and providing the comparison and matching between different gestures. Finally, the PC management entity provides the administrator with an easy tool to manage the device and monitor the system in real time.

SYSTEM SPECIFICATIONS

From a user's point of view, the prototype is a box that contains a touchpad, two push buttons, two LEDs, and a USB connection (see Photo 1). In terms of usability, the most important action is entering passwords, and we wanted it to be intuitive. If the device accepted only single traces as passwords, the user would simply enter the trace in the touchpad and it would be validated. However, for the purpose of adding flexibility and enhancing security, we wanted the device to accept passwords composed of multiple traces. For this reason, the device needs a signal that indicates when the user is done entering the password. Consequently, we introduced what we call the Action button, which is used to signal the start and end of password entry. This kept the task of entering passwords simple: the user presses the Action button, enters one or more symbols in the touchpad, and then presses the Action button again to complete password entry.

Since we added administrative functionalities to the device, we also needed to create a distinction between regular users and administrators. Therefore, we introduced two modes of operation: User mode and Administrator mode. When the device is in User mode, it can only be used to enter a password to try to access the system being protected. When the device is in Administrator mode, it can be used to set the password to a certain combination of traces. For demonstration purposes, we

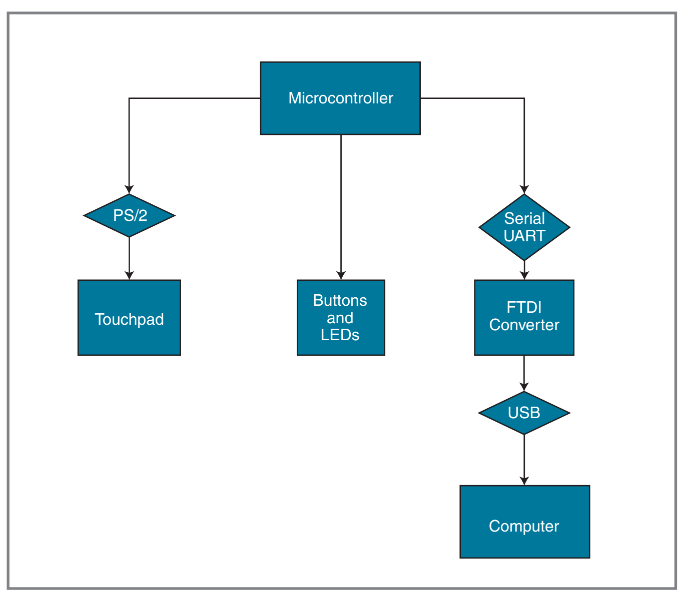


Figure 2—The hardware in the prototype consists of an Atmel ATmega644 microcontroller, a Synaptics touchpad, two LEDs, two push buttons, an FTDI USB converter chip, and a computer. The touchpad communicates with the microcontroller via the PS/2 protocol, and the FTDI chip converts from serial UART to USB protocol to enable communication with the computer.

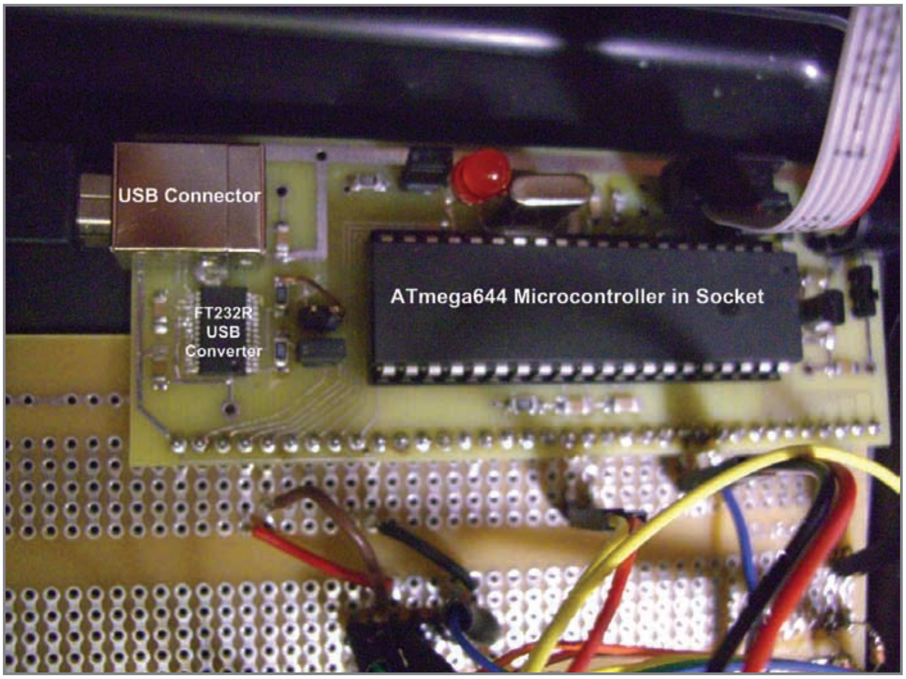


Photo 2—Nathan Chun and Professor Bruce Land at Cornell University gave us the USB prototype board for the ATmega644. It provided all of the hardware necessary to run the microcontroller and interface it via USB to a computer.

used the Administrator button to make this distinction. After pressing this button, you can set the password to any desired value. This button is supposed to be visible only to an administrator. In a real implementation it can be covered, replaced with a key, or simply substituted with a gesture-based administrator password that restricts access to the device.

We used two LEDs to provide feedback on the password validation. If the device is in User mode, after the password is entered, the green LED flashes if the password is accepted, and the red LED flashes if the password is not accepted. If the device is in Administrator mode (i.e., after pressing the Administrator button), the green LED flashes after the password is entered to indicate that the new password was successfully received.

Finally, the box has a USB connection that can be accessed by an administrator. Every device action is constantly sent and can be seen in a computer if the connection is enabled. These actions can be an access request, password approvals and rejections, and hints on the reasons for rejection. When the device is connected to a computer using the USB port, the administrator has the ability to set the password from a computer application and retrieve the approval and rejection counts from the device.

HARDWARE & SOFTWARE

The hardware used in the device consists of a custom PCB built for an Atmel ATmega644 microcontroller that was provided to us by Nathan Chun and professor Bruce Land at Cornell University (see Figure 2 and Photo 2). The board runs on 5 V and a 16-MHz crystal generates the clock. USB communication to a PC is enabled via a Future Technology Devices FT232R USB converter chip, that receives a signal from the microcontroller's universal asynchronous receiver/transmitter (UART). The other hardware includes the touchpad, through which passwords are entered, along with push buttons and LEDs to set access modes and provide feedback.

A Synaptics TM41PUM1311-2 touchpad receives user input. We chose Synaptics because it provided organized documentation for communication with its older touchpads. We used an inexpensive touchpad that had been removed from a Dell laptop, and we determined its PS/2 protocol pinout experimentally (see Figure 3).

To match the high-level structure of the system, we aimed to design organized and self-contained software modules. As a result, we separated the functionalities into independent entities that could be reused in other applications (see Figure 4).

We implemented all of the code for the microcontroller in C, whereas the PC application was implemented using Java. The software organization closely matches the high-level design shown in Figure 1, which was one of our goals in terms of modularity.

PS/2 & TOUCHPAD APIs

The touchpad uses the PS/2 protocol to communicate with a host. For abstraction and modularity purposes, we implemented two separate APIs for PS/2 and the touchpad. The PS/2 API provides only the most basic transmission and reception functions whereas the touchpad API implements touchpad-specific functionalities that use the PS/2 protocol underneath. Some examples of these functionalities are touch information transmission, information queries, and configuration sequences. Furthermore, we implemented a structure that represents the absolute data packet provided by the touchpad and includes a function to parse the received bytes into this representation.

The touchpad API provides the access to the touchpad device. It does not contain the actual touchpad handling, which is in the interface file, because that functionality was specific to our project.

The interface.c file is the device's central software element. It uses the functionalities provided by the APIs to create an interface for the user while managing and monitoring the state of the system. It is also in charge of transmitting information to the PC management application and recognizes asynchronous requests.

The state machine shows a perfect fit with the specification provided in

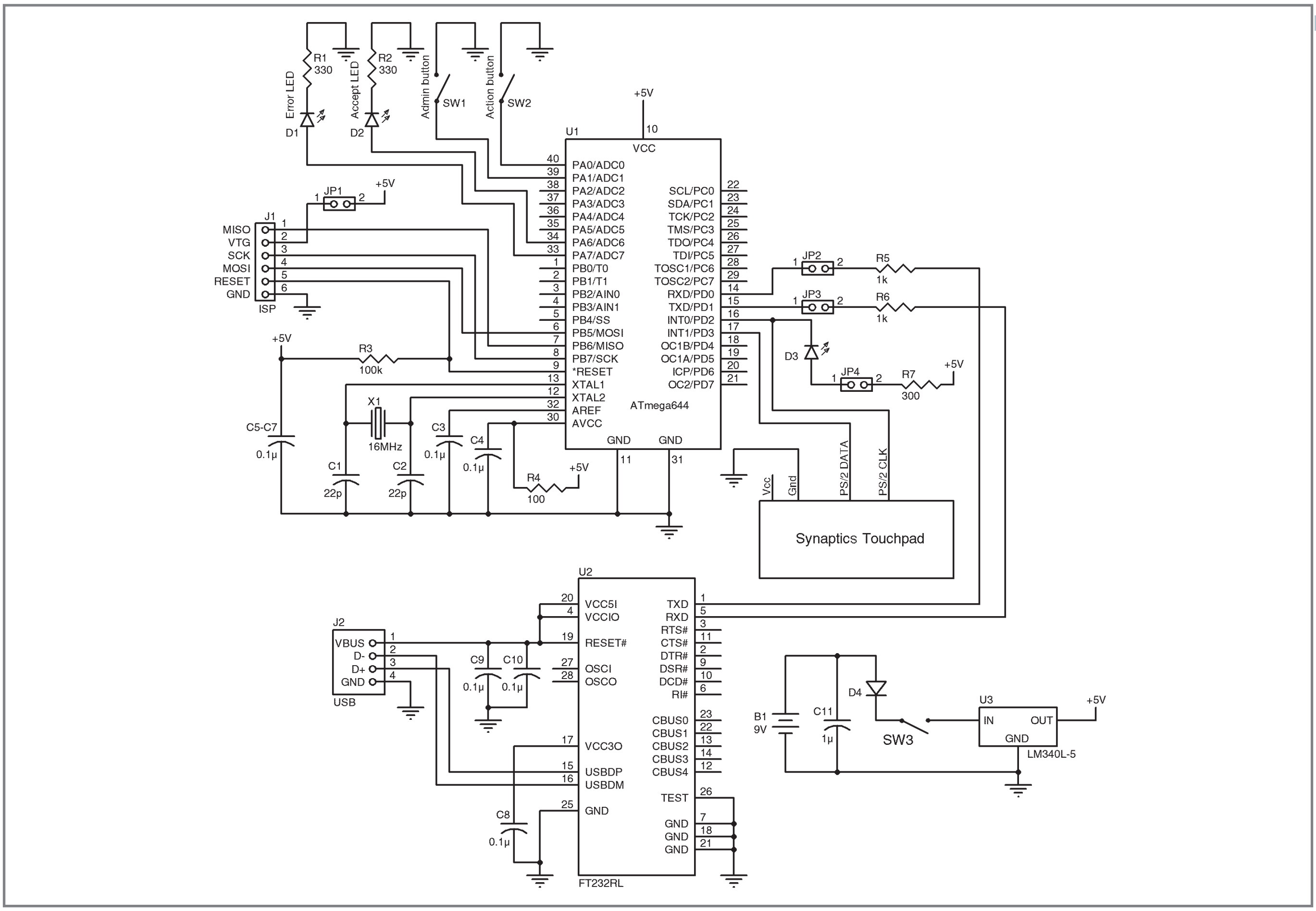


Figure 3—Here is the USB prototype board. You can see the external hardware connected to the board which includes LEDs, push buttons, and a Synaptics touchpad.

the high-level design section (see Figure 5). In the interface implementation, an additional state machine was used for debouncing the buttons, but its implementation is straightforward. Refer to the interface code for implementation details.

The central interface file is also in charge of using the touchpad API to configure and obtain information from the touchpad. We configured the touchpad in Absolute mode, which means that the touchpad sends absolute x and y positions as opposed to delta movements. We also used Stream mode, which means that the touchpad continuously sends data while it is enabled and does not wait for a request-to-send packet. When receiving a password, the touchpad is enabled, and then an external interrupt is triggered after the touchpad pulls the clock line down. Next, the entire packet is received and the touchpad is inhibited until the data is processed. After recording the data (which happens at every sample period), the touchpad is re-enabled and another absolute packet is received. Note that the way to inhibit the touchpad is implemented in the touchpad and PS/2 APIs, and it consists of pulling down the clock.

GESTURES & STROKES

The gestures unit is one of the most important elements in the system. It is where gestures are processed and matched. These features are the basis of the new

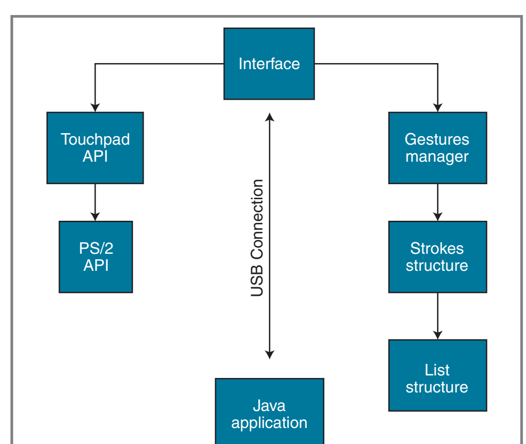


Figure 4—The final software structure consists of self-contained modules for the different functionalities required. PS/2 and touchpad APIs were implemented to obtain raw gesture data. The interface file controls the user interface and manages the system state. The gestures manager converts the raw touchpad data and compares input gestures. A Java application enables easy administration from a computer.

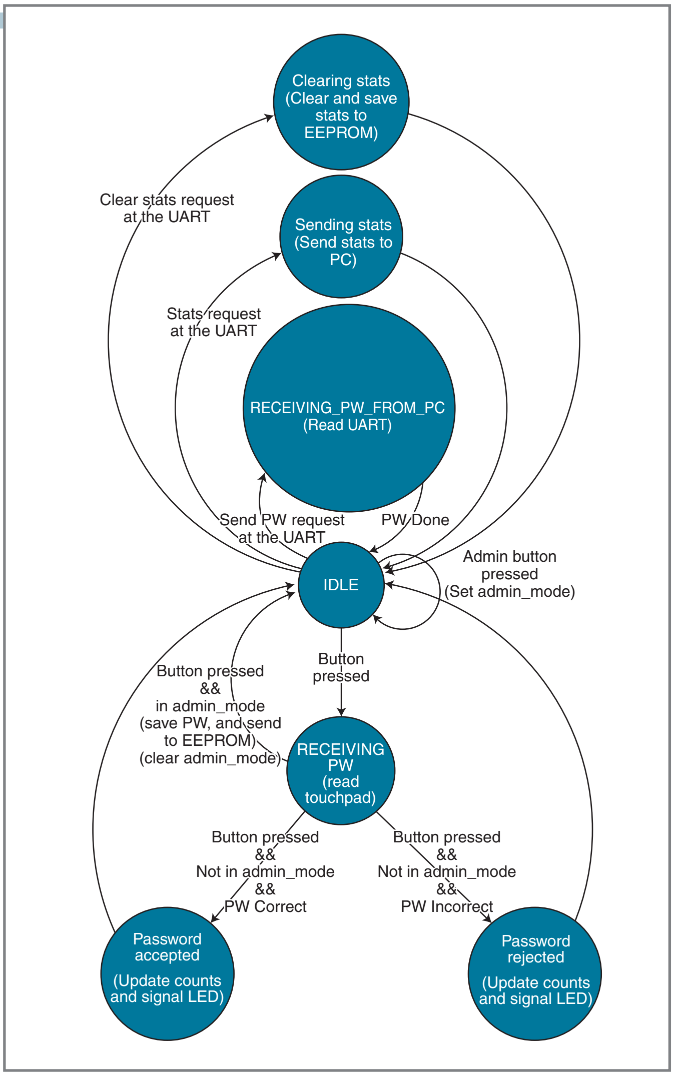


Figure 5—The finite state machine implemented in the main interface file shows how the state of the system flows according to the different actions made by a user or an administrator. Routines called at a certain state or at certain transitions are shown inside parentheses.

security approach.

The unit provides only two relevant public functions. The first converts the data from the touchpad representation of the absolute packet to the actual format that is used in the matching process. The second function receives two gestures and returns whether or not there is a match.

Instead of using raw data and having to implement a complicated algorithm, we created a data structure that represents the data in a more efficient and intuitive manner. This way, we can utilize a simpler and more effective matching algorithm. The data structure, called a stroke, represents each of the traces entered by the user without raising his or her finger (see Figure 6). The stroke structure contains fields for the initial position of the stroke, a dynamic list of delta movements for x and y, and the wait time after the stroke (i.e., time until touching the touchpad again for a new stroke). We chose these fields because, with a sufficient amount of data, they closely represent the shape and timing of a gesture. The fields are populated by constantly sampling the touchpad information while the password is being entered.

To make the data more appropriate to the problem being solved, we transform the received raw coordinates into a finite set of values that fit into a grid. More specifically, the x and y values in the stroke structure can only assume discrete values on a finite range. The reasoning behind this decision is that the user cannot have precision for distances significantly shorter than the width of a finger. Therefore, it does not make sense to take these differences into account when matching. Another advantage of this discretion is that we are able to change the 13-bit coordinate representation provided by the touchpad (which can only be fit into a 16-bit integer) into values that fit in byte variables (characters in C). The grid size is 32 × 24, which means that a position in the touchpad can only have values in this range. This size was chosen experimentally and it gave good results. Masking techniques can be used to further save space in memory. Since we are sampling the touchpad data every 50 ms, a lot of data (for a microcontroller) is involved and space-saving techniques are important.

With this data structure in hand, passwords are easily represented as arrays of strokes. More complicated techniques, such as invariant moments, were not applicable in this case because we needed to capture both shapes and timing and we were not interested in translation, rotation, or scale invariance. Also, we did not use absolute positions to represent the movement because the error gets accumulated and provides an incorrect insight of the shapes involved. More intuitively, if a user starts in a position that is slightly off from the stored password, then comparing absolute coordinates means that every coordinate would have an error even though the shape may be similar. By using the initial position and the delta, we avoid this problem and separate the different dimensions that need to be matched for a correct password.

Since the stroke structure was created with matching as

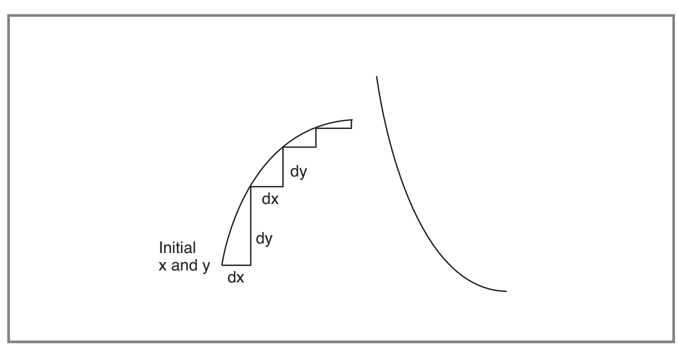


Figure 6—The stroke structure represents each of the traces entered by the user without raising his or her finger. The structure contains the initial position, a list of delta movements, and the wait time until a new stroke is entered.

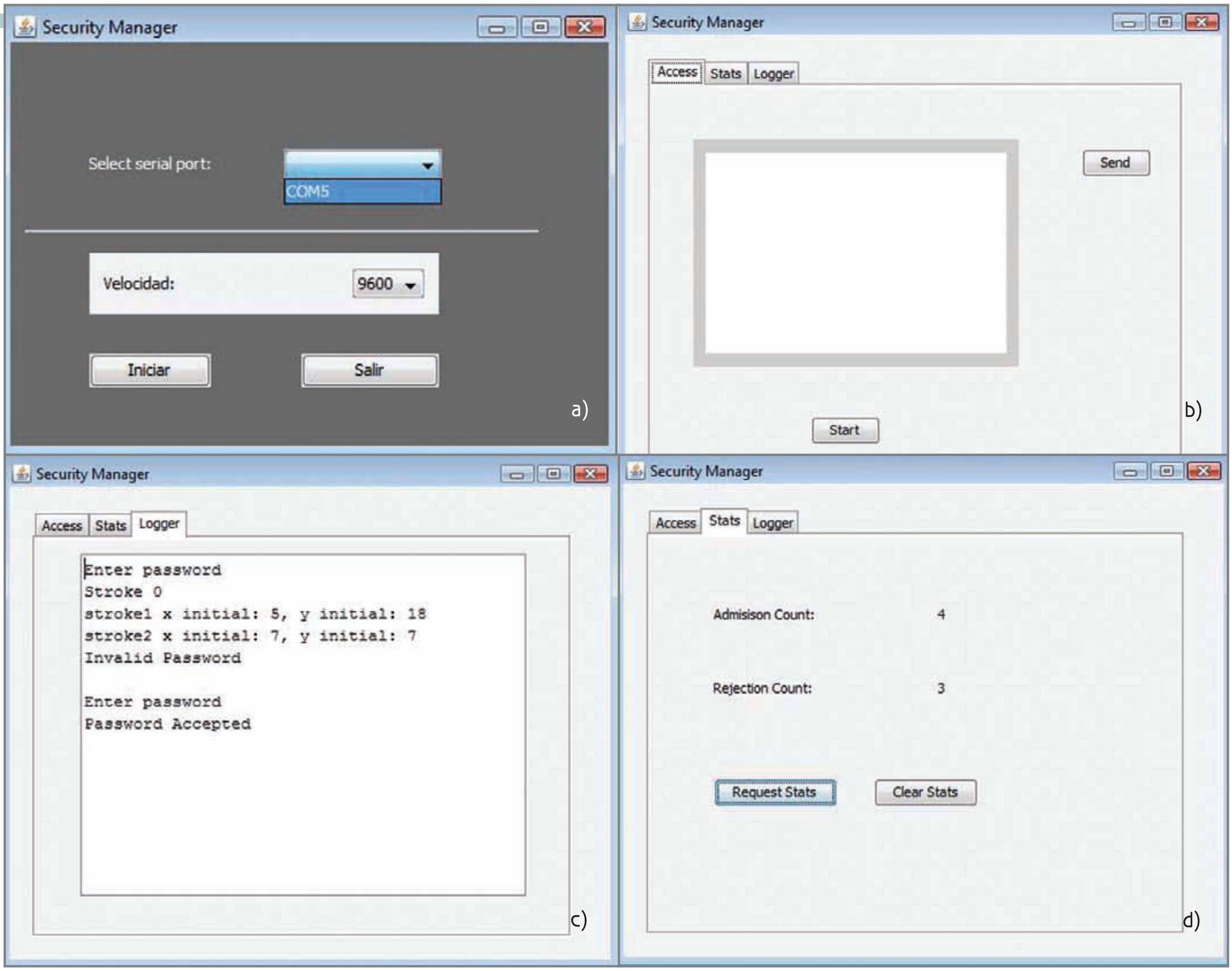


Photo 3—The PC application provides the administrator with three functionalities: setting the password for the device, monitoring it in real time, and reading the approval/rejection counts.

the main goal, the implementation of the match function is reduced to a comparison of the values in the stroke structure for each trace. The distance for each stroke element between the two passwords is calculated and according to certain ranges it may pass or fail to be matched. The intuition is that as long as the initial position for each stroke is close enough, the movements are close enough, and the wait time is close enough, the password is accepted. These tolerance ranges are used as parameters that can be tuned for having a balance between the security of the passwords and usability of the system. The more strict the tolerance ranges are, the more secure the system is—but also the more difficult it is for you to repeat a password. We set the ranges experimentally to values that produced good results, but a lot more testing and training data can be used to obtain the perfect balance.

PC MANAGEMENT APPLICATION

The PC application provides you with the three aforementioned functionalities: monitoring the device in real time, setting the password, and reading the approval/rejection counts (see Photo 3).

One point of interest is that the driver for the USB converter emulates a serial port, which simplified the communication implementation. Also, Sun does not provide support for the serial communications Java API in the Windows operating system; therefore, we needed to add an extra driver called win32comm.dll to the Sytem32 folder in Windows.

The application comes ready to use with any computer. The installation instructions for Windows are as follows: Install the Java virtual machine. Place win32com.dll in the Java\jre(version)\bin and in the System32 folder. Place comm.jar in the Java\jre(version)\lib\ext folder. Place javax.comm.properties in the Java\jre(version)\lib folder. Run SecurityManager.jar, which is in the dist folder.

The application is not a final version and it needs more debugging. Nevertheless, it is usable and anyone can install it and directly monitor the device. Since the password can be set from the computer, the gesture can be produced with a mouse. Therefore, touchpads are not the only input device that can be used for a gesture security system.

ADVANCE PROGRAM

HOT CHIPS 23

A Symposium on High-Performance Chips
August 17-19, 2011, Memorial Auditorium,
Stanford University, Palo Alto, California

HOT CHIPS brings together designers and architects of high-performance chips, software, and systems. Presentations focus on up-to-the-minute real developments. This symposium is the primary forum for engineers and researchers to highlight their leading-edge designs. Three full days of tutorials and technical sessions will keep you on top of the industry. Register for Early Registration rates before Aug.12

Wed. Aug. 17

Morning Tutorial — ***Package-Scale Power Management*** — ***UC Berkeley***

Afternoon Tutorial — ***The Open Compute Project*** — ***FaceBook***

Thursday August 18

ManyCore
- Cavium Networks' OCTEON II CN6880 Multi-Core MIPS64 Processor — **Cavium**
- The IBM Blue Gene/Q Compute chip with SIMD floating-point unit — **IBM**
- The Highly-Efficient Architecture of the Godson-Core Processor — **ICT, Chinese Academy of Science**

Security
- Universal Programmatic Access to High Quality Random Numbers — **Intel**
- The TILE-Gx ManyCore Processor: Hardware Acceleration Interfaces and Mechanisms — **Tilera**
- Building a 40Gbps, Next-Generation Virtualized Security Processor — **Cavium**

Keynote 1 **TBA** **Simon Segars, VP** **ARM**

Memory & FPGA
- The Bandwidth Engine: Serial Memory Chip Breaks 2 Billion Accesses/sec — **Mosys**
- Hybrid Memory Cube: a Re-Architected DRAM Subsystem — **Micron**
- The Xilinx Zynq Embedded Processing Platform — **Xilinx**

DSP
- The XMOS Architecture and the XS1Chips — **XMOS**
- The Worlds' Fastest DSP Core: Breaking the 100 GMAC/s Barrier — **Tensilica**

Miscellaneous
- Rethinking Algorithms: Communication-Avoiding Algorithms — **UC Berkeley**
- Electrons, Photons, Phonons, Waves, Bits, and ID: Microsoft Kinect — **Microsoft**

Panel **Ecosystem Wars: More Than Just Architecture?**

Friday August 19

Networking
- A Low-Power, High-Density 10GBASE-T Ethernet Transceiver — **Aquantia**
- A One Billion Packet per Second Frame Processing Pipeline — **Fulcrum**
- Sereno: A 2nd-Generation, Virtualized Network Interface Controller — **Cisco**

Video
- 1TOPS/W Software Programmable Media Processor — **Movidius**
- Trident Microsystems' Integrated 240 Hz Connected TV platform — **Trident**
- The Intel Quick Sync Video Technology in the 2nd Generation Intel Core Processor Family — **Intel**

Keynote 2 **The Challenge of Building Personal Robots**
Steve Cousins **Willow Garage**

Server
- Poulson: An 8 Core, 32nm Next Generation Intel Itanium — **Intel**
- T4: A Highly Threaded, Server-on-a-Chip with Heterogeneous Support — **Oracle**

Desktop CPUs
- Second Generation Intel Core MicroArchitecture — **Intel**
- Power Management of the 2nd Generation Intel Core MicroArchitecture — **Intel**
- AMD's Llano Fusion APU — **AMD**
- The Performance- and Power-Efficient Bulldozer x86-64 Server for Workstation and Desktop Processors — **AMD**

Please visit us on the web: **http://www.hotchips.org**
or drop us a line via Email: **info2011@hotchips.org**

HOT CHIPS 23

Organizing Committee

Chair
Ralph Wittig — Xilinx
Vice Chair
Larry Lewis — Apple
Finance
Lily Jow — HP
Press
Steve Leibson — Cadence
Member at Large
Allen Baum — Intel
Charlie Neuhauser — Neuhauser Associates
Advertising
Don Draper — True Circuits
Sponsorship
Amr Zaky — Broadcom
Publications
Randall Neff
Registration
Michael Sobelman — Rambus
Facilities and Video
Lance Hammond — Apple
Local Arrangements
John Sell — Microsoft
Volunteer Coordinator
Larry Lewis — Apple
Webmaster
Alice Erickson — Alice Erickson Consulting
Production
Mike Albaugh
IT
Yusuf Abdulghani — HP
Kevin Broch — Apple
Student Representatives
Sarah Bird — UC Berkeley
Yunsup Lee — UC Berkeley
Emeritus
Keith Diefendorff — Apple

Steering Committee

Allen Baum — Intel
Don Draper — TCMM Chair
Pradeep Dubey — Intel
Lily Jow — HP
John Mashey — Techviser
John Sell — Microsoft
Keith Dieffendorf — Apple
Alan Jay Smith — UC Berkeley

Program Committee

Program Co-Chairs
Bevan Baas — UC Davis
Allen Baum — Intel
Program Committee
Krste Asanovic — UC Berkeley
Bryan Chin — Cavium
Forest Baskett — NEA
Bill Dally — Stanford
Will Eatherton — Juniper
Mitsuo Saito — Toshiba
Christoforos Kozyrakis — Stanford
Dan Lenoski — Cisco
Chuck Moore — AMD
Alan Jay Smith — UC Berkeley
Jan Willem Van De Waerdt — Cypress
Sophie Wilson — Broadcom
Rumi Zahir — Intel

Founder Bob Stewart SRE

IEEE computer society

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

SSCS IEEE

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

PASSWORD TESTING & SUCCESS

The prototype can successfully receive, store, process, and compare passwords. Testing was performed to determine that the system was secure enough to keep passwords safe and to ensure that the system did not frustrate users by making it too difficult to match a password.

Three users were given the task of creating passwords then trying to correctly match their password 10 times. The results indicated that, despite a learning curve, everyone was able to match a password. Users were often unsuccessful on their first few attempts until they fully understood the importance of the position in which the symbol is traced and the time used to enter it. Omitting these initial attempts, the average efficiency was 93% (i.e., roughly nine out of 10 attempts to match a previously defined password were successful).

The next set of tests focused on security by evaluating how easy it was for other people to copy user passwords with certain information. These tests involved two people, one serving the role of user and the other serving the role of hacker, or someone trying to replicate the original password. If the hacker only heard a description of the password (e.g., a square), then the percentage of successful password copies was 0%. Similarly, the "hacker" was unable to reproduce the password after seeing it entered by the user. This meant that descriptions and direct observations of the passwords were not sufficient enough to reproduce them. These results demonstrate that, as a prototype, the device is both usable and that passwords are not easily reproducible by others. With even more parameter tuning, the tradeoff between usability and the accuracy of the passwords can be further optimized.

The prototype device satisfied each of the initial goals and demonstrated the capabilities of this new security system approach. The system worked exactly as detailed in the specifications. Passwords were successfully created, stored, and matched, and positive feedback was received from the people who used the device. The software management utility also worked properly across the USB connection.

In the future, testing with a greater sample size could provide more feedback on the tradeoffs between security and usability in our prototype and enable more optimized parameter tuning. Also, we could increase the device's utility by connecting it to the Internet for centralized remote access. Finally, the gesture security approach can be used in various scenarios aside from the one illustrated by our prototype. Entering passwords using a cell phone touchscreen, a laptop touchpad, or even a mouse are among the uses that can give the gesture security approach widespread applicability. ■

Daniel Mejía (dm472@cornell.edu) is originally from Medellín, Colombia. He attended Cornell University and graduated magna cum laude in Electrical and Computer Engineering with a minor in Computer Science. He now works as a Software Development Engineer at Microsoft. Daniel's current interests are software systems, embedded systems, Internet technologies, and machine learning.

Jayce Doose (jkd27@cornell.edu) earned a B.S. with concentrations in Electrical and Computer Engineering and Biomedical Engineering from Cornell University. He is now at Cornell working toward a Master's of Engineering in Biomedical Engineering with a concentration in Bioinstrumentation. Jayce has a strong interest in electronic biomedical devices and physical sensor systems.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar/pub/Circuit_Cellar/2011/252.

SOURCES

ATmega644 Microcontroller
Atmel Corp. | www.atmel.com

FT232R USB Converter chip
Future Technology Devices International, Ltd. | www.ftdichip.com

TM41PUM1311-2 Touchpad board
Synaptics, Inc. | www.synaptics.com

by George Martin (USA)

Design Development (Part 2)

Product Implementation

It's time to dig deeper into the design process. This month schematic capture software is used to define the hardware portion of a design.

Last time we started a product design. I laid out the framework for what an instrument might look like. And I said I was going to design a spring tester because I wanted one. Well, I still want one, but that seems too narrow a focus to keep everyone interested and handing in their homework on time. And remember, homework is 30% of your final grade. (Doesn't that thought just give you chills? Aren't you glad it's no longer true?) So, let's open up the design a bit and change it to an environmental monitor. Let's try to make it useful to monitor your house's environment or perhaps the environment of a container as it's shipped across the country. It's still basically a piece of instrumentation and, as you'll see, the block diagram changes only slightly.

What did we really do to our requirements by changing the design? What additional interfaces on the input and output sides will be needed?

What would a customer like to have in a spring tester? If you recall, I listed our design requirements in my April 2011 column (*Circuit Cellar* 249). We want methods to do the following: measure the characteristics of a spring, compare the measurements of a spring to a standard, calibrate the device, save the results of a measurement, recall the results of a measurement, print the results of a measurement, perform a go/no-go type of test, gather statistical manufacturing data, update software, work with different spring rates, and measure leaf-type springs.

Let's try the requirements again for our monitor. What would make a good environmental monitor? It should include methods to do the following: measure temperature, measure humidity, measure atmospheric pressure, measure accelerations in three axes, measure time/date (and to save and recall these readings), view the readings graphically, set alarms for out-of-limit conditions, update software, and operate from a battery.

So, it's a different list, but just how different? Very little, I think. As I go about the design process for our monitoring product, I'll use hardware schematics and software modules to implement the design. Let's call this new list of requirements, "RequirementsListRevA." I'll make it a separate document and keep it up to date. We can then compare our design to the list and see if we are still on track.

HARDWARE DESIGN

At this point in the design process, we can take a first pass at either the hardware or the software. If we were designing a messaging system that was included in a large pool of flash memories, I would start the software design first. The software design probably would be more difficult than the hardware design in that case. So, starting with the software might uncover and flush out problems earlier in the design process.

In this case, I would start with the hardware design. I'm recommending this because there are so many unknowns in that area. I'm also going to draw a schematic for each of the hardware design modules. I believe I'm going to show you about three different versions of the schematics. The first is the initial design capture, the second is a more refined version with parts identified, and the third would be a version almost completely captured in the design software. I use Altium's Design Explorer (DXP) integration platform for schematic capture and PCB design. Altium has viewer software that you can download to view the design without purchasing the complete software package. And, as a final thought in this first-pass hardware design, we should only generate schematics as a result of requirements. If there is no requirement, there is no need for a schematic.

TEMPERATURE & HUMIDITY

There are several ways to measure temperature. Thermocouples and resistance temperature detectors (RTDs) are probably the two most popular and common. I suspect that measuring humidity will have a temperature component in the calculation or circuitry. So, let's also look at humidity sensors first. I start with component distributors such as Digi-Key or Mouser Electronics and search their websites. You will find digital temperature sensors from IC manufactures (Texas Instruments and National Semiconductor come up). A search for humidity sensors shows units that give a voltage, frequency, or capacitance that's proportional to humidity.

From experience, I know Sensirion's SHT21 digital humidity sensor measures both temperature and humidity and has an I^2C interface. I know I took a big leap here, and if you didn't know about this part, you would spend hours trading off the various temperature and humidity solutions. If your system requires several remote temperature measurements, the SHT21 is probably not a good choice.

The first pass on this function is done. The next task is to draw a schematic. I use Altium, but at this level most of the design packages are basically the same (see Photo 1). I needed to create the symbol for the SHT21 and that's part of the package. I added power and ground connections that come from the datasheet. All the power and ground symbols will be connected together across all the sheets of the schematics. So the V 3.3 and GND_DIG just connect into one big net each across all the schematics. Read *Circuit Cellar* columnist George Novacek's articles about grounding for information about how to deal with connecting all the power and ground signals. I left this schematic rather rough since it's a work in progress. I feel it's important to show you the chaos that's referred to as the design process. Also the SHT21 is an I^2C device. I put module ports (the elements that will eventually connect all the schematics together) because not all the microcontrollers have I^2C ports. Some have only SPI. I do not want to hide this design issue, so I left those connections dangling. Choosing parts is like reading a Charles Dickens novel. Two hundred pages (days) later these decisions will come back. Some return as a long-lost friend with help and advice while others return and bite you in the rear (a technical term).

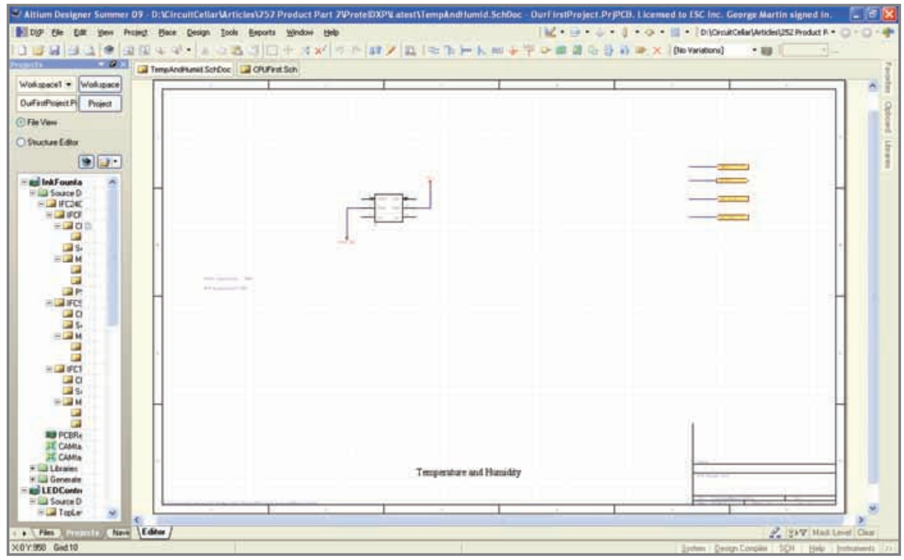


Photo 1—A screen shot showing a schematic made with Altium DXP

REAL-TIME CLOCK

Our requirements necessitate that we keep time of day. Tom Cantrell recently wrote about a device that does just that ("Time Traveler: Embedded Timekeeping and More," *Circuit Cellar* 248, March 2011). Let's use it. It's Microchip Technology's MCP794*xx* Real Time Clock/Calendar (RTCC). You'll find that most of the major IC manufacturers have similar real-time clock (RTC) devices. This device also uses the I^2C interface, so I hope we choose a microcontroller that has an I^2C interface. When I draw the schematic, I show the MCP794*xx*, a crystal, a back-up battery and holder, and all the resistors and capacitors the datasheet recommends.

ACCELERATION

Our requirements specify the measurement of acceleration in three axes. This obviously means a three-axis accelerometer. Going to the distribution webpage again, I first come across an Analog Devices ADXL325 three-axis ±5-g accelerometer. This is straightforward. Power is with 3.3-V DC. Connect capacitors to each of the three outputs to set the bandwidth of the output response and you will have three voltages proportional to the acceleration on each axis. Each of those three outputs can go into three inputs of an ADC and then be digitized to get the data into the digital domain.

If you read the previous paragraph quickly, you might have skipped over the term "bandwidth." What's that reference to bandwidth? It was never specified in the requirements. And, if you think about it a bit, the higher the bandwidth, the higher the sampling frequency required to extract the information. Is the accelerometer going to be powered up all the time? What about shocks or bumps to the system? Do we have to capture them? We could add peak and hold detection circuits to each of the three outputs and digitize those additional three outputs. Actually, each output will be at 3.3/2 V for 0 g. So, the peak detectors need to capture positive and negative gs. Ugh! It's getting complicated.

If you keep searching, you will come across an STMicroelectronics LIS331DLF accelerometer. It's a digital version of the analog device previously described. It has outputs that can be I^{2}C or SPI. It also has internal registers that can set g limits and interrupt the system when these limits are tripped. You can set the sensitivity to 2, 4, or 8 g and it costs only $5 for one quantity. At this point, I would select the STMicro part and start a schematic page for the accelerometer.

ATMOSPHERIC PRESSURE

I searched Digi-Key for pressure transducers and got more than 19,000 results. Using its selection criteria, I found the transducers that would run off of a 3.3-V supply. That narrowed the results to 66 devices. Selecting surface-mount devices, I got one page of results ranging from $4 to $25. The requirements are to measure the pressure that the instrument encounters. I'm assuming a plane trip might be the most extreme condition we encounter, and I'm not going to consider monitoring for sudden loss of cabin pressure.

For the sake of this project, let's select a Bosch Sensortec BMP085 digital barometric pressure sensor. It costs $9 for one and less than $6 per unit for larger quantities. The pressure readings represent altitude from –500 to more than 19,000 m. I think this will meet the intended requirements.

According to the datasheet, the interface is I^{2}C! It certainly is a digital world we live in today. We can design this instrument with purely serial digital interfaces. Also, the pressure transducer provides pressure and temperature readings. Hold that thought.

Again, I would generate a schematic for the atmospheric pressure signal. I would place the device on that schematic. At first pass, these schematics don't seem worth the effort. I find that once they are on paper I can review them, add devices such as decoupling capacitors, and refine the connections. Also, these pages give you a place to put more project-type information, such as power consumption, PCB square inches required, and manufacturing costs.

MEMORY & POWER

I was thinking that a small removable memory card, such as those used in cameras, would be a good choice for our memory. On the memory schematic, you'll see the connector for that device. I've also included a connector for a compact flash-type memory card. The compact flash memory card is not a requirement. I wanted to show the hardware requirements necessary to support such a device. Basically, that interface requires address lines, data lines, chip selection, and read/write control. You can do the interface in either 8 or 16 bits of data.

Again, the schematic shows the interface. The connector looks simple, but it's a much more complicated device. I put USB and Ethernet on one schematic at this point. The selection of the CPU will add a lot more detail to the parts required to connect the CPU to the appropriate interface connectors. Some CPUs contain more of the interface electronics than others. No need to detail the design at this point in the process.

This unit needs to run off of a battery. I try to talk my customers out of building battery chargers as part of a unit. Battery chargers have been known to catch fire and burn down buildings. No need to take on that responsibility. So, let's assume we have a battery pack that can be externally charged. Also, when the unit is connected to the USB, Ethernet, or an external power source, power should come from the external supply and not from the battery. This means diode ORing of some sort. Again, until we select the CPU, we probably should not put much effort into the power supply details.

WHERE ARE WE NOW?

Is this ugly, or what? It is ugly and certainly a mess. I bet you've never seen the design process presented in this manner. Well, this is the way it evolves. Also, it's typical to run into a road block during an initial pass. We did with trying to capture peak accelerations from an analog sensor. But, let's say that one of the devices (humidity) comes in at too high a price or consumes too much power. I would generate a new schematic (Temperature and Humidity Version 2) to try and solve the problem.

We know there is a second source to temperature (the pressure transducer), so perhaps we should include a humidity transducer that gives an analog voltage or a frequency output that is proportional to humidity. Yes, it means restarting the search, finding the component, creating the schematic, and then looking at this approach in the overall system design.

This design process is not like building a house where a foundation is first built and then the house fits perfectly on that foundation. It's more like a sculpture. You've got a block of stone and you start chipping it with sculpting tools. After the first full day of hammering, it still looks like a piece of stone. It's closer to becoming a statue, but it doesn't show.

There is one other schematic that I generate to hold all of the detailed schematics. I typically call this the "Top-Level Schematic." It's mostly a place holder. I might move the removable battery and the On/Off switch to that level. On/Off switch? What On/Off switch? It's not in the requirements, but we probably need one. Again, passing over the design again and again reveals more details that need to be incorporated into the design.

Before we select the CPU, I would take a pass at the design of the software. And next time I will go over that process.

OUR UGLY DESIGN

It's ugly, but we have a design. It won't work and it's not properly connected, but it's a basis for all the next steps. If you're looking into an interface, just follow the steps as if they were part of this project and put it in at the top-level schematic. Your requirements are, of course, different, but you should be able to duplicate these steps for your design.

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

PROJECT FILES

To download pictures of the schematics, go to ftp://ftp.circuitcellar/pub/Circuit_Cellar/2011/252.

RESOURCES

T. Cantrell, "Time Traveler: Embedded Timekeeping and More," *Circuit Cellar* 248, March 2011.

National Semiconductor Corp., "Circuitry for Inexpensive Relative Humidity Measurement," 1981, www.national.com/an/AN/AN-256.pdf.

Sensirion, "Datasheet SHT21 Humidity and Temperature Sensor," 2010, www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT21.pdf.

SOURCES

The Design Explorer (DXP) Integration platform
Altium Ltd. | www.altium.com

ADXL325 Three-axis accelerometer
Analog Devices, Inc. | www.analog.com

BMP085 Digital barometric pressure sensor
Bosch Sensortec | www.bosch-sensortec.com

Humidity sensors and pressure transducers
Digi-Key Corp. (distributor) | www.digikey.com

MCP794xx Real Time Clock/Calendar
Microchip Technology, Inc. | www.microchip.com

Humidity sensors
Mouser Electronics, Inc. (distributor) | www.mouser.com

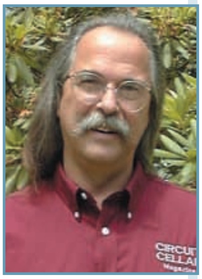
Digital temperature sensors
National Semiconductor Corp. | www.national.com

SHT21 Digital humidity sensor
Sensirion | www.sensirion.com

LIS331DLF Three-axis linear accelerometer
STMicroelectronics | www.st.com

Digital temperature sensors
Texas Instruments, Inc. | www.ti.com

by Jeff Bachiochi (USA)



Vehicle Diagnostics (Part 2)

Pre-CAN Protocol Standards

The first part of this series was a primer on automotive control systems and OBD-II data. This article explores the pre-CAN protocol standards still in use. You can use this information to diagnose trouble codes and more.

After reading my introduction last month, I hope you were curious enough to look for your car's on-board diagnostics II (OBD-II) connector. They were installed around the driver's seat in all automobiles manufactured after January 1, 1996. This single connector contains all of the networks in use today, as well as the control area network (CAN) standard, required for automobiles manufactured after January 1, 2008. We can thank the Society of Automotive Engineers (SAE)—encouraged by early EPA emissions standards—for the foresight of proposing an open-ended standard communications set for the control and monitoring of engines and other subsystems. And just as important was its plan for phasing out the various protocols in use for a single standard.

We are well past the date requiring new automobiles to use the single CAN standard. But, because vehicle lifetimes average 12 years, most of the automobiles on the road today still use one of the pre-CAN standards. Therefore, it is possible that your car uses one of the older standards. The purpose of this series is to explain each of the standards so you can make use of the information available on one of the OBD-II networks for some ulterior motive. This might be to merely diagnose a trouble code, to indicate where to look for a component problem, or to provide some additional information about a vehicle that isn't currently presented (e.g., real-time fuel consumption versus distance or MPG).

As engine compartments become jammed with increasingly complex engine control systems, it is easy to understand that what began as emission control and verification has grown into much more. Today, we have "smart" transmissions, brakes, tire pressure monitoring, and vehicle suspension, as well as creature comforts such as cabin temperature controls, navigation, and entertainment. Our automobiles are morphing

Mode	Operation
$01	Identifies what powertrain information is available to the scan tool
$02	Displays freeze-frame data
$03	Lists emission-related "confirmed" DTCs as four-digit codes identifying the faults
$04	Clears emission-related diagnostic information including clearing stored pending/confirmed DTCs and freeze-frame data
$05	Displays the oxygen sensor monitor screen and the test results gathered about the oxygen sensor
$06	Requests on-board monitoring test results for a continuously and non-continuously monitored system (There is typically a minimum value, a maximum value, and a current value for each non-continuous monitor)
$07	Requests emission-related DTCs detected during current or last completed driving cycle
$08	Enables the off-board test device to control the operation of an on-board system, test, or component
$09	Retrieves vehicle information
$0A	Stores emission-related "permanent" DTCs

Table 1—Each command begins with a mode byte. These modes are used to perform general tasks. Each mode can have numerous formats. A second PID byte further defines functions supporting that mode.

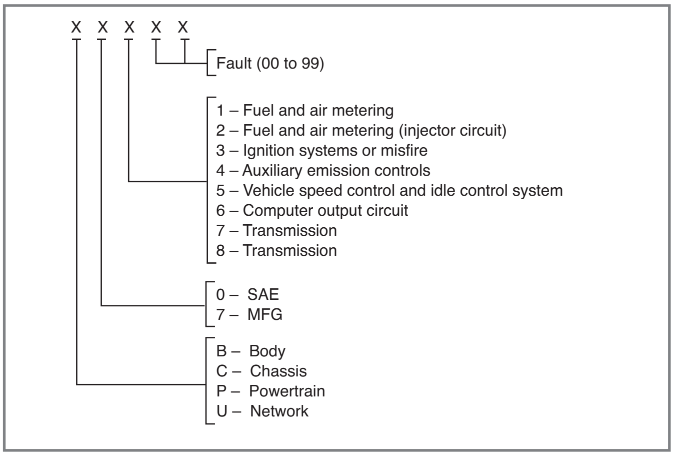


Figure 1—This diagram shows how the five-character diagnostic trouble code (DTC) can be used to determine the general fault area. (Source: AutoTap OBD-II Diagnostic Scanner www.obdii.com/dtcanatomy.html.)

into mobile homes and offices. Much of that is due to the ease with which we can move data.

PACKET DATA

The basic format is a packet beginning with a header that includes a priority, a destination, and a source. The header provides a clue to the importance of the packet along with the intended receiver and requester. Lower-priority values indicate more importance. These values are automatically used in bus arbitration to ensure important messages have priority to the bus. Since all devices on the bus have their own address, they can filter the destination of messages and respond to only their own.

Following the packet's header is the request (or response) data, which is followed by a checksum or cyclic redundancy check (CRC). The data is usually limited to 7 bytes. The data in a request packet uses a mode and parameter identification (PID) format. The mode is a single byte indicating what type of request is being made. Table 1 is a list of the 10 standard modes. Manufacturers may request the use of additional modes specific to their system. For instance, one might request mode 0x03 to discover any diagnostic trouble codes (DTCs). Figure 1 shows the breakdown of the DTCs. (You can locate specific DTCs at DTCsearch.com.) Perhaps you might retrieve a DTC of "P0012," which is a generic trouble code for "over-retarded timing of Camshaft 1." Presuming you know how to readjust the timing, you might want to remove this DTC by issuing a mode 0x04 request. If the problem isn't rectified, it will be reissued again. Any cleared DTCs (using mode 0x04) still remain in mode 0x0A. This is a permanent record of the DTC and remains there until the problem goes away.

Most modes require additional information (besides the mode value). For example, requests using mode 0x01 require a PID. Wikipedia provides an extensive list of PIDs with various modes. Most people know that a vehicle identification number (VIN) is the manufacturer's ID number of a specific vehicle. It can be found on your vehicle registration and normally on a tag on the driver-side corner of the vehicle's windshield. It can also be requested through mode 0x09 PID 0x02. Let's use this example to consider how the various protocols handle requests and replies for the VIN.

ISO 9141-2 & ISO 14230-4

Last month's introduction began with the similar ISO 9141-2 and ISO 14230-4 protocols because they closely match the familiar UART serial transmissions. Both protocols use a 10,400-bps clock for the data transmissions and the EMU goes into sleep mode when communications cease for 5 s.[1] Communications must therefore begin with a wake up. And, if you wish to keep the lines of communication open, you must make sure the packets are periodically transmitted within this "stay awake" window. ISO 9141-2 uses a "slow init," while ISO 14230-4 has an alternative "fast init," which has a speedier wake-up process.

The "slow init" process requires sending a 0x33 at 5 bps. Because the one UART on this microcontroller is used for communication with the user, all non-CAN protocols are bit-banged using software routines. Instead of cranking down the clock on a 10,400-bps routine (for the "slow init" requirement), I used a timer to create a similar output. This is (1 start bit) 200 ms low, (0x33 data byte) 400 ms high, 400 ms low, 400 ms high, 400 ms low, and (1 stop bit) 250 ms high. The EMU then acknowledges this by a communication dance at the nominal data rate consisting of sending back 3 bytes, acquiring 1 byte, and sending back 1 byte (see Table 2). At this point, the EMU is ready to receive communications.

The "fast init" (for protocol ISO 14230-4) essentially consists of a break transmission (a transmission of the active state for longer than 9 bits (at 10,400 bps, that's at least 865 µs). So a simple active 25-ms low (break), followed by a 25-ms high (pause) is all that's necessary. Once this opening sequence finishes the command 0x81, 0x66 will instruct the EMU to open communications until it is told to close communications (or a timeout occurs with a lack of communications for more than 5 s).

A bus-eye view of the initialization sequences for both the slow and fast initialization is shown in Photo 1 and Photo 2. For both of these protocols, you need to set up a timer to make sure a simple command is periodically sent to keep the lines of communication open. As this occurs in the background, you might not be aware that it is happening.

SAE J1850 PWM & VPW

Although these protocols don't require any special "wake-up" or "stay-awake" signaling, they don't use asynchronous UART timing. PWM uses a fixed frequency PWM pulse for each bit time, with a one-third (active) PWM as a "1" and

Protocol	Dance sequence (following 5-bps initialization)		
	Response	Acknowledge	Response
ISO 9141-2	0x55, 0xE9, 0x8F	0x70	0xCC
ISO 14230-4	0x55, 0x08, 0x08	0xF7	0xCC

Table 2—The "slow init" procedure begins with a 5-bps character 0x33 followed by this dance sequence (at the nominal 10,400-bps rate) between the EMU (response) and the user (ACK).

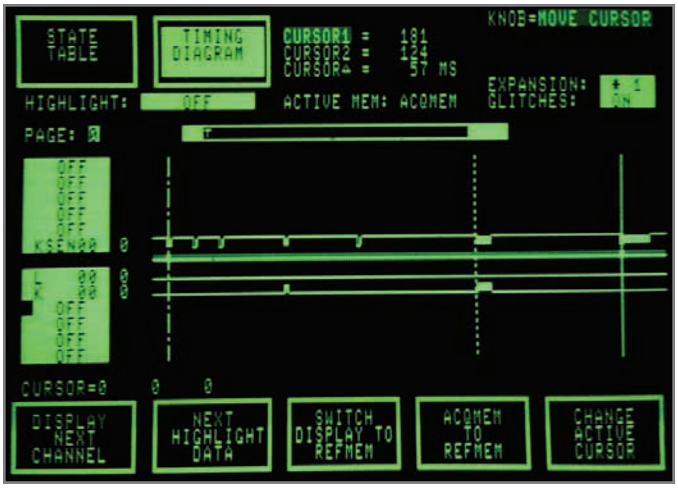


Photo 1—This trace is triggered after the 5-bps "slow-init wake-up" pulse train. The 3-byte response from the EMU is seen in the K_SEN trace, with a 1-byte response on the K line and a 1-byte response again by the EMU (see Table 2).

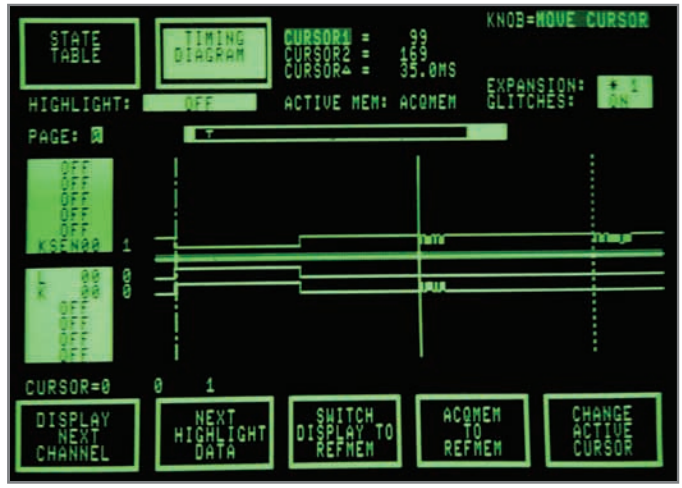


Photo 2—The entire "fast-init wake-up" single pulse. The "wake-up" command follows on the K line with the first response seen on the K_SEN line.

two-thirds (active) PWM as a "0." VPW uses a variable-frequency pulse for each bit, with a passive width of 64 µs as a "0" and a passive-pulse width of 128 µs as a "1." Alternating active pulses use the opposite format with an active width of 64 µs as a "1" and an active-pulse width of 128 µs as a "0." For both I use a timer that's one third the length of a PWM bit time (32 µs for a 96-µs bit time, 8 µs for a 24-µs bit time).

For PWM, each packet begins with a start of frame (SOF) pulse (active pulse of 2× the bit time, followed by a passive pause). Each following bit always begins active. After a timer overflow a "1" returns to the passive state while a "0" remains active for the second third of a bit time. Finally, the last third is always in the passive state. With this protocol bit timing is synchronized by the rising edge of each bit time. The PWM protocol uses an acknowledge (ACK) for each packet transmitted by the source and the destination. Photo 3 shows the packet transmissions and ACKs.

With VPW, each packet begins with an active SOF pulse similar to the PWM protocol except there is no passive portion required. In fact, the first bit (and all of the following bits) begin on a change of state. Each bit time is either 64 or 128 µs. The same timer can be used (32 µs) as in the PWM format. Multiple overflows of 2X or 4X are used to produce a "0" and "1" passive state, respectively. Since the state changes after each bit, it is easy for the receiver to stay in sync with the transmitter. When producing active state bits, the logic is opposite. A "0" is indicated by a 64-µs wide state. A "1" uses a 128-µs wide state. No ACKs are used with the VPW format (see Photo 4).

AT USER INTERFACE

Suppose you need to produce a widget that electronically reads the VIN and displays it for DMV confirmation. The standalone code written to interface with the OBD-II would not require any user intervention. There would be no need for a serial port (unless it is the display interface, such as a serial LCD). My point here is that the code could be much simpler if a user-friendly interface is not required.

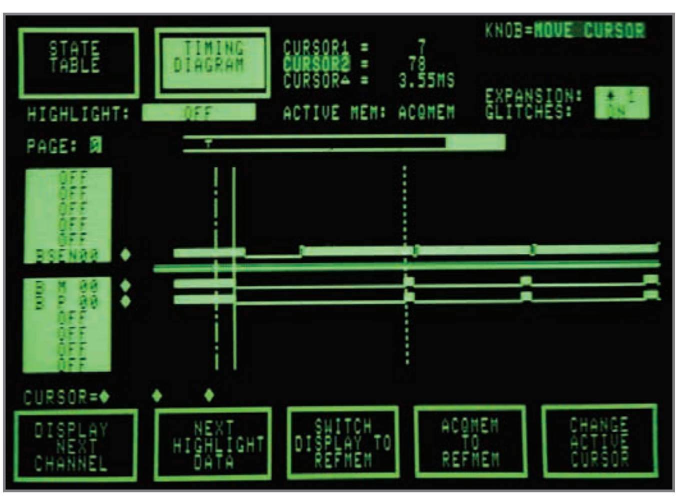


Photo 3—Look closely at the tail end of the command sent on the B+/– traces and you can see the EMU ACKing. When the EMU responds with data, you must ACK the transmission.

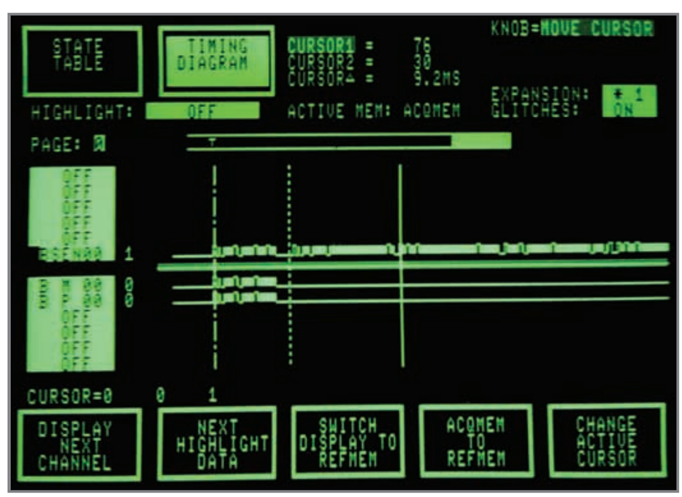


Photo 4—The VPW uses no ACKing. When a smaller amount of the transmission is displayed, you can begin to see the actual bit timing as in this screen shot.

AT Command	Description
<CR>	Repeat the last command
E0, E1	Enable, disable echo
L0, L1	Enable, disable linefeed
CAF0, CAF1	Enable, disable autoformatting
DP	Display current protocol
DPN	Display current protocol number
H0, H1	Enable, disable headers
S0, S1	Enable, disable spaces
SHxyz	Set header to xyz
SHxxyyzz	Set header to xxyyzz
TPx	Set to protocol x
TPxA	Set to protocol x (auto if necessary)

Table 3—These AT commands are implemented in the present project. While these commands are only a fraction of those supported on the ELM processor, they will enable access to most of those necessary for requesting data from the nine supported protocols.

Without a dedicated task, such as just displaying the VIN, you must be open to all kinds of finagling and enable a user to ask for data.

Back in 1977, Hayes Communication presented the "AT" command set to enable configuration of their 300-bps Smartmodem. Commands would only be recognized if they were preceded by the ASCII characters "AT." It's easy to imagine running out of letter combinations for potential commands in some all-inclusive standard. So, it's the idea of recognizing the difference between data and a command (and not a standard list of commands) that enables the AT command to live on in more products than the modem.

Last month, I mentioned ELM Electronics as creating a OBD-II-to-serial user interface chip. I based this series of columns on emulating this chip, as it has been adopted as "the" standard for OBD-II interfacing. The scan tools and applications found for vehicle diagnostics speak ELM. For those not interested in playing with the low-level code I'm presenting, you can purchase a preprogrammed microcontroller from ELM. ELM microcontrollers use an AT-style command set to enable users to change parameters and ask for specific information from any of the potential interfaces.

I've started implementing those AT commands that will get you started. Table 3 gives you an idea of what is available with this project. First and foremost, you need a way of specifying the use of a protocol or enabling the device to search through the nine available protocols looking for one that will answer a request. The user interface expects any input to be actual data bytes that are to be stuffed into a packet (request) unless it begins with an ASCII "AT." A carriage return (CR) indicates the end of input and the preceding data should be acted upon. An input of "ATTPx" is the command to set a particular protocol, where "x" is the protocol of choice (see Table 4). By appending an "A" to the command "ATTPxA," the requested protocol will be used. But, if there isn't a good answer, it will automatically go through the list until it finds a working protocol or runs out of protocols.

As input is received, it is stacked into an input buffer. Each alpha character is converted into uppercase. When a CR is received, the buffer is searched for acceptable commands or data (which has been received in pairs of hexadecimal digits). One to seven values are considered "legal" for a packet of data to be sent via the requested protocol. Unless this data forms an actual ODB-II command, it won't receive a reply. So, the user must know what to send. In the CAN protocol, the first data byte of the packet states how many bytes (of the eight potential) are significant. This number can be calculated and automatically added or can come directly from the input string. This automatic formatting can be enabled/disabled via the "ATAFx" command.

The serial port connection for this project provides you with a window into the microcontroller. You will need to open your favorite terminal program and connect to this port to see what's going on. When the circuit is powered up it will respond with the "ELM" sign-on message, which will enable my emulation code to be recognized by third-party software. You should see this on your terminal:

```
ELM327 v1.4b
>
```

To receive the VIN, enter "0902" as an input string. This requests a mode 0x09 (infotype) PID 0x02 (VIN). If the present protocol gets a response, you will see:

```
>0902
SEARCHING...
49 02 01 00 00 00 31
49 02 02 47 31 4A 43
49 02 03 35 34 34 34
49 02 04 52 37 32 35
49 02 05 32 33 36 37
```

If you see the phrase "SEARCHING...," then the present protocol didn't work and other protocols were tried until a response was found. At this point you won't know what protocol actually received a response. This can be queried by entering the following display protocol (DP) command:

```
>ATDP
AUTO, ISO 9141-2
```

If that matrix of digits doesn't look much like a VIN, it's because the VIN was longer than the number of characters that can be transmitted in a single packet. There is also some formatting here that helps identify how the multiple packets are to be reassembled. The first two digits are a reflection of the original command sent, which was "0902." Bit 6 of the mode byte was set and transmitted as "4902."

Protocol number	Protocol
1	J1850 PWM
2	J1850 VPW
3	ISO 9141-2 (slow initialization)
4	ISO 14230-4 (slow initialization)
5	ISO 14230-4 (fast initialization)
6	ISO 15764-4 (500 kbps, 11-bit CAN)
7	ISO 15764-4 (500 kbps, 29-bit CAN)
8	ISO 15764-4 (250 kbps, 11-bit CAN)
9	ISO 15764-4 (250 kbps, 29-bit CAN)

Table 4—A list of protocols recognized by this ODB-II circuitry and application

Byte 3 is the packet number of the response. Here you received packets one to five. If you toss out the leading filler bytes 0x00, you have 17 binary values for the VIN. The ASCII values of those bytes indicate a VIN of "1G1JC5444R7252367."

At least that's what I would have seen if my 2000 Caravan supported the 0902 VIN command. In reality, it is not supported. How do you know what's supported? The obvious answer is that only supported requests get a reply. However, there is a mechanism for determining this. PID 0x00 has the function of indicating which PIDs are supported for each mode. Requesting a "0900" command returns:

```
>0900
49 00 01 3C 00 00 00
49 00 01 3C 00 00 00
```

Hmm. There are two identical responses. What's with that? Unless we turn on the header display, these look identical. After using the enable headers command "ATH1" the responses make more sense:

```
>ATH1
>0900
48 6B 41 49 00 01 3C 00 00 00 7A
48 6B 40 49 00 01 3C 00 00 00 79
```

Here you see from the headers that the first response is coming from ECU 41 (header byte 3) and the second response is coming from ECU 40. Previously, bytes 4 to 6 contain a command and a packet number. The last byte is a checksum of the data that spans bytes 7 to 10. If you expand the data for these 4 bytes into their binary equivalent, you get:

```
3C 00 00 00 = 0011 1100 0000 0000 0000 0000
0000 0000
PID3 supported—^
PID4 supported——^
PID5 supported———^
PID6 supported———-^
```

These 4 bytes indicate the PID support for PID1-32 (0x00–0x20). A "1" (in a bit position) indicates support for that PID. PID20 is another support PID. If the last bit was a "1," then PID32 (0x20) would hold the support information for PIDs 33–64, and so on. You can see that the VIN, PID 02, is not supported in this vehicle.

MULTIPLE ECUs

As I noted in the first part of this series, OBD-II makes room for up to eight ECUs. Expect to see these slots fill up in the future. My elderly Dodge Caravan shows two ECUs, an engine control module (ECU 10) and a transmission control module (ECU 18) are being used (circa 2000). Initially, the header default values are set to request information by using a functional address that all ECUs will respond to. Note that each of these responds with a physical address, in this case 10 and 18. I can carry on private conversations with any of the individual ECUs by using their addresses in the headers, which begin every packet. So, there are a few commands that enable me to set the header values. The header for the protocols discussed here requires 3 bytes, following the set header "ATSH" command. The three pairs of hexadecimal bytes indicate xx-priority, yy-source, and zz-destination. Higher-priority values mean lower priority. The source is where the request comes from; this interface defaults to an address of 0xF1. As previously stated, the destination can be a functional address or the physical address of a specific ECU.

When headers are displayed, you'll note that the last byte (checksum) is displayed as well. The ISO 9141-2 and ISO 14230-4 protocols use a simple 8-bit checksum initialized to zero. The J1850 protocol uses the "CRC-8-SAE 1850" for calculating CRC values. I included an 8-bit look-up table in the application to help simplify this function.[2] To use the table, begin with the CRC initialized to 0xFF. For each byte, you can obtain a new CRC by first creating a table pointer. (XOR the present CRC with the new byte.) Use this pointer to get a new CRC value from the table (of pre-calculated 8-bit CRCs).

INTO THE FUTURE

Next time, I'll finish up with the last of the four protocols, ISO 15765-4 CAN. This is the mandatory future of all manufactured vehicles, and this future has already begun. You might have noticed that last month's schematic used a generic connection for TX and RX, the user serial communications channel. Next month, I'll also look at alternatives to the standard RS-232 link.

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

REFERENCES

[1] SAE International, "E/E Diagnostic Test Modes, Product Code J1979," http://standards.sae.org/j1979_201009, 2010.

[2] J. Bachiochi, "Error Checking," *Circuit Cellar* 250, 2011.

RESOURCES

Search Engine for DTCs, 2011, www.dtcsearch.com.

SOURCES

PIC18F2480 Microcontroller
Microchip Technology, Inc. | www.microchip.com

ELM327 v1.4b OBD-to-RS-232 Interpreter
Elm Electronics | www.elmelectronics.com

STN1110 Multiprotocol OBD II-to-UART interpreter and ECUsim 2000 Multiprotocol OBD II simulator
ScanTool.net, LLC | www.scantool.net

NOW, THERE'S EVEN MORE TO DISCOVER.

NEW!

+

+

The upgraded Elektor PLUS subscription!

- All 11 issues including the Summer Circuits edition
- Included in your PLUS subscription: Annual DVD 2011
- **50%** cheaper than normal retail price
- Up to **40% discount** on selected Elektor products
- Elektor is delivered to your doorstep every month
- Read your copy before everyone else
- **NEW:** On your personalized Elektor PLUS website, you have permanent access to the three latest editions of the magazine in PDF format, as well as to a fast Elektor search engine!

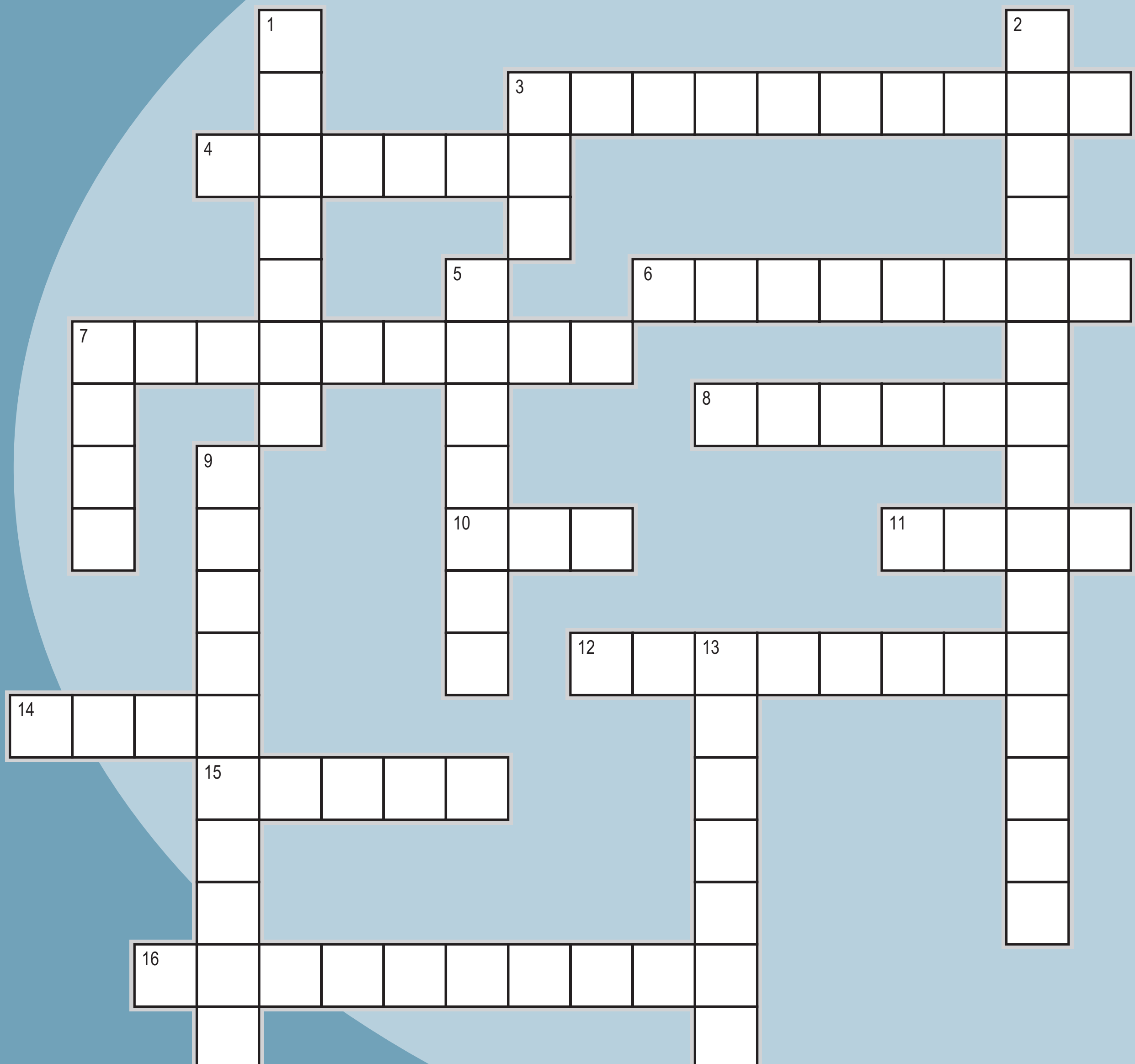
NEW: exclusive access to www.elektor-plus-usa.com!

Your Elektor PLUS subscription gives you **exclusive access** to the new website **www.elektor-plus-usa.com** where the three latest editions of Elektor magazine are available in PDF files (i.e. the current issue and the two preceding ones). With a simple click you download the complete edition (front to back!) or any single article. **www.elektor-plus-usa.com** also supplies the most extensive Elektor search engine found on the web (all issues since 1998).

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

elektor

CROSSWORD



Across

3. An image of your monitor [two words]
4. The "L" in LUT [two words]
6. Uses TCP/IP to serve billions
7. Highly caffeinated connection
8. Uses a telephone to make a connection [two words]
10. Digital-to-analog in reverse
11. An activity that uses a board or a mouse
12. Collection of instructions
14. A seemingly endless list of instructions
15. An emphatic search engine
16. Your 'Net ID [two words]

Down

1. Stored on a computer or in a jar
2. Can be confused with "Just kidding" [three words]
3. To secretly collect and report information, or a data standard
5. Serial data transmission with software [two words]
7. 3,600 s
9. Positive sign [two words]
13. French thinker (1768 – 1830) known for his contributions in physics and math

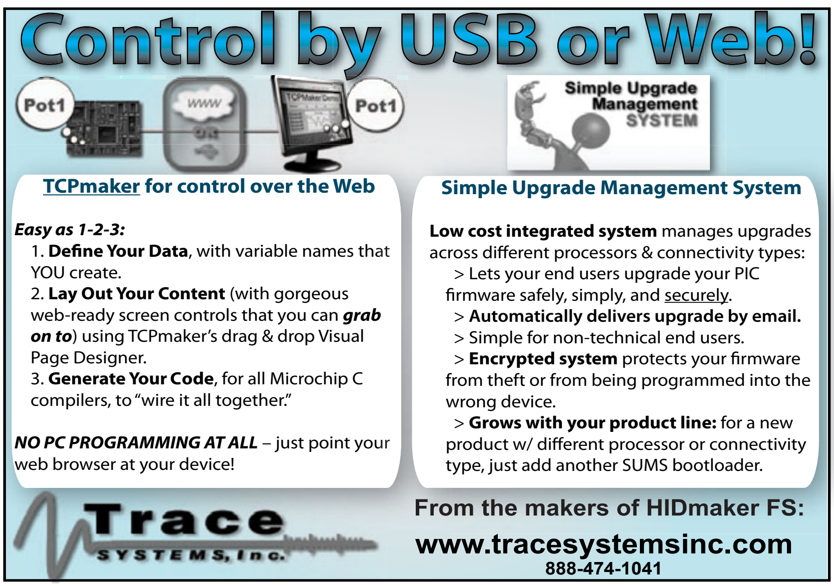
The answers will be available in the next issue and at www.circuitcellar.com/crossword.

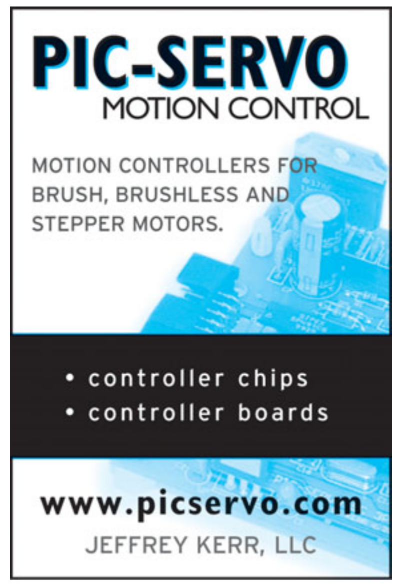
IDEA BOX

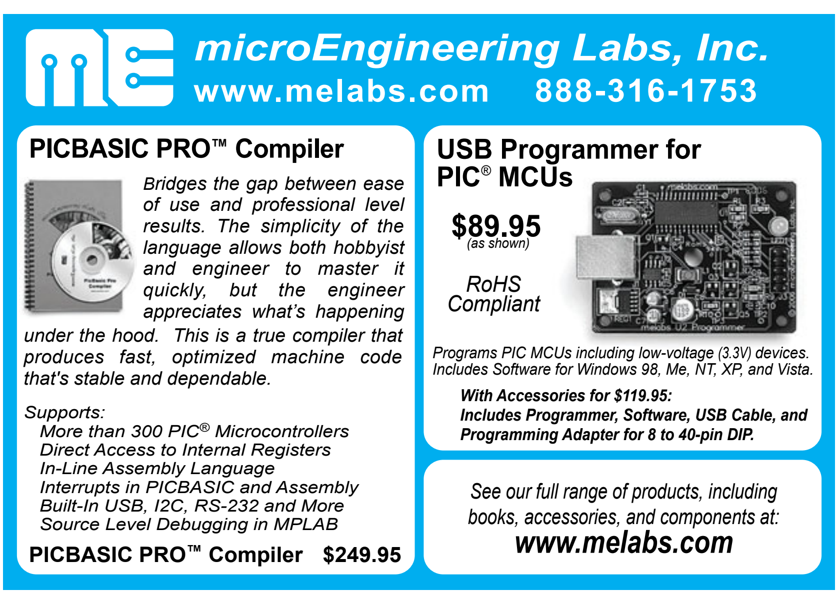
THE DIRECTORY OF PRODUCTS AND SERVICES

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, PO Box 180, 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.







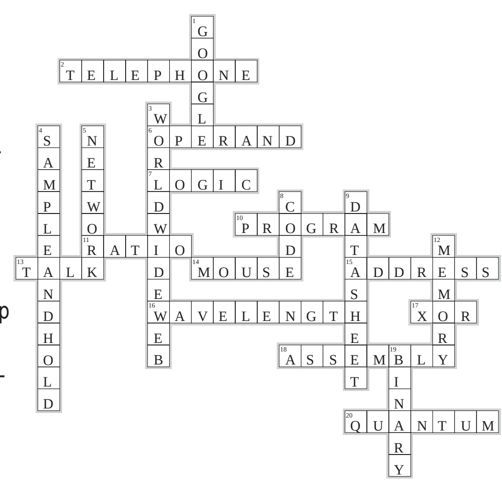
CROSSWORD ANSWERS from Issue 251

Across

2. **TELEPHONE**—Used in lieu of text, e-mail, IM, etc.
6. **OPERAND**—Gives instructions to operate
7. **LOGIC**—Uses reasoning
10. **PROGRAM**—Plan
11. **RATIO**—1:2
13. **TALK**—Most basic form of communication
14. **MOUSE**—A device cats are especially fond of
15. **ADDRESS**—A designated location
16. **WAVELENGTH**—The distance between two points
17. **XOR**—May denote exclusivity
18. **ASSEMBLY**—Speaks the language of programmable devices
20. **QUANTUM**—An energy value

Down

1. **GOOGLE**—Search
3. **WORLDWIDEWEB**—Woven by a very large spider [three words]
4. **SAMPLEANDHOLD**—S/H [three words]
5. **NETWORK**—Allows for sharing, if you're in the loop
8. **CODE**—A communication process that converts information into symbols
9. **DATASHEET**—Provides performance details
12. **MEMORY**—Storage
19. **BINARY**—Minimum of two



INDEX OF ADVERTISERS

PREVIEW of August Issue 253

Theme: Embedded Development

Stochastic Chemical Kinetics on an FPGA

Standby Current Management in Embedded Systems

Access Control System Design: Build an MCU-Based PIN Reader

Find a New Direction: Build a Low-Power Digital Compass

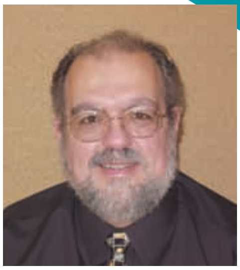
THE DARKER SIDE **RF and High-Speed PCBs:** How to Avoid Basic Mistakes

THE CONSUMMATE ENGINEER **Electro-Hydraulic Servo Valves**

ABOVE THE GROUND PLANE **Stepper-Motor Drive:** The Basics

FROM THE BENCH **Vehicle Diagnostics (Part 3):** ISO 15765-4 and the CAN Protocol

PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

Reality, Maybe

Last month I described the gory details of how my gaming laptop had self destructed. At the end, the only hope was a tech myth about possibly resurrecting the laptop by reflowing its circuit board in a kitchen oven to fix fissures and whiskers. While Viking may be happy to hear about new reasons to buy their appliances, I just can't bring myself to believe in the myth enough to sacrifice either what's left of my computer or my perfectly good oven. Does that make me too smart to believe everything I read or just chicken?

So, how many other stories like this are leading us down the wrong path? Here are a few that I have seen being discussed lately.

Expensive cables always work better than cheap ones. You're kidding, right? But people fall for this every day. You've just dropped $1,000 on a new HDTV and the sales guy asks if you have a 10′ HDMI cable to go from your receiver to the TV (so you can get your new toy working tonight). Because he tells you his cable is "premium quality"—like the TV you just bought—you fork over an extra $100 for an HDMI cable. (That cable is $17 online and every bit as good. Of course, we won't discuss the 500′ reel of monster cable I bought for my home theater way back when). Score one for the bad guys.

Adding more monitors increases productivity. Well, yes and no. In my case, a couple of extra monitors help me work faster because I use them like post-it boards to hold spreadsheets, text memos, and stationary webpages rather than flipping through overlapping windows on a single monitor. (Of course, adding a fourth monitor may have contributed to blowing up my laptop, but that's another story.) Studies suggest that using an extra monitor, like I do, aids productivity. Using the extra monitor to run Hulu, webcams, e-mail, and video games, not so much.

MPAA and RIAA follow your every move on P2P sites. Not directly, but they're trying hard by using whatever sources they can. Apparently, they contract third-party services to monitor the Internet and watch where "their" stuff shows up illegally (in their estimation) and how it is being downloaded. Opinions suggest that the occasional song or two won't create ripples, but frequent large downloads of movies or software will definitely attract attention. If you are a big BitTorrent user, short of employing an anonymizing proxy website, you're toast.

There will be catastrophic results if you turn off your PC without first shutting down Windows. Way back when, maybe, but it shouldn't be happening anymore. Since XP SP2, I think hard shut downs haven't been quite as catastrophic as the good old days. I certainly don't advocate just pulling the plug, but unless you have other hardware problems, it will reboot (and frequently scold you for ignoring Mother Microsoft). However, if you are a real Neanderthal around the power cord, I suggest you switch to Windows 7. It seems to be more idiot-proof and forgiving.

Moore's law will always be true. Perhaps, but at some point, we may not pursue it quite as rabidly. Basically, the issue is cost, not science. Moore's law is generally about computing power and the number of transistors we can pack on a chip. Opinion suggests that continued decreases in the cost of chip production have enabled the "law" to repeat doubling of computing power, but that may not always be the case in the future. The limit on the "law" may be economic, rather than scientific.

The Intel i7-2600K processor in my new computer has almost a billion transistors and it is indeed faster than the quad-core I bought a few years ago. Sandy Bridge processors, such as the 2600K, use 32-nm architecture to fit all of those transistors in such a small space. But the fabrication facilities to produce them aren't cheap, and Intel's next venture, called Ivy Bridge, has an even smaller 22-nm process. Needless to say, the factories to make these wafers cost billions! The technology may be there 10 years from now to make 12-nm transistors that exponentially add to today's speeds. But companies may simply choose not to produce them at the same rate as Moore's law just to slow down the expense. We won't be going back to six-transistor radios anytime soon, but the people waiting for mega-processing power to create true AI may have to wait a bit longer.

So, that's my short take on reality maybes. There are still more "myths" out there like "unlimited broadband" (what a crock) and "overclocking blows up your PC" (they must be kidding), but something should always be left as an exercise for the reader. In the meantime, I think I'll go sit down in my big stuffed chair and reflect on how life wasn't all that bad with just a six-transistor radio.

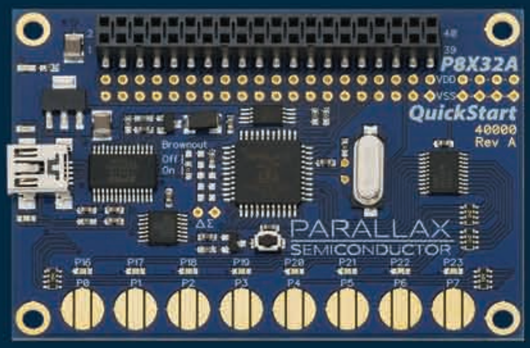
steve.ciarcia@circuitcellar.com

Steve

PARALLAX SEMICONDUCTOR

Parallax Inc. recently announced the creation of a new division, Parallax Semiconductor, formed specifically to focus support for OEMs with volume commercial applications using the company's ICs, such as the multicore Propeller.

The creation of this new group has no impact on the existing Parallax business. Vice-President Ken Gracey says "Parallax continues full-speed ahead with our historic mission serving a diverse range of customers and applications with innovative products. Parallax Semiconductor simply extends that mission to meet the needs of a larger commercial audience."



One of the things that differentiates Parallax Semiconductor is our premier customer support. Not only do we strive to develop outstanding application notes, documentation, and software, but we have many ways to ensure our customers are able to get the help they need from a variety of different portals. We will make sure customer questions and projects are routed to the correct department such as Tech Support or a Field Application Engineer. Customers can feel confident their needs will be met.

Free samples for industrial applications available at **www.parallaxsemiconductor.com**

Or call to speak to an FAE: (916) 632-4664

Find us on Twitter, Facebook, and YouTube