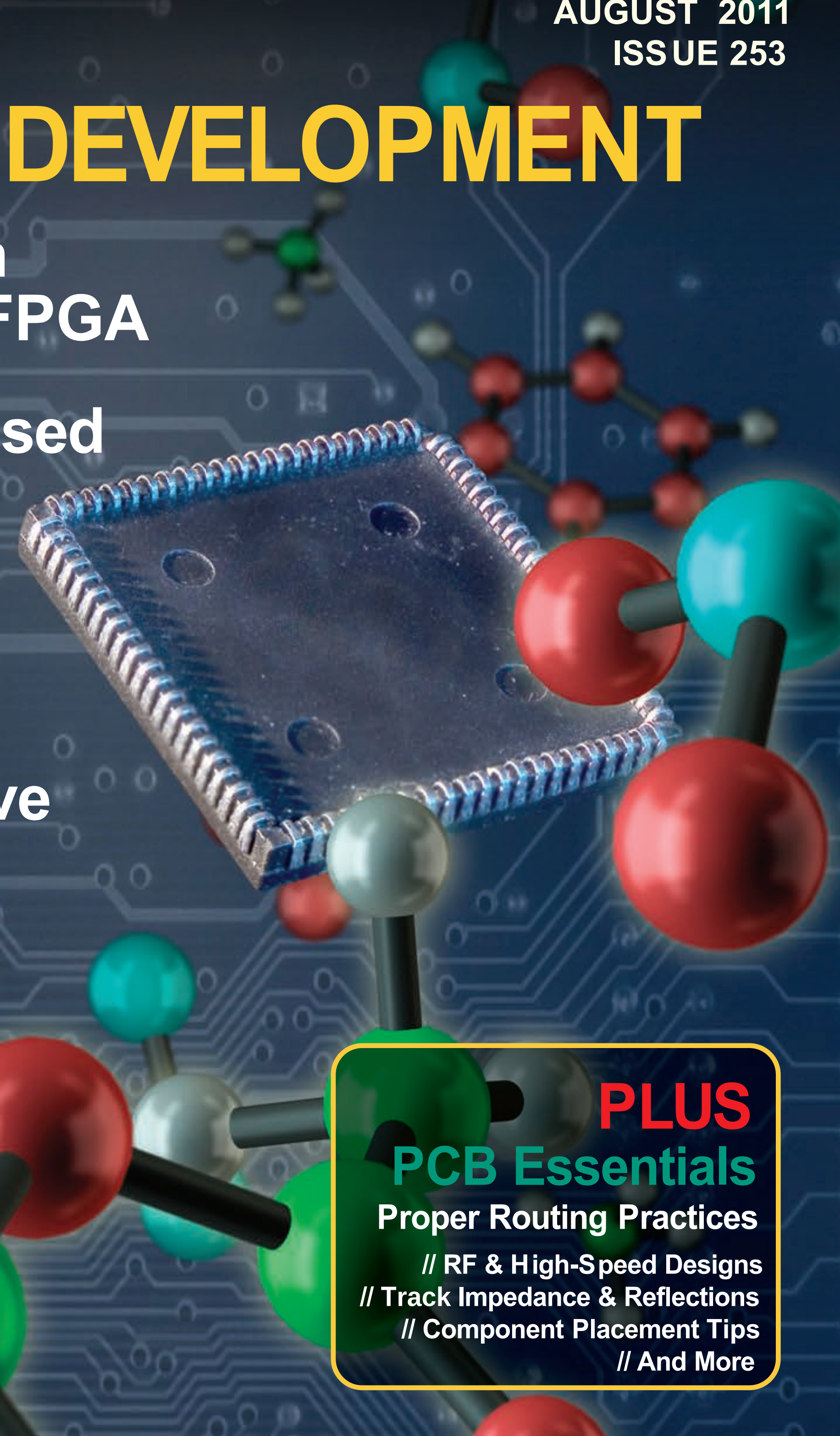
CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

AUGUST 2011
ISSUE 253

EMBEDDED DEVELOPMENT



Chemical Reaction Modeling with an FPGA

Microcontroller-Based PIN Reader

Design & Program a Digital Compass

Stepper Motor Drive Examination

Electro-Hydraulic Servo Valves

$7.50 U.S. ($8.50 Canada)

www.circuitcellar.com

PLUS

PCB Essentials

Proper Routing Practices

// RF & High-Speed Designs
// Track Impedance & Reflections
// Component Placement Tips
// And More

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:

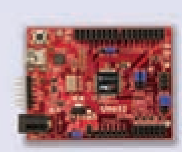
- Application development using an environment based on the original Arduino IDE, modified to support PIC32 devices while still supporting the original Arduino line. Leverages existing code examples, tutorials and resources.
- Pin-out compatibility with many existing Arduino shields
- Higher performance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

Development Board Comparison

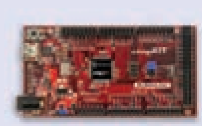
Feature	Core	Performance	Program Memory (KB)	RAM (KB)	Additional Features
chipKIT™ Uno32	32-bit	80 MHz	128	16	PMP/PSP/RTCC
chipKIT Max32	32-bit	80 MHz	512	128	USB, 2x CAN, Ethernet, DMA, RTCC

IT'S EASY TO GET STARTED!

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



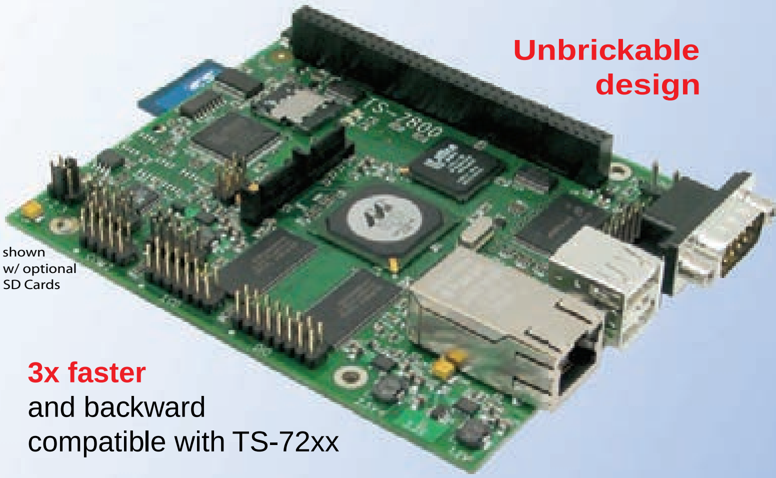
chipKIT Uno32 (TDGL002)



chipKIT Max32 (TDGL003)

Embedded Systems

High-End Performance with Embedded Ruggedness



Unbrickable design

shown w/ optional SD Cards

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

New Frontiers for Embedded Tech

By now you've heard that the Elektor group—which owns *Circuit Cellar* and *Elektor* magazines—did some additional branching out in June 2011 with the purchase Audio Amateur. The result is that the group now publishes two more excellent monthly magazines, *audioXpress* and *Voice Coil*. This focus on growth and development isn't relegated solely to the company's accounting and marketing professionals. It's also a focus in *Circuit Cellar*'s editorial meetings. We're well aware of the exciting developments in the electronics industry, and thus we're always on the lookout to find how embedded tech devices are being used in new ways. Our goal for this issue was to highlight embedded tech in different fields of interest, and then create the most well-rounded issue to date. I'm happy to say we succeeded. We provide articles on topics ranging from servo valves to FPGA-based chemical modeling to standby current management. As you'll see, embedded tech is being used in many interesting places. Let's review.

Embedded Development: Alberto Ricci Bitti, an embedded systems designer in Italy, is the sort of well-rounded engineer who inspires others. His interests range from programming to wireless communications. On page 16, Alberto provides interesting details about some of the six projects he wrote about in *Circuit Cellar*.

Electro-Hydraulics: On page 20, George Novacek covers electro-hydraulic servo valves (EHSV). He presents insightful information about hydraulic actuators, torque motors, and more.

FPGAs and Stochastic Chemical Kinetics: Turn to page 24 to read about an amazing convergence of chemistry and electrical engineering. Bruce Land explains how to use an FPGA to accelerate chemical reaction modeling.

DIY Portable Navigation: You can build a simple-yet-handy digital compass with the right parts and a little know-how. Jesse Marroquin used a microcontroller, a magnetic sensor, an LCD, and a display driver to get it done (p. 30).

Current Management: To be a successful electrical engineer, you must know how to manage current and prevent power dissipation. Sivakumar Govindarajan provides useful tips on page 36.

Access Control: Aleksander Borysiuk built an MCU-based PIN code reader. On page 42 he presents the hardware and RTOS information you need to build your own.

Stepper Motors: Need to incorporate a stepper motor in a design? Ed Nisley presents need-to-know stepper motor configurations and describes a classic motor drive system (p. 50).

CAN Protocol: Jeff Bachiochi wraps up his three-part series "Vehicle Diagnostics" on page 56. This month he covers the CAN protocol.

PCB Design: When you need some electrical engineering advice, Robert Lacoste is your man. He is skilled in everything from RF design to DDS theory to time domain reflectometry. On page 64 he provides helpful tips on proper PCB routing.

As you can see, embedded tech is everywhere.

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
Carey Penney

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.

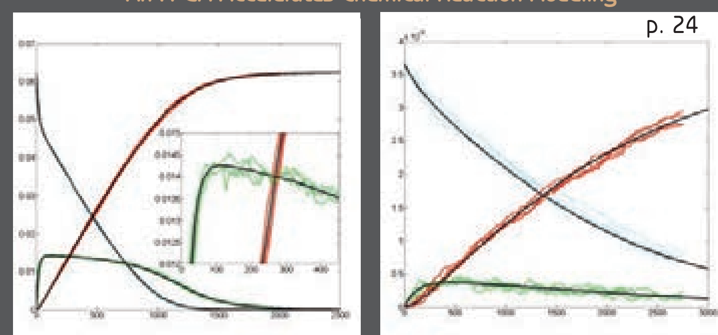
CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Vernon, CT 06066. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $45, Canada/Mexico $60, 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 253

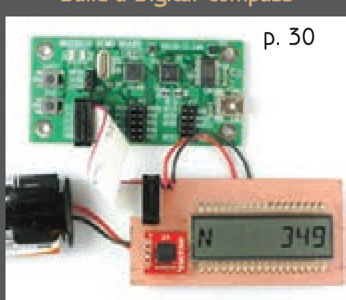
August 2011 • Embedded Development

An FPGA Accelerates Chemical Reaction Modeling



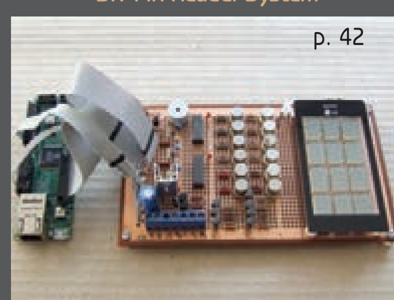
p. 24

Build a Digital Compass



p. 30

DIY PIN Reader System



p. 42

Stepper Motor Configurations

p. 50


DAS ORIGINAL SEIT 1994
PCB-POOL®
Beta LAYOUT
Europe's leading prototype supplier now also in America
Order direct from our US facility!
PCB-POOL RULES!
DAS ORIGINAL SEIT 1994
PCB-POOL
Beta LAYOUT
order!
PCBs 1-6 layers
free!
Free Stencil
fast!
8 hour service
NEW!
cool!
Aluminium-Core IMS PCBs
Beta LAYOUT
www.pcb-pool.com
email : sales@pcb-pool.com
Toll Free USA : 888-977-7443
All registered brands remain the registered trademarks of the respective manufacturer !
p-cad 2006
Altium Designer
EDWin
cadence
GraphiCode
PROTEUS
NATIONAL INSTRUMENTS
RS-274-X
Easy-PC
Sprint Layout
PULSONIX
www.pcb-pool.com
Beta LAYOUT

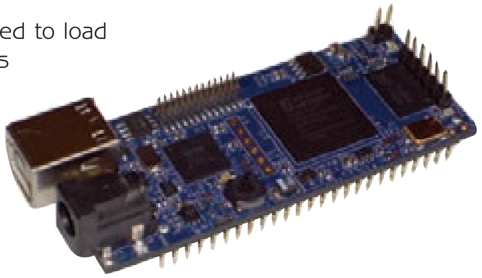
XILINX FPGA & FTDI USB HIGH-SPEED 2.0 MODULE

The **DLP-HS-FPGA2** is a high-speed, low-cost, compact prototyping FPGA module based on silicon from Xilinx and FTDI. The module uses an XC3S400A-4FT256C FPGA, which is larger than the DLP-HS-FPGA. It has the same high-speed USB 2.0 interface based on the FTDI FT2232H, and 32 M × 8 DDR2 SDRAM from Micron, like the previous version.

The DLP-HS-FPGA2 can be used for rapid proof of concept or in educational environments. The module includes a 10,000-line reference design for the Spartan 3A FPGA. The design was written in VHDL and built using the free Xilinx ISE WebPACK tools. The reference design includes a USB interface block, a user I/O block, a DDR2 SDRAM interface, a heartbeat pulse generator, and a clock generator.

As an added feature, the second channel of the dual-channel USB interface is used to load user bit files directly to the SPI flash. No external programmer is required. All that is needed to load bit files to the FPGA on the DLP-HS-FPGA2 is a Windows software utility (which is included with the module), a Windows PC, and a USB cable.

The DLS-HS-FPGA2 module costs **$179.95** and includes a working reference design.

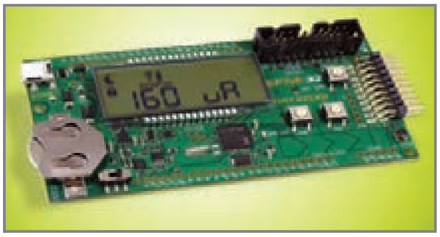


DLP Design, Inc.
www.dlpdesign.com

STARTER KIT FOR ENERGY-SENSITIVE PRODUCT DEVELOPMENT

The **EFM32TG-STK3300** is a starter kit for Energy Micro's ARM Cortex-M3 based EFM32 Tiny Gecko microcontrollers. The kit provides users with the functionality to create ultra-low-power system designs.

The starter kit features active mode current consumption of only 160 µA/MHz, making it the largest in a family of 23 Tiny Gecko devices. The kit includes an 8 × 20 segment LCD supported by Tiny's sub µA LCD controller and a selection of light, touch, and motion sensors, backed by the MCU's low-energy sensor (LESENSE) interface. Tiny's LESENSE interface with "wake-on-touch" capability runs independently of the Cortex-M3 core and enables a mix of up to 16 resistive, capacitive, or inductive sensors to be autonomously monitored while the microcontroller remains in its 900-nA Sleep mode. Other low-power peripherals offered by the kit's MCU include a 150-nA low-energy UART and a 350-µA, 1-MSPS, eight-channel, 12-bit ADC. The starter kit also offers a choice of GPIO pins, serial communication ports, and debug connections.



The EFM32TG-STK3300 integrates Energy Micro's advanced energy-monitoring system (AEM), which enables system current consumption and voltage to be accurately viewed in real time, and code to be debugged for adverse energy drains. Visualization and analysis of the kit's energy consumption data is via energyAware Profiler, a free download available from Simplicity Studio.

The EFM32TG-STK3300 costs **$69**.

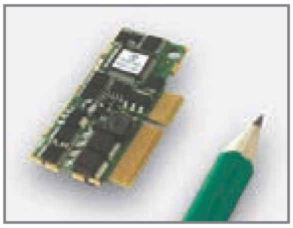
Energy Micro AS
www.energymicro.com

COMPACT, FLEXIBLE, 75-W INTELLIGENT DRIVE SOLUTION

The **iPOS** family of intelligent servo drives provides high-power density on compact boards. Modularly designed to cover both low- and high-volume applications, the first member of this family, iPOS3602 (36 V, 2 A, 75 W), is a complete motion control and drive solution packed on only 21 × 54 mm of PCB space. The drive integrates basic motor control functions, motion control, and PLC features on a small-sized plug-in module. Equipped with CAN/CANopen interfaces (with additional EtherCAT interface also available), the iPOS drive controls any rotary or linear brushless, DC brush, or step motor.

iPOS3602 executes complex motion programs at drive level, using its built-in motion programmer and the high-level Technosoft Motion Language (TML). It can also operate as an intelligent EtherCAT and CANopen slave. The motion capabilities of the iPOS card include position or speed profiles (trapezoidal, S-curve), third-order PVT and first-order PT interpolation, electronic gearing and camming, and analog or digital external reference, complemented with the cyclic synchronous position, speed, and torque modes specific to EtherCAT.

Contact Technosoft or a distributor for pricing.



Technosoft, Inc.
www.technosoftmotion.com

NEW PRODUCT NEWS

Edited by John Gorsky

RISC-BASED TOUCHSCREEN COMPUTER

The **SeaPAC R9-8.4** combines a powerful RISC-based embedded computer with a bright 8.4" TFT LCD to create a wide-temperature, rugged, flat-panel computer designed for a variety of HMI and control applications. Featuring LED backlight technology, the system offers an extended operating temperature range of –30° to 70° C without requiring a heater or cooling fan.

Powered by a 200-MIPS ARM9 microprocessor, the SeaPAC R9-8.4 is available with up to 256 MB RAM and 256 MB flash memory for maximum performance in embedded systems. Standard I/O includes Ethernet, serial, USB, CAN bus, and digital interfaces. For intuitive operator interface, the system includes a resistive touchscreen that is suitable for a wide range of industrial environments. Designed for panel-mount applications, the SeaPAC R9-8.4 provides an aluminum front bezel that maintains NEMA 4/IP65 protection from sprayed liquids.

The SeaPAC R9-8.4 software package is equipped with the Sealevel Talos I/O Framework, which offers a high-level object-oriented .NET compact framework (CF) device interface. This interface provides an I/O point abstraction layer with built-in support to easily interface the system's I/O. The package also includes Windows CE 6.0 BSP binary and low-level drivers for system I/O as well as Linux support.

Pricing for the SeaPAC R9-8.4 starts at **$1,599**.

Sealevel Systems, Inc.
www.sealevel.com

NPN

CHIPSET & FIRMWARE DETECTS HAZARDOUS ARC FAULTS

Designed to detect hazardous DC arc faults in photovoltaic (PV) systems, the **SolarMagic** arc detection chipset is an analog front end (AFE) that features three highly integrated ICs and multi-band dynamic filtering firmware. The SM73201MM 16-bit, 50 to 250 kSPS, differential input micropower ADC digitizes the arc signal after the AFE gain and filtering stage and sends the digital signal to the microcontroller. The SM73308MG low-offset, low-noise, RRO-operational amplifier provides the V_{REF} mid-point for the arc-detect AFE. The SM73307MM dual-precision, 17-MHz, low-noise, CMOS input amplifier provides gain and filtering of the arc signature signal.

The RD-195 arc detection reference design, including an evaluation board, bill of materials, and schematic can be downloaded at www.national.com/rd/RDhtml/RD-195.html.

The SolarMagic arc detection chipset is priced at **$7.90** in quantities of 1,000, including a license for the MBDF firmware.

National Semiconductor Corp.
www.national.com

FIVE-DEGREE MOTION-SENSING MODULE

LSM320DL multi-sensor modules include five degrees of freedom, high thermal and mechanical stability, and advanced power management options. The modules are well suited for a wide range of consumer and industrial applications, such as high-precision motion-detection based applications in mobile phones, navigation systems, game controls, and many portable electronic devices.

LSM320DL modules feature a three-axis digital accelerometer with a two-axis digital gyroscope and can detect acceleration up to 16 g and angular rate up to 2,000 dps along the pitch and yaw axes. The modules include power-down and sleep modes, which is beneficial for battery-operated portable devices with power constraints. They also feature an embedded first-in, first-out (FIFO) memory block, so there is no need for continuous communication between the sensor module and the host processor. The devices can operate with any supply voltage in the range of 2.4 to 3.6 V.

LSM320DL multi-sensor modules are priced at **$3.50** in 1,000-piece quantities.

STMicroelectronics
www.st.com

USB ANALOG I/O & FLEXIBLE-SIGNAL CONDITIONING

The **DAQ-PACK Series** is an integrated multifunction data acquisition and control system that can add portable, easy-to-install high-speed analog and digital I/O capabilities to any PC or embedded system with a USB port. The system performs signal conditioning, such as RC filtering, and offers current inputs, RTD measurement, bridge completion, thermocouple break detection, voltage dividers, small signal inputs, and sensor excitation voltage supply. Sustained sampling speeds up to 500 kHz are available for 32, 64, 96, or 128 single-ended or differential analog inputs. Groups of eight channels at a time can be independently software configured to accept different input ranges. A unique, real-time internal calibration system enables the system to compensate for offset/gain errors to provide accurate readings. The DAQ-PACK Series also features two 16-bit analog outputs, 16 high-current digital I/O lines, and a programmable 16-bit counter.

Available accessories include a variety of cables and screw terminal boards for quick and easy-to-use, out-of-the-box connectivity. In addition, a DIN rail-mounting provision and a gold-zinc mounting plate are available for panel mounting.

Pricing for the DAQ-PACK Series, which includes an enclosure, power supply, and USB cable, starts at $**872**.

ACCES I/O Products, Inc.
www.accesio.com

OSCILLOSCOPES DESIGNED WITH EDUCATORS IN MIND

Designed for educators, **InfiniiVision X-Series** low-cost/high-performance oscilloscopes feature an oscilloscope, logic analyzer, protocol analyzer, and WaveGen function generator all in a single box.

Agilent's 26 DSO and MSO models (2000 X-Series and 3000 X-Series oscilloscopes, ranging from 70 to 500 MHz) offer flexibility with features including Agilent's Educator's Oscilloscope Training Kit, which contains built-in training signals and an oscilloscope lab guide and tutorial for quick set up and easy upgrades; professional teaching tools using a graphical slide set to quickly teach oscilloscope and measurement basics; and a built-in 20-MHz function generator. The series also offers the ability to upgrade bandwidth and memory, and to add 16 digital channels and WaveGen and measurement applications at any point after purchase (features unique to Agilent scopes), all with a standard three-year warranty.

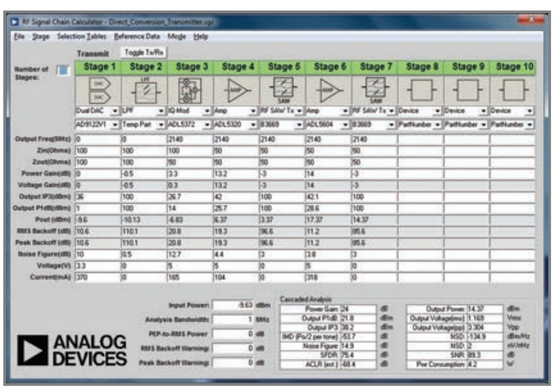
Pricing for the 26 Agilent InfiniiVision X-Series oscilloscopes, DSO and MSO models, starts at **$1,230**.

Agilent Technologies, Inc.
www.agilent.com

RF DESIGN TOOLS SIMPLIFY RF SYSTEM DEVELOPMENT

The **ADIsimRF** design tool is the software accompaniment to Analog Devices's complete portfolio of RF-to-digital functional blocks, enabling engineers to model RF signal chains using devices from ADI's RF IC portfolio. Enhancements to the ADIsimRF 1.5 include more visibility into signal levels within the signal chain, the ability to set RMS and peak-power warning thresholds, and calculators for measuring interstage power loss.

The ADIsimRF design tool provides calculations for parameters within an RF signal chain, including cascaded gain, noise figure, IP3, P1dB, and total power consumption. The ADIsimRF tool also contains embedded data from many of ADI's RF ICs, which designers can easily access using pull-down menus. Device tables assist in component selection.



ADIsimPLL 3.4 development software has been upgraded to support new products in ADI's PLL portfolio, the ADF4351 PLL for base station and general-purpose applications, and the ADRF6850 integrated broadband receiver for satellite applications. The ADIsimPLL design tool is a comprehensive PLL synthesizer design and simulation tool. All key nonlinear effects that can impact PLL performance can be simulated, including phase noise, Fractional-N spurs, and antibacklash pulse.

The new versions of these design tools are available free of charge from ADI's website.

Analog Devices, Inc.
www.analog.com

SINGLE-CHIP IR MEMS TEMPERATURE SENSOR

The **TMP006** enables contactless temperature measurement in portable consumer electronics with a single-chip passive infrared (IR) MEMS temperature sensor. This capability enables system designers to optimize performance while providing a more comfortable user experience. The TMP006 can also be used to measure temperature outside the device, enabling new features and user applications.

The TMP006 integrates an on-chip MEMS thermopile sensor, signal conditioning, a 16-bit ADC, a local temperature sensor, and voltage references on a single 1.6 × 1.6 mm chip. It provides a complete digital solution for contactless temperature measurement that is much smaller than other thermopile sensors.

The low-power sensor uses only 240-µA quiescent current and 1 µA in shutdown mode. It also supports a wide temperature range of –40° to 125° C with an accuracy of ±0.5° C (typical) on the local sensor and an accuracy of ±1° C (typical) for the passive IR sensor. Communication with the TMP006 is via an I^2C/SMBus digital interface.

An evaluation module, the TMP006EVM, costs **$50**. The TMP006 costs **$1.50** in 1,000-unit quantities.

Texas Instruments, Inc.
www.ti.com


Where the next big thing
meets the light of day.
Somewhere a back room conversation has caused a burst of inspiration.
And it will be unveiled here. Come see a bright idea light the way to tomorrow.
2012 International CES
THE GLOBAL STAGE FOR INNOVATION
PRODUCED BY CEA
Tuesday, January 10–Friday, January 13, 2012
Las Vegas, Nevada | Register Now at CESweb.org

Test Your EQ

Edited by David Tweed

ANSWERS for Issue 252

Answer 1—No, it isn't true.

In bypass applications, the only aspect of "behaves like a capacitor" we really care about is the magnitude of its impedance, which needs to be low in the frequencies of interest. The circuit being bypassed generally does not care about the phase angle—i.e., capacitive or inductive—associated with that impedance.

A capacitor's impedance is at a minimum at its self-resonant frequency, and is typically "low enough" for at least an order of magnitude on either side of that frequency.

However, this does explain why it is often necessary to use several different sizes of bypass capacitors in very broadband applications. Each size keeps the power supply impedance low over a few decades of frequency.

Answer 2—There are many types of data communications systems, which, in lieu of using a self-clocking code, rely on there being enough one-zero and zero-one transitions in the data itself to provide a frequency reference. Long strings of ones or zeros can make such a system temporarily fail while it tries to recover timing, so designers use what are called "data scramblers" (pseudorandom number generators) to randomize the data.

This works really well when the payload is, for example, a set of multiplexed audio (telephone) channels. However, in these days of purely digital packet-based (e.g., Internet) traffic, it is possible for the payload of a packet to match relatively long sequences in the data scrambler, producing long strings of ones or zeros even after scrambling.

In fact, if one knows the characteristics of the scrambler being used—and this is usually specified in the relevant public standard—it is possible to deliberately design a "killer packet" that will achieve this. Even though the alignment of the packet with the scrambler is still random, sending enough of them will disrupt a typical network within a few seconds (or less).

Answer 3—The algorithm is relatively straightforward. The combination of the adjust() calls and the left shift of everything constitutes multiplying a 3-byte (6-digit) number by two in BCD. The original 16-bit number is converted to decimal one bit at a time starting with the MSB, by multiplying the result by two and then adding the next bit from the original binary number.

If you work through an example that has just one bit set in the binary number, you'll see its decimal "weight" computed in the result. If multiple bits are set, their weights are combined in the proper way.

Note that the function adjust() is equivalent to the x86 instruction decimal adjust for arithmetic (DAA), except that the adjustment is being done before the shift, which is perfectly valid.

Answer 4—Yes, this algorithm generalizes easily. You just need to specify the correct number of bytes to hold the BCD result for the size of binary number that you're converting:

Binary bytes:	1	2	3	4	5	...
Loop iterations:	8	16	24	32	40	...
Decimal digits:	3	5	8	10	13	...
BCD bytes:	2	3	4	5	7	...

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

Microcontroller-Based Innovation

An Interview with Alberto Ricci Bitti

Alberto Ricci Bitti is an Embedded Systems Designer who develops industrial controllers and instrumentation interfaces for Eurek Elettronica. In June 2011, I interviewed Alberto about his projects—both old and new—and his passion for hardware and solving "complex problems."
— Nan Price, Associate Editor

NAN: *Circuit Cellar* readers know you as an innovative designer and article contributor. Can you tell us more about where you live and what you do?

ALBERTO: I live in Lugo, a small city located in the territories of Emilia Romagna, northern Italy, that is known for its agricultural tradition, its small industries, and for being the Italian "motor valley" (Ferrari and Lamborghini are just around the corner).

I'm an embedded systems designer and a software/firmware developer with a solid passion for hardware. My skill set has been defined as quite an "anomaly" in the industry, which is traditionally split between the worlds of hardware and software engineers. Although I don't excel in either of the two worlds, all I can say is that I like getting my hands dirty with solder flux and I know how invaluable an oscilloscope is, but I'll also spend hours refining a graphical user interface (GUI) to improve its usability or its aesthetic appearance and refactoring its code to make it more clean.

Currently, I work for a small company, Eurek Elettronica. It started as a third-party manufacturer (it was one of the first SMT assembly lines in the territory), later it added a development division to fulfill the requests for a complete design-to-product service. The design lab consists of five people.

NAN: What first sparked your interest in electronics and embedded design?

ALBERTO: I've always been fascinated by electricity, lamps, and magnet coils. I was the kind of kid who loved making his own toys. When I was 12, I built a pinball machine (well, sort of). When I was 14, a buddy showed me his dad's laboratory, which was full of fascinating instruments, and handed me some electronics magazines. A few days later I asked for a subscription to an electronics magazine as my birthday gift and I started experimenting with electronics. I haven't stopped since then, and I'm 47.

My first circuit was a 555-based "metronome." Without understanding the basics, I started trying any combination of resistor/capacitor values, then adding photoresistors, and even replacing the loudspeaker with an LED spinning on a motor. You get the idea.

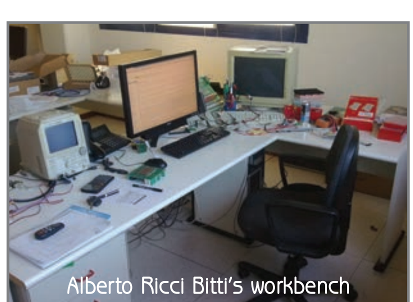
Later came the home computer era. I developed my programming skills writing software for the Commodore C64, Zx Spectrum, and Acorn BBC ("Beeb") home computers. Meanwhile, I worked on my electronics skills by building all sorts of projects I was able to find in magazines. Then, one day, I merged the two fields interfacing an A/D converter to the expansion bus of the C64—building an electrocardiogram (ECG) and testing it with the help of a friend who was starting to study cardiology. "Don't try this at home," I should say, but it was memorable fun.

NAN: When did you first begin reading *Circuit Cellar*? What drew you to the magazine?

ALBERTO: Circa 1990, a friend handed me a copy of *Circuit Cellar* and I immediately loved it. At that time I was reading *Elektor* and the now defunct *Electronics and Wireless World*. I was attracted by *Circuit Cellar*'s no-frills, "by engineers, for engineers" attitude. Clearly, not a magazine for the masses, with a very strong focus on its engineering niche. I don't know how to say it in English, but *Circuit Cellar* projects weren't just original, they were "true."

NAN: What was your first MCU-based project?

ALBERTO: It was a hotel TV set, developed for a major hotel chain, built around an 8051 derivative from Philips. It provided hotel guests with TV checkout, weather information, smartcard-based movie channels, room facilities operated from a TV remote, and all the bells and whistles in 8 KB of EPROM. Onscreen display was implemented writing to teletext chips (a text service similar to closed captioning, popular in Europe) connected to the controller via the same I^2C bus used for the tuner, audio volume, card reader, and I/O pins. Programming was done in a specialized


Alberto Ricci Bitti's workbench

language (similar to BASIC); debugging was both painful and expensive, requiring command-line driven emulators connected to bond-out chips. In retrospect, it was incredible how many functions were packed into such a small amount of memory and how lucky I was to get all that stuff running on my debut in the microcontroller arena. Actually, it was a very educational experience, where I got hands-on experience of infrared remote control, video, teletext signals, I^2C buses, smartcards, and RF tuners. And, most importantly, I learned that one can't really do firmware development without a 'scope on his desk.

NAN: You have a knack for designing small, MCU-based projects for everyday usage, many of which can be completed fairly quickly. In fact, on your website (www.riccibitti.com), many of your projects are touted as, "fast and easy," "very few parts," or "no programming skills required." Tell us what compels you to build these types of simple projects.

ALBERTO: Like every engineer, I'm fascinated by complex problems (and at work, I often work on complex projects that take months to complete), but I instinctively seek the simplest or the most elegant solution I can think of.

I appreciate the beauty of simple solutions. I also like to use—or more appropriately hack—everyday devices. Some of my designs (the calculator-based logger or the DVD-based thermometer) have been reviewed as examples of "lateral thinking," but are also examples of functional abstraction: a graphing calculator is a device for visualizing data (therefore, you can use it as a building block for a data logger), and a DVD player is a device for showing pictures (therefore, you can use it for a fancy video thermometer).

With regard to my website, I selected simple designs because I wanted to encourage people to try hands-on projects. The web doesn't fit complex content, often you land on a webpage following a link or through Google. Reading online is a context-less task, and it's more fatiguing than reading a book, and a completely different experience.

Sometimes readers are fooled by the apparent simplicity of a design. For example, I would classify the "Tiny Planet" GSM I/O port as an "expert" topic, despite the fact that it is built from just a handful of parts.

NAN: You've designed many projects, do you have a favorite?

ALBERTO: My most loved design is a hobby project, the Nutchip (www.nutchip.com). It is a preprogrammed controller implementing a simple state machine (four inputs/outputs), with some handy features like a remote control receiver, a timer, a comparator, and a serial port. I was inspired by the 555 chip, which is both useful and versatile, and wanted to develop a digital counterpart.

I recognized that simple automation tasks (like a burglar alarm or a door opener) would fit simple state machines with truth tables in place of program languages. A specialized editor helps fill a truth table; you can download the table to on-chip flash by means of the PC serial port. At that time, I was studying the fundamentals of human computer interaction (HCI), so I tried to develop a state table editor that was as user-friendly as possible.

The Nutchip concept was original and worked well for hobbyists and schools. It worked so well that some textbooks included a chapter dedicated to Nutchips—and I'm very proud of it. People used Nutchip far beyond my expectations: I even received the schematic of a four-story lift controller built around a single Nutchip!

NAN: *Circuit Cellar* features so many amazing projects because of innovative authors like you. You have had six articles published about your design projects. Several were security-related. In *Circuit Cellar* 167, you described a MC68HC908-based wireless monitoring system. In *Circuit Cellar* 202, you presented the "Witness Camera," which is a self-recording surveillance camera. Do you design security systems for work-related purposes, or are these designs just for your personal use?

ALBERTO: Except for the wireless monitoring system, all the articles I published were derived from personal projects. Complex work projects don't fit well in the space of an article (or webpage), and sometimes customers don't want to have the inner workings of their devices exposed.

All of the articles were derived from contest entries. I like to enter design contests for the competition. Fixed schedules help me stay focused and get things done in time. But I also like the feeling of freedom that allows me to research, reshape the design, or experiment with new ideas as development advances. Different from designs I make for a living, contests have no limits, other than using a specific microcontroller brand. For example, while developing the Witness Camera, I initially didn't plan to use voice menus. Once I started writing the code, I realized it would be simple to play audio files on a speaker. I tested a brute-force solution, bit-banging audio files to the PWM output, and it worked: the combination of a remote control and audio responses was just perfect for the application (and much better than an LCD, as the camera can be purposely concealed).

The discovery process I just described is very valuable, but sometimes you can't exploit it at work. A change like that would have required a change in the specifications from the customer, which is always a long and difficult task in itself.

NAN: One of your latest designs is a programmable touchscreen LCD solution built around an Atmel AVR flash microcontroller. How does the MCU work in the design?

ALBERTO: In the Touch8 LCD solution, the Atmel AT90CAN128 AVR processor does everything: samples the touchscreen, draws the graphics and fonts, handles the event queue, runs the GUI, and even runs the user application.

Yet, more than half of the flash and RAM are available for user code. Touchscreen and timer updates are quick tasks handled by interrupts; and all the user application must do is to poll the GUI from time to time (a few times per second guarantees responsive graphics).

Customers using the Touch8 module are usually surprised by how well an AVR performs redrawing a few windows on the screen, and end up asking, "How can you put all that stuff in just 4 KB of RAM?" But it should not surprise older engineers, who remember windowed GUIs running the Commodore 64 or Sinclair Zx Spectrum.

The real beauty of the Touch8 solution is in its code. I made every effort to make it as simple as possible, hiding complexity inside the GUI library and exposing just a few useful, flexible functions and C structures. It was a challenging task, given the hard constraints of using flash wherever possible to save precious RAM. The demo code clearly shows that it doesn't need to be complicated to add a GUI to even the simplest embedded device.

Another feature I'm excited about (but at first goes almost unnoticed by customers, at least until they try it firsthand) is the capability to compile the code and to test it on a Windows platform. That is, you can run the same code in a window on the PC screen and simulate touches with mouse clicks. Debugging on a PC is much faster and more effective than flashing the code and debugging with an in-circuit emulator (ICE). PC debuggers offer everything an ICE does, plus access to the PC operating system and file services. Think of opening a file and logging an unlimited number of debug messages using just an "`fprintf`" statement: a trivial task on the PC, not-so-easy or impossible on an actual target. For medium-to-complex applications, most of the code can be debugged on the PC alone. For example, in an industrial power analyzer application, I replaced ADC sampling with DDS waveform generator code, which allowed me to test the measurement routines (offset compensation, RMS extraction, power, distortion analysis) with mathematically correct waveforms over a variety of test signals and frequencies. Test waveforms were generated in Excel and stored in comma-separated text files. Once the metering engine was OK, I switched from the PC to the target board, hooking the ADC sampling routines to the real converters. Therefore, debugging on target was limited to verifying that the converters were working correctly, a trivial task taking almost no time and a much simpler signal generator. (For more information and videos about Touch8, visit www.eurekelettronica.it/prodotti_eurek/touch8/intro/touch8.html.)

NAN: Can you describe some recent work-related projects?

ALBERTO: Eurek's market is varied, so is my job. My most recent project was an automatic nail (staple) shooter for frame assembly: it consisted of a Linux touchscreen (ARM9) connected to two I/O boards (STMicroelectonics's STM32 Cortex-M3) and two motor control boards (Freescale DSP) by means of a CAN bus. My job was to develop the firmware for the I/O boards, specify the communication protocol, analyze end-user workflow, and design and write the GUI software.

It was an interesting and challenging project. The hardware wasn't ready when the project started, so I simulated all the parts on the PC before testing the system on real hardware. The GUI runs on Linux, which makes it easy to compile it for the PC and the embedded touchscreen with minor changes.

The I/O boards required more work. I selected the GCC C++ compiler, whose targets include the x86 (PC) or ARM/Cortex processors. An RTOS is useful when dealing with communication protocols, so I included FreeRTOS (because it is available for both PC and Cortex platforms).

On real hardware, the touchscreen application and the I/O boards are connected by means of a CAN bus; but for PC simulation, I replaced CAN bus calls with Posix sockets. In the same way, I wrote a bare-bones simulator for the motor control board, as its firmware was developed by a consultant and not readily available.

I ended up with source code that can be compiled to run on both platforms (Linux and STM32), under conditional compilation. I can't stress enough how useful it is to have the code running on the PC during development and debugging.

Working on the GUI was a completely different task. GUIs are not too difficult to code or to draw: the real challenge is to understand end user needs and workflows, and *not* think as an engineer. Although I've read a lot about HCI—and tried to put into practice the wise words of gurus the caliber of Norman, Cooper, and Nielsen—I often found myself rewriting code because what seemed like a good idea on paper just didn't work well in an interactive environment. With GUIs, every detail matters: size of the icons, colors, placement, even varying the size or orientation of the display (from the computer screen to actual LCD). Each detail can change the user experience substantially.

I was very satisfied with the result. After the first run of machines hit the market, I got a phone call from the customer who told me how surprised he was that no end user called him for the "usual" explanations on how to set up and operate the machine. Hey guys, this is very rewarding!

Other recent projects include: a touchscreen power meter and power factor controllers (64 × 128 BW display, AVR controller, 12 outputs for capacitor banks); a touchscreen data logger/graphic visualizer for photovoltaic panels (128 × 240 BW display, STM32 Cortex-M3 controller, USB, RS-485, flash memory storage for 10 years' worth of data), a touchscreen-operated, boilerless, energy-saving espresso brewing unit (64 × 128 BW display, AVR controller), a drycleaning machine controller (10" 800 × 600 color display, rotary encoder, 64 outputs, 64 inputs on two slave boards connected via CAN), and a general-purpose touchscreen module and GUI library and development environment (64 × 128 BW display, AVR controller, CAN, RS-485, RS-232, I²C).

NAN: What other projects are you currently working on?

ALBERTO: At work, I'm writing the firmware and user interface for a 10″ touchscreen-operated coffee bean roaster. Now you know why Italians make the world's best espresso.

The next project will be a touchscreen, network-enabled dry cleaning machine, equipped with an 80 output /input I/O subsystem by means of a CAN bus.

At home, I'm working on a sous-vide cooking controller (touchscreen, PID-controlled SSR output). Did I mention I love to cook?

NAN: What embedded technologies excite you the most?

ALBERTO: I'm a very strong supporter of 32-bit microcontrollers with megabytes of flash memory, embedded Ethernet/USB/card/serial, and the capability to run high-level languages, RTOSes, and even operating systems.

Don't get me wrong: any engineer worth his salt needs to understand and write some Assembly and be familiar with the underlying hardware (peripherals, buses, clock distribution, interrupt structures). I've done my homework (I once did real-time video generation with carefully crafted Assembly for processors with just 512 words of flash), but times are changing. Nowadays, I'm pleased to work with ARM/Cortex chips and C++.

Commonly available devices, like cell phones, navigators, and players, have raised end-user expectations for all kinds of electronics devices. Once upon a time, a red/green/flashing LED was enough for a heating central, but today end users expect at least an LCD with clear messages, and a event log on a USB (if not a web interface they can browse from a smartphone, another exciting topic in itself).

Webservers and USB stacks are within the capabilities of 32-bitters, but designers need to resort to common libraries to keep code manageable (and not to reinvent the wheel at every design). I can't stress enough how effective it is to have the same code running on both a PC and embedded platform. Not to mention Linux, another topic on its own.

Another technology I'm watching is embedded computer vision. High-end electronics and automotive designers have already triggered the market, as well as the gaming industry (e.g., Kinect, Wii controller). I'm waiting for the right project to give me the excuse to put together a cheap Linux board, a webcam, the OpenCV library, and to start experimenting.

As cameras and processors get cheaper and more powerful, new possibilities open up. A $50 camera and processor kit is great for shape recognition, quality control, level measurement, and for an unlimited number of production processes. A $25 kit is just right for a tool cart that follows you in the workshop (carefully avoiding obstacles) or for a forklift that centers automatically on a pallet. A $10 kit is OK for a point-and-shoot tally counter/ pocket inventory. A $5 kit could be great for a trash can that opens its lid when garbage is coming.

A third intriguing technology is energy harvesting (especially when coupled to a wireless link). New chips can hide the tricky part and make the technology simple, reliable, and ready to be included in many designs. The idea of having a device running forever, making its own energy, is beautiful in itself. I think it's also the pathway to deeply embedded objects, with electronics disappearing inside the things: think of an advanced wireless door knob, for example.

Also, it's not uncommon to have the overall cost of connectors, cables, batteries, battery compartments, and lids exceed the cost of silicon. As break-even approaches, many ideas can become a reality. Wouldn't you love a battery-free garage door opener, weight scale, thermostat, or alarm clock? ■

by George Novacek (Canada)

Electro-Hydraulic Servo Valves

You know that electric actuators are great for positioning mechanical devices, but electro-hydraulic servo valve (EHSV) systems can often do it faster and with a lot more power. This article explains how.

One common task of control systems is to accurately position a mechanical device, be it a part of machinery, a control surface of an aircraft, or an engine fuel injector. The business end of such a system, that is the muscle moving the mechanical bits and pieces, is an actuator.

Today we see many electric actuators, thanks to the recent developments of permanent magnet brushless motors. Great strides have been made in their power compared to weight and cost ratios. Presently, electric actuators can be found in applications traditionally owned by hydraulics, such as jet engine thrust reversers and braking systems. The European "Electric Airplane" initiative is a good example of where the technology is headed. Yet, when it comes to raw power, response time, and weight and cost versus performance ratio, hydraulic systems with typically 3,000 psi (i.e., 204.14 atm – atmosphere) fluid pressure are still hard to beat. Some systems are hybrids in the sense that a local electric pump generates the hydraulic pressure for the actuator.

Hydraulic actuators are simple in principle. Basically, they're bicycle pumps, although their design and manufacturing is quite a different story. But, since we are electrical engineers, we don't need to worry about the gory details of mechanical engineering. To use such actuators, we only need to understand

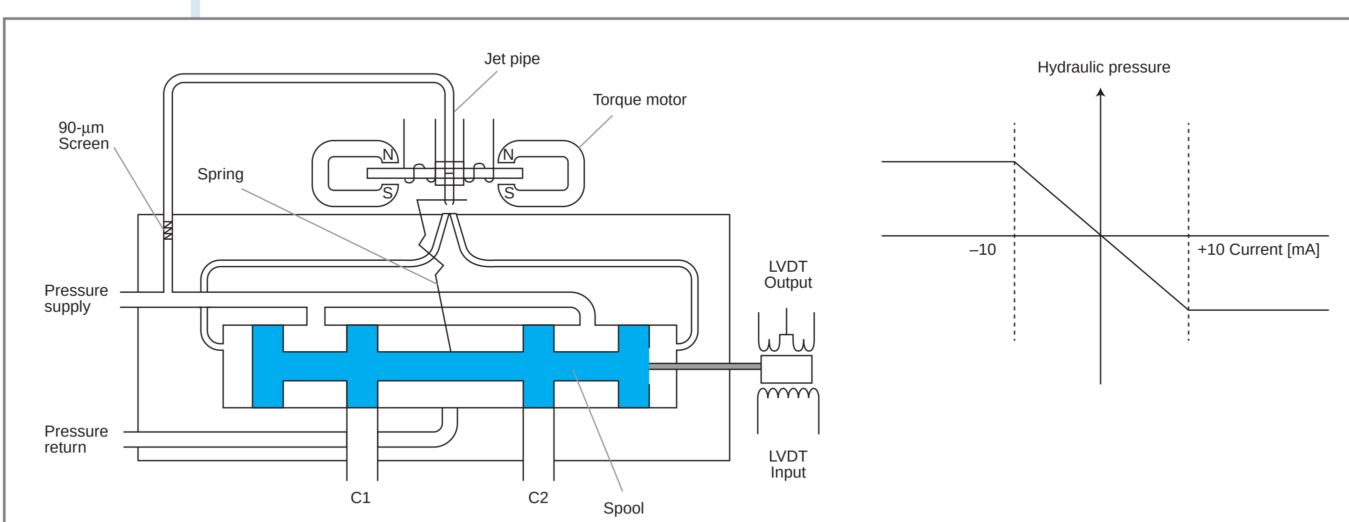


Figure 1—An electro-hydraulic servo valve. As the current changes within ±10 mA, the pressure flow changes linearly until saturation.

how they work and how they interface with electronic controllers.

HYDRAULIC ACTUATORS & EHSVs

That enormous force can be generated by hydraulic actuators, where the output force is the result of the input pressure multiplied by the ratio of the output cylinder area over the input orifice area, has been known since antiquity. The main issue you face when an electro-hydraulic system is being designed is how to control the flow of the pressurized fluid into the actuator.

In its simplest form, a solenoid-actuated valve is opened or closed to control the flow. This forms a so-called "bang-bang" control system, which is easy to understand and build. It has only two stable states, but the best we can achieve in terms of control precision is the output variable to oscillate around the set, that is the commanded variable. It is possible to pulse-width modulate (PWM) the valve, as some systems do, to achieve linear control. That means opening and closing the valve for varied periods of time. Unfortunately, a typical reaction time to open or close a common solenoid valve is around 30 ms. This severely limits the bandwidth of the control loop and is, therefore, only suitable for a limited number of applications.

Enter the electro-hydraulic servo valve (EHSV). It is a hydraulic-forward amplifier with power gains of 10^4 to 10^6, enabling precise and fast motion control with only modest space and weight demands. Its control input is typically ±10-mA current to control, in linear fashion, the hydraulic fluid flow to the actuator. While there are many variants of this valve, the fundamental principle remains the same (see Figure 1).

TORQUE MOTOR

The electrical interface part of the valve is formed by a subassembly called a torque motor. Traditionally, an eletromagnet has been used, but other movements, such as piezoelectric, have been developed. Torque motors often comprise two coils for redundancy, each typically 1 kΩ, and each capable of moving the jet pipe 100% of its travel. Often both coils are connected in parallel, such that a failure of one still enables full operation of the EHSV, but they can also be separately driven. A simplified current driver schematic is shown in Figure 2.

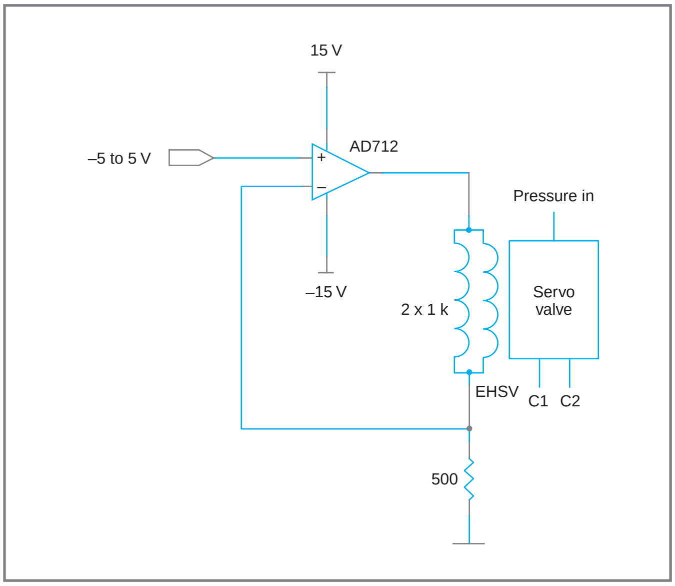


Figure 2—EHSV current driver

The torque motor moves a jet, connected through a small diameter jet pipe to a hydraulic source, directing the hydraulic fluid stream to either orifice leading to each end of the spool cavity, respectively. In Figure 1, the spool body is shaded to make surrounding cavities stand out. The directed stream pushes the spool left or right, thus opening and closing orifices C1 and C2. The spool cavity receives full hydraulic pressure from the hydraulic supply and as the spool moves, the flow is directed to either C1 or C2, delivering the pressurized fluid to the actuator. With the spool centered, no pressure flows to the actuator and the fluid is returned to the supply through the pressure return. The spring provides a feedback and holds the spool in the center when there is no current flowing through the torque motor.

The graph in Figure 1 shows the flow in C1 and C2 as a function of the torque-motor current. At typically ±10-mA drive the spool completely opens the output orifice, C1 or C2 respectively, and the pressure saturates.

LVDTs

In safety-critical systems a linear variable differential transformer (LVDT) may be connected to the spool, as shown, to monitor its excursions. EHSVs are precision instruments. We know how much the spool must move for a given torque-motor current. This movement is measured by the LVDT and, in the controller, compared with the command current. A discrepancy between the two variables indicates a fault.

Why monitor the spool and not the actuator position? Impurities in the hydraulic fluid, despite its being filtered, may cause the EHSV to jam. The LVDT thus provides predictive fault detection which enables the system to take quick evasive action, such as de-energizing the main valve supplying the hydraulic power to the system. I have seen requirements as short as 60 ms to shut off the pressure in case of a fault. This is no small task for fault verification and solenoid valves handling 3,000 psi.

Today, due to cost pressures, many EHSVs lack the LVDT monitor and their fault detection relies on monitoring the movement of the actuator. This has some shortcomings. First, the signal-to-monitor is a derivative of the position feedback, a high-pass filter in fact, that amplifies the system noise, usually the result of external perturbations. This calls for a compromise, often resulting in long fault detection time. Second, imagine the hydraulically controlled part stuck or slowed down by some excessive load. That is often the time when the hydraulic pressure is needed to overcome the load and move the part, but without the LVDT monitor, the slow or nonresponsive actuator would indicate a fault and the pressure would be disconnected. To avoid such situations a thorough system analysis must be performed before the architecture can be finalized. At a later date, if you were to discover that without spool monitoring the customer specifications cannot be met, you may encounter an expensive modification, indeed.

DETAILS & DESIGN

In addition to various types of torque motors, EHSVs are also constructed as closed-loop controllers in their own right, some having several stages (the first spool controls a second, much larger spool) for large gain and precise characteristics of the hydraulic flow control. They can also be biased to provide nonsymmetrical pressure. And they are fast: a 1-kHz bandwidth is not unusual. The construction details may remain transparent, but you should understand how the hydraulics work in principle. To design the electronic controller, all you need to know are the type of signals, the type of position feedback sensors, the EHSV's control current range, and the system's required bandwidth. Mechanical engineers should define all of that for you.

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 of embedded control systems for aerospace applications. George wrote 26 feature articles for Circuit Cellar *between 1999 and 2004.*

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:

—

Wireless Module Control
An MCU-Based Irrigation Control System
by Tom Kibalo
***Circuit Cellar* 223, 2009**

Tom describes a PIC-based prototype design for a wireless water control system for EarthBoxes, which are gardening boxes with water reservoirs and fertilizer strips for cultivating plants and vegetables. The scalable design can accommodate various multiple-box configurations. Topics: Valve, Irrigation, RF, Sensor

—

Inertial Rolling Robot
by Jeff Bingham and Lee Magnusson
***Circuit Cellar* 200, 2007**

Jeff and Lee's H8/3664-based rolling robot is capable of inertial movement. A DC electric motor is attached to a pendulum and suspended inside an inflated ball, which provides the driving force. Topics: Servo, Robot

—

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

by Bruce Land (USA)

Stochastic Chemical Kinetics on an FPGA

You can use an FPGA to accelerate chemical reaction modeling. Here you learn how Verilog code enables you to compute chemical kinetics solutions in parallel on an FPGA. The parallel hardware runs simulations faster than similar simulations on a PC.

This article describes how hardware used on a field-programmable gate array (FPGA) can speed up the modeling of some reactions. The topics of stochastic algorithms and MATLAB stochastic simulation are also covered.

As you read this, there are thousands of chemical reactions going on in your body. Some are very fast. For instance, the binding of neurotransmitters in your brain occurs on a time scale of microseconds, while protein production has a time scale of seconds or minutes. Understanding physiology requires understanding how chemicals react and how fast each reaction is. Chemical kinetics is the name given to the study of reaction speed. But why would an article on chemical kinetics appear in this electronics engineering magazine? The short explanation is that hardware instantiated on an FPGA can accelerate the modeling of some reactions and is an interesting example of parallel computation. The details follow.

CHEMICAL REACTION

To make things specific, let's consider one reaction, the breakdown of starch into sugar by the enzyme amylase. Amylase is present in saliva and is the reason that rice or potato becomes sweeter as you chew, as the starch is converted to sugar. An enzyme is a chemical that accelerates the reaction rate of another chemical, so you can immediately see that this is a nontrivial (and nonlinear) example. The reaction is often represented as:

$$\text{amylase} + \text{starch} \leftrightarrow (\text{amylase-bound-to-starch}) \rightarrow \text{amylase} + \text{sugars}$$

This should be read as follows: Amylase and starch in a water solution react to form a loosely bound entity that breaks the starch apart into sugars. The sugars then fall off the amylase, so that the amylase is free to react with more starch. The bidirectional arrow representing the binding also suggests that the combination of amylase-bound-to-starch can also fall apart without conversion to sugar. However, the second reaction, where the bound combination falls apart to the sugar products, never reverses. This formulation is known as a Michaelis-Menten description of the reaction.

There are three reaction rates associated with this reaction: the binding rate of amylase to starch, the unbinding rate of the combination back to amylase and starch, and the reaction rate to sugar. Let us now consider what determines the reaction rates. Since the chemicals are in a water solution, the individual molecules are dispersed and have to physically collide with each other to react. The rate at which they collide is partly determined by how many there are in a given volume. Doubling the concentration doubles the chances of collision and therefore the reaction rate. The rate is also sensitive to all kinds of influences (e.g., temperature and pH), but we will model all those effects as a *rate constant* independent of the concentration of the chemicals.

Now we need a bit of notation. We will use square brackets to mean *concentration of x*, written as [x]. We will call the rate constant of the binding reaction k1, the unbinding rate constant k2, and the conversion rate constant k3. The proportionality of reaction rate with concentration implies that for the first reaction the rate is k1 × [amylase] × [starch]. The classic way of solving this system is to convert the rates to differential equations, then solve the set of differential equations. Quite often, such as in this example, we cannot solve the equations analytically, but need to simulate the solutions. For these reactions we can write:

$$\frac{d\,[\text{amylase}]}{dt} = -k1 \times [\text{amylase}] \times [\text{starch}] + (k2 + k3) \times [\text{amylase-bound-to-starch}]$$

$$\frac{d\,[\text{sugars}]}{dt} = k3 \times [\text{amylase-bound-to-starch}]$$

Hidden in these equations is the assumption of differential smoothness in concentration. For large numbers of molecules (e.g., the number of water molecules in a teaspoon), differential smoothness makes sense because removing one molecule is completely undetectable. But, what if the volume of interest is a single cell that might have only a few copies of a molecule in a tiny volume? The concentration is still high, so the reaction goes quickly, but the concentration can only change in units of one molecule and cannot change smoothly, so the differential equation approach fails. Instead, you have to treat each molecule as having some probability of reacting. This finally leads us back to the topic of this article: using an FPGA to accelerate probabilistic simulations of chemical reactions by counting reaction events based on random number generation.

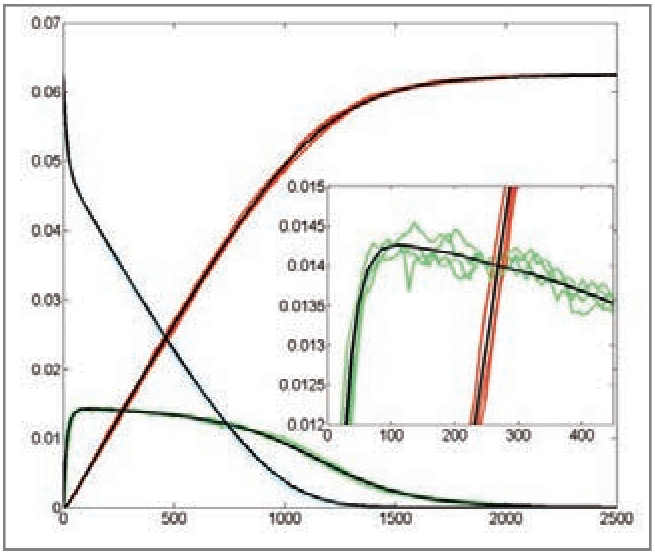


Figure 1—Michaelis-Menten kinetics were used to test the stochastic solver. The blue curve is the substrate number, the red curve is the product, and the lower curve is the concentration of bound enzyme. The black lines are the values computed by a MATLAB differential equation solver. The inset shows the stochastic variability near the bound enzyme peak.

THE ALGORITHM

The approach we will take is to step back to a more fundamental level than differential equations and treat each potential reaction event as a random occurrence, biased by chemical concentration and rate constants. But, to emphasize that chemicals are discrete molecules, we will replace concentrations by *number* of molecules, with the notation Nmolecule. During any small interval of time we can calculate the probability of the reaction occurring as the product of rate constants and number of molecules, just as before. So we *could* map reaction calculations to parallel multipliers on the FPGA, then ask if the product, say, k1 × Namylase × Nstarch, is greater than a uniform random number and allow one reaction to occur if it is. However, hardware multipliers tend to be limited in number on FPGAs, relative to general logic elements, so we need to simplify the calculation. As explained in Lukasz Salwinski and David Eisenberg's 2004 *Nature Biotechnology* article, "In Silico Simulation of Biological Network Dynamics," if we compare each number of molecules (and rate constant) to a uniformly distributed random number then the probability of (Nchemical>random-number) is proportional to number (or rate constant).[1] Since each chemical number, rate constant, and random number is independent, the probability that the product of the concentrations and rate constants will be above some value is equal to the probability that:

```
(k1>rand1) && (Namylase>rand2)
  && (Nstarch>rand3)
```

where && is the usual logical *and* operation. Thus, we have replaced expensive multiplies with cheap, fast, random number generation and logical operations. Each term of the differential equation is replaced with these stochastic operations and the left-hand side rate term is replaced with an increment/decrement or no change operation.

We still need to generate random numbers without using multipliers. It is possible to generate high-quality pseudorandom numbers using a fairly long shift register with XOR feedback from the appropriate stages to stage 1. You can find out how to do this by searching online for "linear feedback shift register." In a 1983 paper in the *Journal of Computational Physics,* A. Hoogland, et al. show how to construct a high-quality 16-bit or 32-bit pseudorandom number in one or two shift cycles by folding a long (127-bit) shift register into 16 sections and shifting all of them at once.[2] The Verilog code provides details.

Some care has to be taken so that the actual reaction rates are not too large or too small. If they are too large, more than one or two reactions may happen per time step, which makes the process of computing the probabilities harder. If they are too small, the simulation takes too long. I modified Salwinski and Eisenberg's system to enable up to two reactions at each time step, whereas their system used only one. Limiting the actual reaction rate to an average probability of no more than 0.085 per step keeps the error rate due to missed events below 0.01% (based on a Poisson distribution). If you can relax the missed events to 0.1%, a probability of 0.15 may be used. Computing the actual reaction rate means multiplying out the rate constant and the one or two chemical concentrations (as a fraction of 2^{16}, since I used 16-bit concentrations). For instance, a first-order reaction with rate constant 4,096 and concentration 8,192 would have a reaction probability of:

$$\left(\frac{4{,}096}{2^{16}}\right) \times \left(\frac{8{,}192}{2^{16}}\right) \text{ or } \left(\frac{1}{16}\right) \times \left(\frac{1}{8}\right) = 0.078$$

HARDWARE ORGANIZATION

The hardware on the FPGA is organized into several modules. System control is a state machine that sequences through eight states. In the first state, all random numbers are generated and reaction results are computed. In the next seven states, the reaction results are added/subtracted to the various chemical number counters. The logical result involves cycling between computing reactions, then updating chemical numbers on each time step. There is a chemical module instantiated for each different chemical and a reaction module for each different reaction path. All the reaction and chemical modules compute their contribution to the current time step in eight clock cycles, independent of the number of chemicals or reactions.

The code fragment in Listing 1 shows the structure of the enzyme reaction when it is converted into Verilog code. There is a module defined for each chemical and one defined for each reaction. Each chemical is defined by one hardware module that outputs the current number of molecules for the chemical. Inputs include an initial concentration, slots for up to six increment/decrement commands (from reaction modules), the reaction clock, the reaction state, and a reset command. Internally, the chemical module is a state machine that uses the increment/decrement commands to compute the updated number of molecules at each time step.

Each reaction is defined by one hardware module that outputs increment/decrement commands to feed back to chemical modules. Inputs are the number of molecules of one or two chemicals, a rate constant, the reaction clock, the reaction state, a random number seed, and a reset command. The random number seed should be distinct for each different reaction module. Internally, the reaction module computes six random numbers in parallel (three for each of two possible reactions at each time step), compares them to the molecular numbers and rate constant, then determines if zero, one, or two reactions actually occur by performing the previously described logical and operation.

I wanted to be able to visualize the reactions, so I wrote a time series VGA display module that takes three different data inputs and plots them as they are calculated in three different colors. The VGA interface uses dual-ported memory on the FPGA as display memory and operates completely in parallel with the reaction state machine. The screen refresh side of the VGA controller reads from memory (using Altera M4K memory blocks) when it needs to draw the screen, while the data formatting side of the controller writes to display memory in a state machine. Since it takes a lot of small time steps to compute some reactions, not every reaction time step is plotted on the VGA. The state machine handles the display time increment.

PERFORMANCE REVIEW

The reaction simulation results from the FPGA were compared to the same stochastic algorithm coded in MATLAB and to a differential equation formulation coded in MATLAB. (Refer to Alexander van Oudenaarden, "Systems Microbiology: A living cell as a well-stirred bioreactor," MIT 2009.) Two chemical systems tested were the

Listing 1—This Verilog code defines the reaction for simulating an enzymatic reaction. Three chemicals and three reactions are defined. The full Verilog code contains the *chemical* and *reaction* modules.

```
// Michaelis and Menten ////////////////////////////////
/////////////////////////////////////////////////////
// define A + E <-> AE -> S + E  (enzyme reaction)
/////////////////////////////////////////////////////
wire [15:0] A, S, AE, E ; // concentrations
// concentration inc/dec from reactions
wire [2:0] AtoAE_inc, AtoAE_dec,
       AEtoA_inc, AEtoA_dec,
       AEtoS_inc, AEtoS_dec;

// Handy constants used for unused inputs to modules
parameter no_chem = 16'hffff, no_inc = 3'b000 ;

// Read this as:
// For chemical A the initial condition is 240 molecules.
// When A is converted to AE, decrement A.
// When AE is converted back to A, increment A.
// Four increment inputs are not used.
// All chemical modules need the state variable, reaction clock and reset.
chemical chem_A( A, 16'd240,
                      AtoAE_dec, AEtoA_inc,
                      no_inc, no_inc,
                      no_inc, no_inc,
                      state, reaction_clock, reset);

chemical chem_S( S, 16'h0000,
                      AEtoS_inc, no_inc,
                      no_inc, no_inc,
                      no_inc, no_inc,
                      state, reaction_clock, reset);

chemical chem_E( E, 16'd60,
                      AtoAE_dec, AEtoA_inc, AEtoS_inc, no_inc,
                      no_inc, no_inc,
                      state, reaction_clock, reset);

chemical chem_AE( AE, 16'h0000,
                      AtoAE_inc, AEtoA_dec, AEtoS_dec, no_inc,
                      no_inc, no_inc,
                      state, reaction_clock, reset);

// define the forward and backward reactions.
// inc/dec output signals are nonzero  if the reaction occurs.
// unused concentration inputs should be set to no_chem=16'hffff.
// Read this as:
// If the reaction of A+E to AE occurs, set the inc/dec lines to nonzero.
// The input chemicals are A and E, with rate constant 16'hffff
// All reaction modules require state, clock, reset and a unique seed.
reaction AtoAE(AtoAE_inc, AtoAE_dec, A, E, 16'hffff,
                      state, reaction_clock, reset,
128'haaaaaaaa54555555+seed_offset);

reaction AEtoA(AEtoA_inc, AEtoA_dec, AE, no_chem, 16'h0010,
                      state, reaction_clock, reset,
128'haaaaaaaa55555555+seed_offset);

reaction AEtoS(AEtoS_inc, AEtoS_dec, AE, no_chem, 16'd256,
                      state, reaction_clock, reset,
128'haaaaaaaa53555555+seed_offset);
```

Michaelis-Menten formulation of enzymatic action explained earlier and a nonlinear oscillator, known as the Oregonator model, because it was originally developed in 1974 at the University of Oregon by Richard Field and Richard Noyes.[3]

The Michaelis-Menten formulation is a good test case because it is familiar to all biochemists and includes a nonlinear term. Figure 1 shows a typical result comparing the FPGA hardware with MATLAB differential equation code. Since the numbers of molecules are fairly large in this example (4,096 substrate, 1,024 enzyme), we expect the FPGA stochastic simulation hardware to closely approximate the differential equation solution because the stochastic RMS variation is proportional to the square root of the number of molecules. So the RMS variation should be around:

$$\frac{\sqrt{1,024}}{1,024}$$

or about 3%. The inset suggests a few percent RMS variation near the peak of the bound enzyme. The close correspondence between the MATLAB and FPGA results suggests that the FPGA parallel design is correct.

Figure 2 shows the effect of dropping the number of molecules down to only 60 enzyme molecules. At the peak, only about 25 enzyme molecules are bound and the statistical fluctuations are:

$$\frac{\sqrt{25}}{25}$$

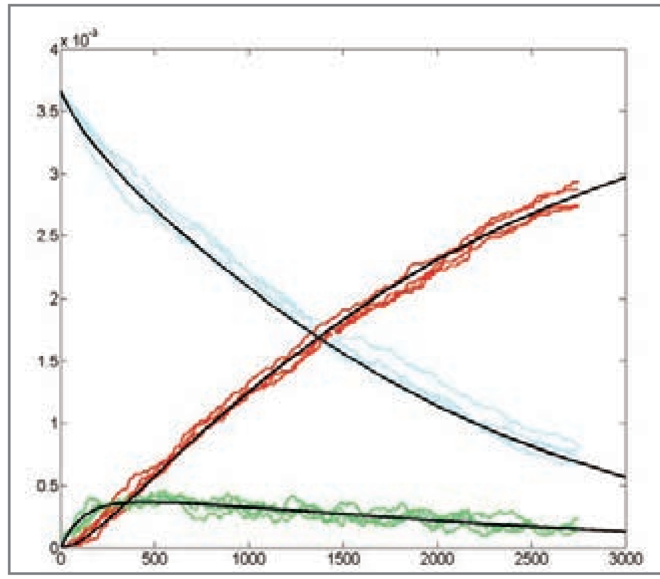


Figure 2—Michaelis-Menten kinetics at lower concentration. The blue curve is the substrate number, the red curve is the product, and the green curve is the concentration of bound enzyme. The black lines are the values computed by a MATLAB differential equation solver. The stochastic variation is quite noticeable, and in some ways is a better representation of what goes on in individual biological cells.

or about 20%, which is clearly visible. In this case, the averaging properties of the differential equation become obvious. In real cells, the number of enzyme molecules is

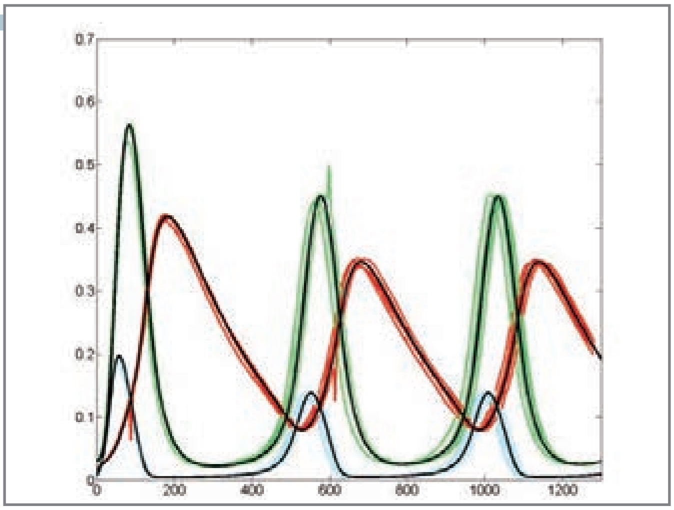


Figure 3—Oregonator kinetics calculated by MATLAB ODE solver (black lines) and the FPGA stochastic solver. There are three communication glitches on the traces. Blue is Y1, red is Y2, and green is Y3.

often quite small, and the reality of stochastic variation directly affects cell operation and is more "real" than the differential equation average.

The Oregonator is a good test case because it is a classic example of a stiff system. A stiff system is one in which there is a large range of characteristic reaction rates. This feature makes the system harder to solve with a differential equation solver, but it works well with a stochastic solver. This system was used by D. Gillespie as a test case for an exact stochastic simulator method.[4] The reaction scheme is:

$$X1 + Y2 \rightarrow Y1 \text{ (rate constant c1)}$$
$$Y1 + Y2 \rightarrow Z1 \text{ (rate constant c2)}$$
$$X2 + Y1 \rightarrow 2Y1 + Y3 \text{ (rate constant c3)}$$
$$Y1 + Y1 \rightarrow Z2 \text{ (rate constant c4)}$$
$$X3 + Y1 \rightarrow Y2 \text{ (rate constant c5)}$$

The Xs represent large pools of chemicals and do not change concentration during simulation, so they are just constants. The Zs are reaction products that are not reused and therefore do not need to be modeled. The three Y molecule numbers are shown in Figure 3. The results of seven stochastic runs are plotted. You can see that there is considerable variability, centering on the differential equation solution (black lines).

FAST COMPUTATION

Even though the clock rate of the FPGA is rather low compared to a PC, the computational rate of the FPGA stochastic simulation is faster than the PC because so many operations can be carried out in parallel. In the case of the Oregonator simulation, 30 16-bit random numbers are computed in one cycle to support computation of the reaction and three chemical concentrations are updated in seven cycles. The update rate is independent of the number of chemicals or reactions (up to the size limit of the FPGA), so bigger models show more speedup over the PC solution. The MATLAB stochastic simulator I wrote took 870 s to run on my desk machine (a 3.2-GHz core duo with 8-GB memory) and 8 s to run on the FPGA (reaction clock set at 25 MHz), a factor of 100 speedup. The Oregonator model (with serial readout and VGA display) uses about 15% of the Cyclone II FPGA on an Altera DE2 development and education board. The VGA alone requires about 2% and the serial readout module is of negligible size.

Author's note: For additional information about stochastic chemical reaction simulation and code, visit the Cornell University ECE 5760 course website http://people.ece.cornell.edu/land/courses/ece5760/Chemical_Simulation.

Bruce Land (bruce.land@cornell.edu) is a senior lecturer in Electronics and Computer Engineering at Cornell University. He has a BS in Physics from Harvey Mudd College and a PhD in Neurobiology from Cornell University. This year, Bruce is teaching three courses on the following topics: microcontrollers as components in electronic designs, designing FPGA circuits for embedded applications, and electronic bio-instrumentation. For more information, visit http://people.ece.cornell.edu/land.

REFERENCES

[1] L. Salwinski and D. Eisenberg, "In Silico Simulation of Biological Network Dynamics," *Nature Biotechnology*, 2004.

[2] A. Hoogland, J. Spaa, B. Selman, and A. Compagner, "A Special-Purpose Processor for the Monte Carlo Simulation of Ising Spin Systems," *Journal of Computational Physics*, Volume 51, Issue 2, August 1983.

[3] R. Field and R. Noyes, "Oscillations in Chemical Systems IV. Limit Cycle Behavior in a Model of a Real Chemical Reaction," *Journal of Chemical Physics*, 1974.

[4] D. Gillespie, "Exact Stochastic Simulation of Coupled Chemical Reactions," *Journal of Physical Chemistry*, 1977.

RESOURCES

Cornell University, School of Electrical and Computer Engineering, "ECE 5760: Stochastic Chemical Reaction Simulation," http://people.ece.cornell.edu/land/courses/ece5760/Chemical_Simulation.

A. van Oudenaarden, "Systems Microbiology: A Living Cell as a Well-Stirred Bioreactor," MIT, September 2009,http://web.mit.edu/biophysics/sbio/PDFs/L2_notes.pdf.

SOURCE

Cyclone II FPGA, M4K memory blocks, and DE2 development board
Altera Corp. | www.altera.com

by Jesse Marroquin (USA)

Find a New Direction

A Low-Power Digital Compass

You can use a low-power microcontroller to build a digital compass. This article details the planning, design, and programming processes.

The magnetic compass has changed very little throughout its history, but recent developments enable you to readily implement an electronic compass with no moving parts. The typical three-chip system described in this article displays the most recent heading on an LCD, and can be used as a basic portable navigation device (see Photo 1).

Several low-power microcontrollers are available to meet the requirements of this system, but for the one shown (a Maxim Integrated Products MAXQ610 microcontroller), low power consumption and a wide operating voltage range (1.7 to 3.6 V) make it a good fit for battery-powered applications. Its 64 KB of flash-memory based program storage and 2 KB of data RAM easily support applications that are written in high-level languages, such as C. Low active-mode current for the device is just 1.4 mA (typical) at 1 MHz and 3.5 mA at 12 MHz. Its ultra-low-power stop-mode current (200 nA typical) and wake-up timer (based on an 8-kHz nanopower ring oscillator) enable this microcontroller to consume minimal power while in Stop mode.

A SparkFun Electronics HMC6352 digital compass module senses the magnetic field and calculates the heading (see Figure 1). Its internal sensing element consists of two orthogonally oriented magneto-resistive sensors that sense the horizontal components of the earth's magnetic field. In addition to the sensors, it includes signal conditioning circuitry and a microprocessor. The HMC6352 operates over a wide voltage range and requires only an external bypass capacitor. It communicates with the microcontroller over an I^2C-bus interface.

The display subsystem consists of a universal NXP Semiconductors PCF8576C LCD driver and an eight-character, 14-segment Varitronix VIM-878-DP LCD. When generating the drive signals for static or multiplexed LCD panels, the driver handles up to four backplanes and up to 40 segments. It also communicates via an I^2C-bus interface, operates over a wide voltage range, and includes display RAM and circuitry for generating the LCD bias voltage. The VIM-878-DP panel's 14 segments per character can display both alpha and numeric symbols.

PROTOTYPE

The hardware prototype includes a Maxim demo board for the MAXQ610 microcontroller, a custom board for

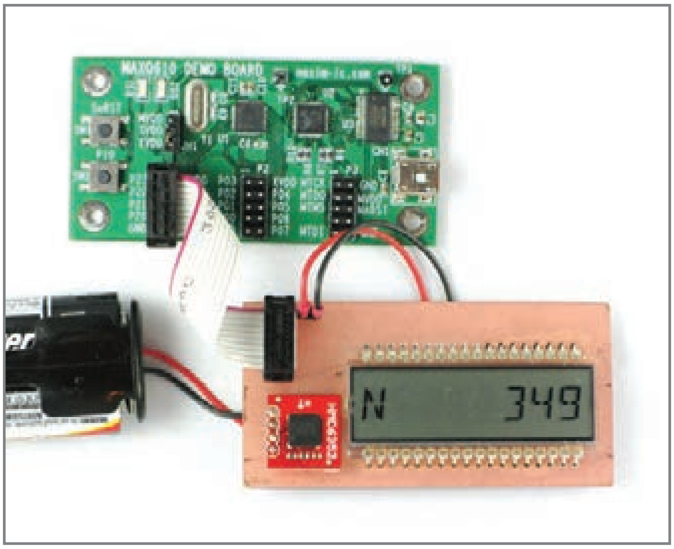


Photo 1—A Maxim MAXQ610 microcontroller, a custom board containing the magnetic sensor, an LCD, and a display driver provide all of the necessary functions for a digital compass.

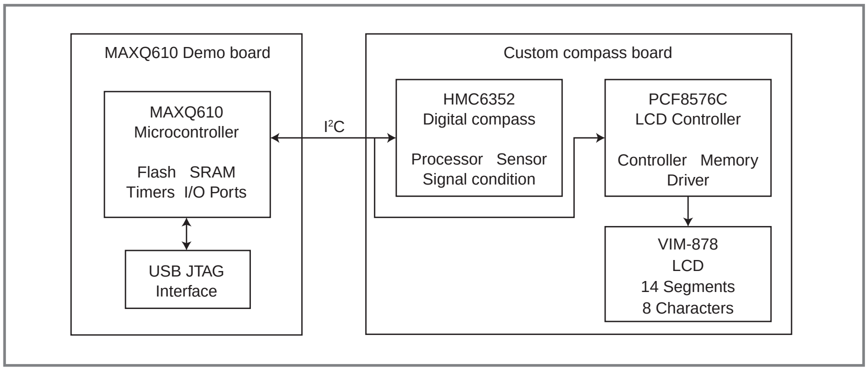


Figure 1—The basic electronic compass system consists of a Maxim Integrated Products microcontroller (MAXQ610), a magnetic sensor, a SparkFun Electronics digital compass module (HMC6352), and a NXP Semiconductors LCD driver (PCF8576C) coupled to a Varitronix LCD panel (VIM-878-DP).

the magnetic sensor, an LCD driver, and a display driver. The demo board includes two headers for access to I/O ports, two LEDs, a push button, and a USB-debugging interface. The interface is compatible with the Maxim MAX-IDE and IAR Embedded Workbench development tools. In selecting hardware for this project, the high priorities were low active power and, more importantly, low standby power.

The HMC6352 is a complete magnetic compass in a single multichip LCC package that requires only a single bypass capacitor. The module operates over a voltage range of 2.7 to 5.2 V, with a typical 1-mA steady-state current at 3 V, and a typical 1-µA sleep mode current at 3 V. The LCD controller-driver and VIM-878-DP display provide a low-power, high-contrast visual interface. This driver-display combination operates with minimal external components and communicates over an I²C bus. The LCD controller has an operating voltage range of 2 to 6 V, with a maximum current of 60 µA at 3.5 V in low-power mode.

CONNECTION, POWER & CONTROL

Connecting the compass module and LCD controller to the microcontroller could not be simpler because both peripherals include an I²C-bus interface. The MAXQ610 board does not include an I²C interface, but you can easily implement an I²C communications block in the software. The block then executes I²C communications by directly driving the microcontroller's port pins.

Push button SW2 on the demo board enables you to begin the calibration mode. The push button is connected to a port pin that provides edge-triggered interrupts and an internal pull-up to VDD. Finally, two AA batteries in series provide power for the compass. A jumper on the demo board enables you to isolate the microcontroller power-supply pins from the debug interface so that the interface can remain unpowered when the system is deployed.

The firmware controlling the compass was written in C, using IAR Embedded Workbench. The basic program flow starts after a power-on-reset initializes the system, causing program execution to settle into an infinite loop. Code within this loop periodically awakens the compass module and starts a heading measurement, which takes a maximum of 6 ms to complete. When the measurement is complete, code within the main loop reads the heading result and puts the compass module back to sleep. The heading results are then formatted and sent to the LCD controller for display. With the measurement done and the new heading displayed, software instructs the microcontroller to go back into its Stop mode.

Either the wake-up timer or the push button external interrupt can bring the microcontroller out of Stop mode. Once out of Stop mode, the firmware determines whether the push button or the wake-up timer was responsible for restarting execution. If the push button was pressed, the system immediately debounces the push button input and enters the calibration mode. Otherwise, execution jumps to the top of the infinite loop. The compass is calibrated by rotating the module one or more times in a 20-s period, and the display flashes "CAL" while the Calibration mode is active. When calibration is complete, the display changes back to its Heading and Degrees mode.

The microcontroller includes a wake-up timer that triggers an interrupt after a certain number of cycles from the 8-kHz ring oscillator (set by the user). The oscillator continues to operate while the microcontroller is in Stop mode, and if the wake-up interrupt is enabled, the microcontroller generates an interrupt when the counter value becomes zero. The wake-up timer has a 16-bit count-down value, so the maximum wake-up period is a little more than 8 s (i.e., $(2^{16} - 1)/8$ kHz). In this application, a value of 12,000 cycles yields a wake-up period of 1.5 s.

HARD & SOFT INTERRUPTS

The push button on the MAXQ610 evaluation board (SW2) connects between ground and port 1 pin 0. The port pins of the MAXQ610 provide an optional weak pull-up to VDD when operating in Input mode and the pins of ports one and three can also function as external interrupts. This design uses both of these features, with port 1 pin 0 configured as an input. An internal pull-up, easily shorted to ground when

Interrupt	Vector address	#pragma vector=n
Power fail	0x20	0
Memory fault	0x28	1
External 7–0	0x30	2
IR Timer	0x38	3
Serial port 1–0	0x40	4
SPI/External 15–8	0x48	5
TimerB 1–0	0x50	6
Wake-up/watchdog	0x58	7

Table 1—Interrupt vector addresses and `pragma` assignments for Maxim Integrated Products's MAXQ610.

the button is pushed, keeps the input high, otherwise. The interrupt for this pin is configured to trigger on the falling edge of the input. Switch debouncing is implemented as a loop that requires the input to remain at a steady state for 200 ms. A directional reading cycle is started by a soft interrupt generated by the internal wake-up timer.

In the MAXQ610, an interrupt vector table (IVT) contains entries that combine the interrupts from several sources, thereby simplifying the job of identifying the source of the interrupt. The MAXQ610 is a relatively new device, and documentation for the IAR C compiler has not been updated to correct the setup for the MAXQ610 IVT. Table 1 shows the MAXQ610 interrupt vector table entries.

To identify a C routine as an interrupt vector, the IAR compiler requires a `pragma` directive and the `__interrupt` keyword. C functions for the external zero and wake-up timer interrupts are shown in Listing 1. The `#pragma vector=n` in Listing 1 directives instruct the compiler and linker to generate the appropriate interrupt vector table entries for functions that follow the directives. In this example, `pragma` instructs the compiler to place a jump instruction to the `pushButtonISR` function at code address 0x0030. Similarly, the compiler places a jump instruction to the `wakeupISR` function at code address 0x0058. The `__interrupt` keyword marks the function as an interrupt routine and also instructs the compiler to use a return-from-interrupt instruction instead of a simple return when the function exits. Other compiler-specific code may be generated with the use of this keyword.

Listing 1—A C routine for the external zero (reset) and wake-up timer interrupts

```
#pragma vector=2
__interrupt void pushButtonISR(void) {
   /* Clear interrupt flag */
   EIFO_bit.IE0 = 0;
   /* Signal main that button has been pushed */
   calibrate = 1;
}

   #pragma vector=7
   __interrupt void wakeupISR(void) {
   /*
     The wake-up timer is reloaded and the
     interrupt cleared by the following sequence.
   */
   WUTC.WTE = 0;
   WUTC.WTE = 1;
}
```

I²C BUS EMULATION

The I²C-bus master is implemented in firmware and supports clock stretching for slow devices. The bus

Listing 2—A short C routine reads direction data for the compass over the I^{2}C interface.

```
unsigned int hmcReadData(void){
  unsigned int val = 0;

  /* Generate start condition */
  i2cStart();
  /* Send the device address */
  i2cTxByte(HMC_READ);
  /* Get the MS byte of the heading result, ACKing the byte. */
  val = i2cRxByte(0) << 8;
  /* Get the LS byte of the heading result, NACKing the byte. */
  val += i2cRxByte(1);
  /* Generate stop condition */
  i2cStop();

  return val;
}
```

master is built by combining the two functions `i2cTxBit` and `i2cRxBit`. The first (`i2cTxBit`) clocks a bit onto the bus, and the second (`i2cRxBit`) clocks a bit in from the bus. These two functions support clock stretching for slow peripherals. Using these bit functions, the `i2cTxByte` and `i2cRxByte` functions transmit and receive bytes on the bus. The function `i2cRxByte` takes an argument that is the master's ACK/NACK response to the byte being received. Additional functions `i2cStart` and `i2cStop` generate the start and stop bus conditions. The function `hmcReadData` illustrates use of the I^{2}C interface. A short C routine reads direction data for the compass over the I^{2}C interface (see Listing 2).

THE HMC6352 COMPASS

ASCII commands sent over the I^{2}C bus enable you to talk to the HMC6352. Table 2 is a full list of the commands supported. The device in this article is configured for Standby mode (the default configuration). In Standby mode, the device waits for commands from the master before performing a measurement. Other modes of operation include Query mode, in which a new measurement is started after the previous measurement results are read out, and Continuous mode, in which the device performs continuous measurements at a programmable rate. Firmware places the device in Sleep mode when it is not in calibration mode or calculating a new heading.

To get a new heading value, the firmware wakes the compass with a "wake up" command (W), and then puts the MAXQ610 in Sleep mode for at least 100 µs. When the compass is awake, the firmware sends a "get data" command (A), and once again puts the MAXQ610 in Sleep mode. After 6 ms, the compass is ready to deliver the heading results: 16-bit binary values, read out with the most significant byte first. The heading value is in tenths of a degree, with a range of 0 to 3,599. After the new heading is retrieved, an S command puts the compass back to sleep.

Calibration is accomplished in much the same way as getting a new heading. The MAXQ610 awakens the compass with the W command, and then starts the calibration process by issuing the Calibrate command (C). Then, the CPU sleeps for 20 s while the compass performs its internal calibration routine. After this sleep period, the CPU issues the "exit calibration" command (E) and finally puts the compass back to sleep with the S command.

THE DISPLAY CONTROLLER

The PCF8576C LCD driver is paired with a VIM-878-DP LCD. Together these devices provide heading readouts in degrees, and a textual cardinal (N, E, S, or W) or intercardinal (NE, SE, SW, or NW) point. During system initialization, the PCF8576C is configured for low-power operation, 1:4 drive mode, and a 1/3 bias level. The

Command byte ASCII (hex)	Description	Time delay
w (77)	Write to EEPROM	70 ms
r (72)	Read from EEPROM	70 ms
G (47)	Write to RAM register	70 ms
g (67)	Read from RAM register	70 ms
S (53)	Enter Sleep mode (Sleep)	10 ms
W (57)	Exit Sleep mode (Wake-up)	100 ms
O (4F)	Update bridge offsets (S/R now)	6,000 ms
C (43)	Enter User Calibration mode	10 ms
E (45)	Exit User Calibration mode	14,000 ms
E (45)	Save Op mode to EEPROM	125 ms
A (41)	Get data, compensate, and calculate new heading	6,000 ms

Table 2—Commands supported by SparkFun Electronics's HMC6352 compass module

1:4 multiplexed VIM-878-DP LCD dictates the 1:4 drive-mode setting, and the PCF8567C datasheet recommends a 1/3 bias level when using 1:4 drive mode.

The display memory in the controller is configured as a 40 × 4-bit array, and each of the 16-segment digits maps onto four memory addresses. Each byte written to display RAM is split across two display memory addresses. The upper nibble is written to the current address pointer and the lower nibble is written to the current address pointer plus one. After each display memory write, the address pointer is incremented by two.

By configuring all of the blocks for low-power operation and leveraging the MCU's Stop mode, the system can operate continuously on two AA batteries for a significant length of time. Using the evaluation board, the system's active- and idle-mode currents were measured. The active current was 4.5 mA and the idle current was just 20 µA. The system is set to update the heading information every 1.5 s. The measurement time to obtain a new heading was 20 ms. With these values, you can calculate the average current drawn by the system:

$$\frac{(\text{active current in microamps}) \times (\text{time to calculate heading})}{(\text{total time}) + (\text{idle current in microamps})} \times \left(1 - \left(\frac{\text{time to calculate heading}}{\text{total time}}\right)\right)$$

$$4{,}500\left(\frac{0.02}{1.52}\right) + 20\left[1 - \left(\frac{0.02}{1.52}\right)\right] = 79.73\ \mu\text{A average current}$$

Assuming the capacity of an AA battery is 2,500 mAh, the expected continuous runtime is 3.5 years (i.e., 2,500/.07973 = 31,355 hours).[1]

That run time is close to, if not longer than, the shelf life of the batteries. Size AAA batteries have about half the capacity of AA cells, so the system should run about 1.75 years. Allowing for losses due to self discharge and other effects, a system should run for 1 to 1.5 years on AAA cells, and 2.5 to 3 years on AA cells. Thus, a high level of device integration and simple serial-communication channels translate into a highly functional device with low firmware overhead. This was essential to my goal of creating a low-power, low-parts count digital compass.

Jesse Marroquin (jesse.marroquin@maxim-ic.com) is a senior member of the technical staff with Maxim Integrated Products. After graduating from Texas State Technical Institute at Waco, he worked in the telecom and semiconductor industries before joining Maxim. Jesse's technical interests include embedded Linux, programming, and microcontroller based designs.

REFERENCES

[1] Energizer Holdings, Inc., "AA Data Sheet," http://data.energizer.com/PDFs/E91.pdf, and "AAA Data Sheet," http://data.energizer.com/PDFs/E92.pdf.

SOURCES

IAR Embedded Workbench
IAR Systems | www.iar.com

MAXQ610 Microcontroller, MAX-IDE, and IAR Embedded Workbench development tools
Maxim Integrated Products, Inc. | www.maxim-ic.com

PCF8576C LCD Driver
NXP Semiconductors | www.nxp.com

HMC6352 Digital compass module
SparkFun Electronics | www.sparkfun.com

VIM-878-DP LCD
Varitronix Ltd. | www.varitronix.com

by Sivakumar Govindarajan (India)

Standby Current Management in Embedded Systems

The standby current of any system is consumed when the system is inactive. A successful embedded system engineer must intelligently manage the quiescent current to prevent unnecessary power dissipation.

The standby current of any system is a quiescent current consumed when the system is in an inactive/standby state. This standby current is also widely referred to as "dark current" or "sleep current" in the industry, terms that are interchangeably used in this article. It is important to manage this quiescent current to ensure the longevity of a battery, use less power dissipation, and improve the life of power components. This is of paramount importance in battery-operated portable systems and in the automotive industry. Consider mobile phones that can last for days in standby mode, or the latest generation of cars that come with dozens of electronic units. If the quiescent current is not well managed in your car or mobile phone, you won't be able to start your car in the morning or enjoy mobile conversations without constantly using a charger. Typically, the automotive industry calls for sub-mA dark current for electronic control units.

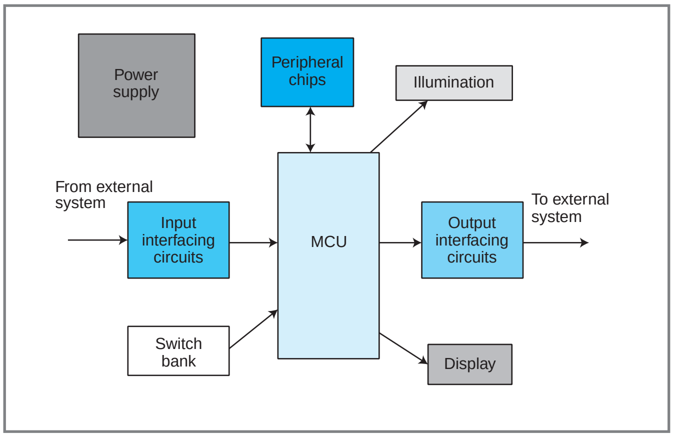


Figure 1—Generic embedded system architecture

CURRENT DRAINERS

In an embedded system, there are many blocks to be considered for the successful management of quiescent current. Figure 1 shows a typical embedded system block diagram in which all the blocks are potential current drainers during standby mode if they are not properly designed. The quiescent current shown in Figure 1 can be defined as:

Iq (System) =
Iq (Power supply) + Iq (Peripheral chips) + Iq (MCU) +
Iq (Input interfacing circuits) + Iq (Output interfacing circuits) +
Iq (Display driver) + Iq (Illumination driver) +
Iq (Switch bank driver)

POWER SUPPLY

We'll start with the power supply as it's an essential block in any electronic system. Based on application requirements, there are two choices when selecting a regulated power supply: linear and switching. In both topologies, there exists quiescent current consumed by the regulator IC and its associated components. Considering cost and system requirements, it can be difficult to identify the right power supply solution from many available options in the selected power supply topology.

As in any power supply design, the first step is to estimate typical and maximum load current. This is an

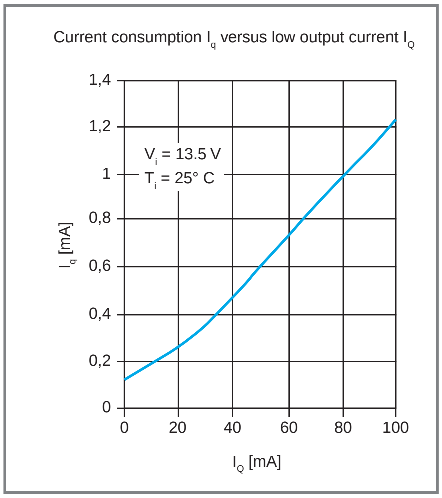


Figure 2—Load current versus regulator quiescent current of the IFX25401

essential step in selecting the right power supply topology and sizing of the power supply. The second step is to estimate the standby current of the system (minimum load), which can be challenging in the initial stage, as most of the other parts of the system are still on the drawing board. Many designers opt for designing a power supply at a later stage so that they get good visibility to the load currents. Nevertheless, it is possible to fairly estimate the standby current at the initial stage by going through the datasheets of all architectural components (e.g., display drivers and peripheral chips). The summation of the individual quiescent current of the chips provides a ballpark number, as detailed in the previous equation.

Typically, the quiescent current of low-drop-out (LDO) linear regulators is a function of load current and is directly proportional. This data is essential for ultra-low standby current designs. Designers should carefully check the details on the power supply datasheet. If they are not in the datasheet, ask the IC supplier to provide the data or at least perform a lab analysis. Note that typical standard regulators, such as 78*xx*, consume a lot of quiescent current and are not suitable for ultra-low quiescent current designs.

For example, Infineon Technologies's IFX25401 low dropout linear voltage regulator datasheet provides a good graph for load current (I_Q) versus the regulator quiescent current (I_q) (see Figure 2).

In the case of DC/DC converters, quiescent current of the converter IC is usually managed by pulling down the switching frequency combined with intelligent switching schemes. For example, Texas Instruments's TPS62110 DC/DC converter uses PFM mode to reduce quiescent current to 20 µA. Figure 3 shows the input voltage versus the quiescent current of this IC.

Now, based on the estimated standby current of all components in the system, you should select a power supply IC that has a fairly low quiescent current for the minimum load current (standby load current of all other components) to satisfy the overall system standby current as shown in the preceding equation.

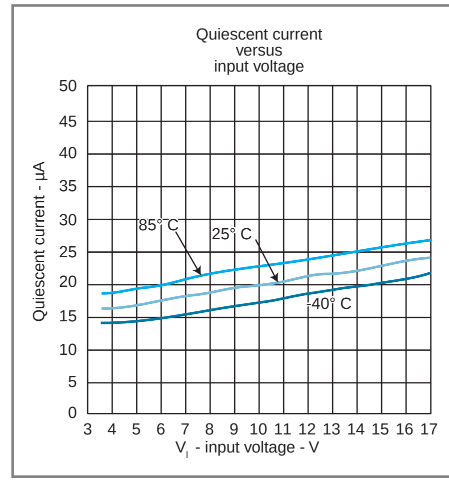


Figure 3—Input voltage versus quiescent current of the TPS62110

PERIPHERAL CHIPS

In an embedded system, any device can be interfaced as a peripheral device with different schemes. It is the designer's responsibility to select the right devices with the right schemes for the application to meet the requirements and cost.

Here is one such scheme: If the peripheral IC has a dedicated enable/standby pin, it is easy to put it in an inactive state by asserting it. However, sometimes the standby current of the IC is much higher for ultra-low quiescent current designs. In such cases, it may be wise to completely kill the power to the device by providing a switched rail, as shown in Figure 4.

This kind of solution is useful if standard TTL chips are interfaced (e.g., with MCU as I/O expander, shift registers), as these chips are significant current-consuming devices. However, care must be taken while completely switching power to ICs to ensure that there are no side effects in the system, especially due to current sneak paths. This can be achieved by placing all the interfacing ports of the MCU in proper states before switching off the peripheral device.

Some side effects include: Switching off the supply can make the chip's pins exhibit high impedance that could be more susceptible to EMI. This should be taken care of at the system level by proper pull-up or pull-down resistors, if needed. There is a possibility of inadvertently driving the switched rail by an MCU through internal clamping diodes of peripherals if the proper state is not maintained on the MCU interfacing pins. So, attention must be paid at both the MCU and the external devices end. More on that later.

If the peripheral devices are connected with some communication protocol, such as SPI, I²C, or UART, designers should determine whether or not they have proper standby commands for power-down modes. If not, it is advisable to go

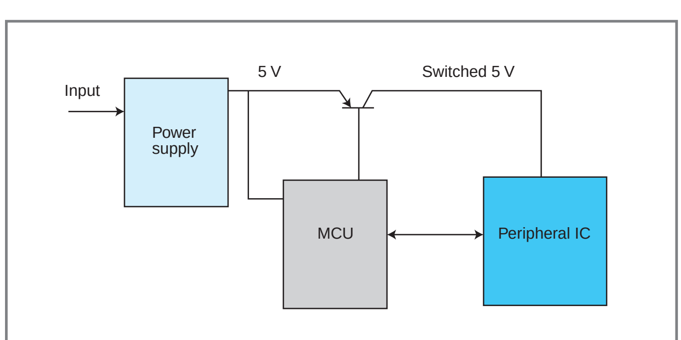


Figure 4—Switched rail concept for peripheral devices

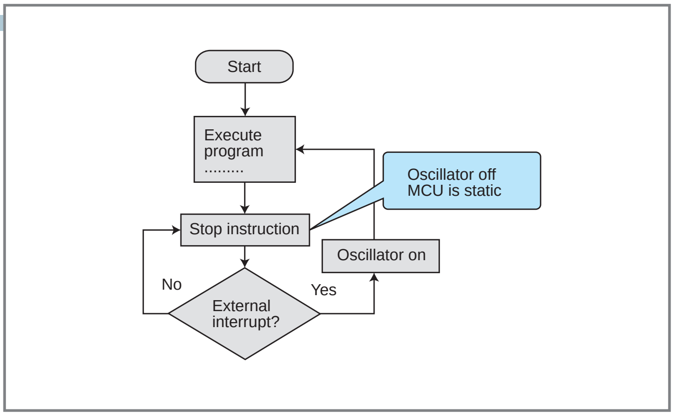


Figure 5—Typical control flow for MCUs in Stop mode

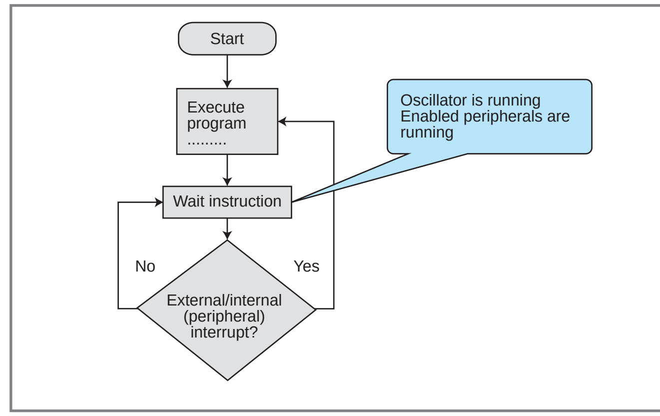


Figure 6—Typical control flow for MCUs in Wait mode

with a switched-rail scheme, as explained, if the quiescent current of the device under consideration is high. If the total quiescent current of the peripheral ICs is insignificant to the total system standby current, it is better to leave them always connected to the power supply.

MICROCONTROLLERS

The microcontroller with software is the brain of any embedded system, along with supporting on-chip peripherals. Due to improved semiconductor manufacturing processes and the low cost-per-transistor inside MCUs, MCUs currently come with a lot of muscle power with numerous peripherals, increased flash/RAM size, and high-speed processing power. However, during Standby mode, most of these features are not required to function except for a few to do housekeeping activity of the system. Fortunately, almost all the latest-generation MCUs offer finer control over individual peripherals, CPU clocks, and other on-chip subsystems through software that enables designers to implement power-saving modes in the system as needed.

Power-saving implementation in the MCU is highly influenced by overall system requirements. Designers must thoroughly understand what the system should do while idling, during an inactive state, and during a sleep/standby state. This clear understanding will enable the designer to answer critical questions, such as: Does the system run when in an inactive state? Can the system be completely stopped? Have all the system's ports been properly configured for Standby mode? Can the system clock of the MCU be reduced? Can an MCU's internal oscillator be used during Standby mode? Can on-chip peripherals be switched off? If not on all of the peripherals, at least on the ones that are not required during Standby mode?

Oscillator mode	Maximum operating current at 5 V
32 kHz (LP oscillator mode)	60 µA
4 MHz (XT oscillator mode)	1.4 mA
8 MHz (HFINTOSC mode)	2.3 mA
20 MHz (HS oscillator mode)	4.8 mA

Table 1—Microchip Technology PIC16F8xx operating current for various oscillator configurations

In any MCU, complete Stop mode is the ultimate power-saving mode, provided all other on-chip peripherals are also in Power-down mode. So, if the system does not have any job to do when in standby, it is best to keep the MCU in complete Stop mode. In this mode, the main clock oscillator is switched off and the MCU will be in a static state. Since the clock itself is switched off, to wake up the MCU, you usually must provide an external interrupt. Upon receiving an interrupt, the clock oscillator starts running and the MCU resumes execution. See Figure 5 for a typical Stop mode process.

If the system is required to do some job while idling, but with reduced performance, Wait mode or clock throttling is preferred.

In Wait mode the MCU's oscillator will run, but the MCU will not execute a program. The MCU starts dozing off as soon as it encounters a Wait instruction in the program. It will come out of this mode only upon receiving an interrupt. Unlike Stop mode, an on-chip peripheral interrupt can also resume the MCU from Wait state. This is the key difference between Stop and Wait mode. Each MCU manufacturer uses different names for Stop and Wait modes (e.g., sleep, power down, doze). See Figure 6 for a Wait mode process.

With the clock throttling method, the clock frequency can be throttled down with the help of an on-chip PLL, low-speed internal oscillator based on available options in the MCU. Care must be taken when throttling the clock as it will directly affect all peripherals supplied by a throttled clock. Throttling is usually done when

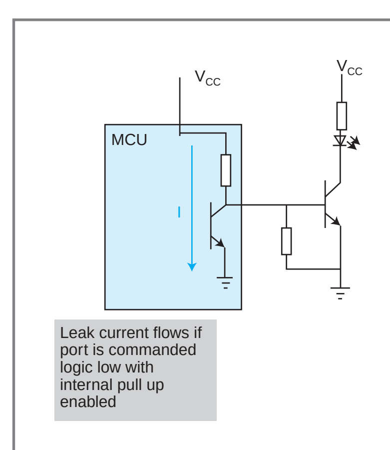


Figure 7—A simple LED driving circuit that could complicate things if the MCU is not properly configured

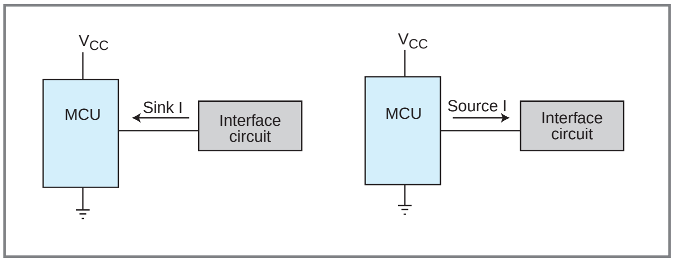


Figure 8—Interfacing current flow path

there are no time-critical items, such as communication (e.g., UART, SPI, I^2C) as it directly affects baud rate.

See Table 1 for an extract from Microchip Technology's PIC16F8*xx* MCU datasheet. Based on the oscillator configuration, the current ranges from 60 µA to 4.8 mA for a 5-V operation.

Typically, on-chip peripherals in MCUs consume less current when the clock input is inhibited. This is often done either by an enable/disable bit in the control register of each peripheral or by peripheral clock routing registers in the clock module. For effective power-saving implementation, designers must identify the peripherals that are not required to be active during Standby mode and disable them when entering Stop/Wait modes.

For example, the PWM peripheral that powers illumination (like in a mobile phone) might not be required to run during Standby mode and it is wise to save some µA or mA of current.

Designers should also concentrate on the I/O ports configuration of MCUs for Standby mode. Due to various off-chip peripherals/circuit-interfacing requirements, many designs require the usage of internal pull-up or pull-down resistors at the I/O port. It is essential to understand the effect of internal pull-up/pull-down resistors on the current leak path.

For example, consider the simple LED drive circuit shown in Figure 7. There are many ways of implementing this simple LED drive interface circuit. This example is just one option to demonstrate the effect of incorrect port configuration. To make the circuit function without damaging the BJT/MCU port, the port should be configured as open collector output with pull-up enabled. For normal operation, when commanded for output high, the pull-up resistor will provide bias to the BJT and the LED will be on. To switch off the LED, command the port to output low.

In Standby mode, if it is driven low with the pull-up enabled, there will be current flowing in the pull-up resistor (as shown with the blue arrow in Figure 7). This current would substantially add up with standby current in case of sub-mA design.

To avoid this kind of design pitfall, configure the port as push-pull without pull-up and set the commanding output low. However, for normal operation it should be reconfigured for open collector with pull-up enabled to avoid damaging the BJT/MCU port. This is an example; designers should pay attention to similar points for pull-down and

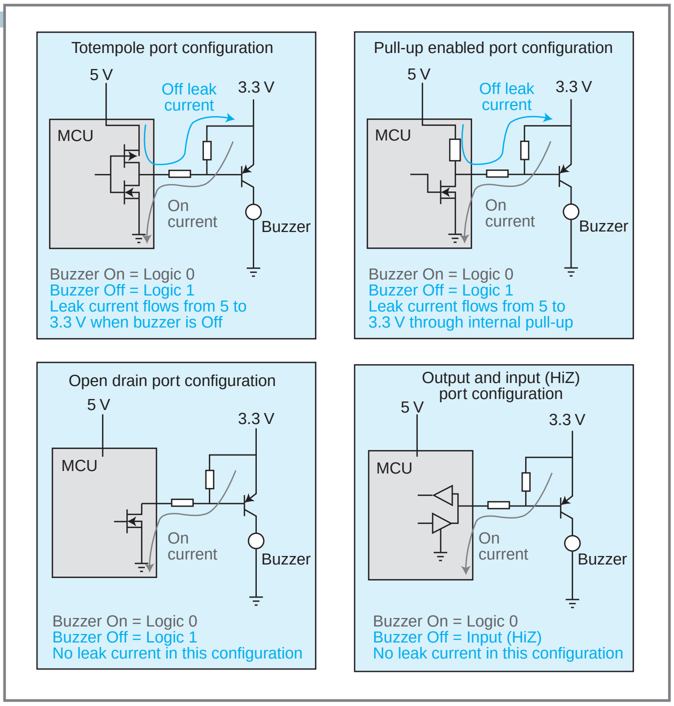


Figure 9—Mixed-signal interfacing current paths

any other potential current leak paths within the MCU through ports and peripherals.

MCU INTERFACING CIRCUITS

The MCU communicates with the external world through the interfacing circuits. Typically, interfacing circuits are composed of ICs and discrete components. As Figure 8 shows, fundamentally, there are only two current flowing paths in MCU interfacing, sinking current to the port and sourcing current from the port, leaving aside internal leaks, as explained in Figure 7. Either of these currents could contribute to increasing the standby current if careful consideration is not given in the interface circuit design.

Often, current leak problems occur in the mixed usage of different power rails in the system. This is due to the fact that MCUs usually operate at 1.8 to 5 V while interfacing circuits might use a high or low voltage rail from the MCU rail. This may cause the leak current to flow at interfacing node from high rail to low rail. Figure 9 and Figure 10 show different scenarios and ways of addressing each issue. Figure 9 provides a glimpse of what could go wrong when interfacing circuits with MCU. There are many types of interfacing circuits that could cause leakage current trouble when proper consideration is not given in the design. Fortunately, with the kind of circuits shown in Figure 9, it is possible to stop the leakage current by changing software with proper configuration. In reality, not everything can be addressed in software. Imagine a scenario if leakage cannot be addressed using software. The net effect would be design reiteration, new PCB, new testing, etc., which would burn a big hole in the project's budget and timing. One such scenario is shown in Figure 10. In this case, there will be leak current flowing from the 12-V rail to the 5-V rail because the I/O ports of MCU are usually diode clamped to the rails, as shown. This type of issue can't be addressed in software, and hardware redesign is required.

TIP OF THE ICEBERG

Embedded circuit designers should carefully analyze all possible current paths for normal and standby modes during the design stage to avoid last-minute surprises, added costs, and schedule overruns. It is also the responsibility of the hardware/system engineer to compile a proper specification detailing port/peripheral configuration to the software team to ensure the design works as intended. The examples provided are potential issues that can be caught and addressed during design reviews to avoid design re-spins. This article explores a few possibilities that represent just the tip of the iceberg. A much bigger obstacle might be underlying in complex mixed-signal multiprocessor embedded systems. So, watch out! ■

Sivakumar Govindarajan (author.gsiva@gmail.com) is a professional engineer with an MS in Software System Engineering, a degree in Information Systems Engineering, and a diploma in Electronics and Communication Engineering. He holds a senior position at a leading multinational Tier-1 automotive company based in Chennai, India. Siva is responsible for defining automotive product architecture and product development engineering.

SOURCES

IFX25401 Low dropout linear voltage regulator
Infineon Technologies | www.infineon.com

PIC16F8xx Microcontroller
Microchip Technology, Inc. | www.microchip.com

TPS62110 DC/DC converter
Texas Instruments, Inc. | www.ti.com

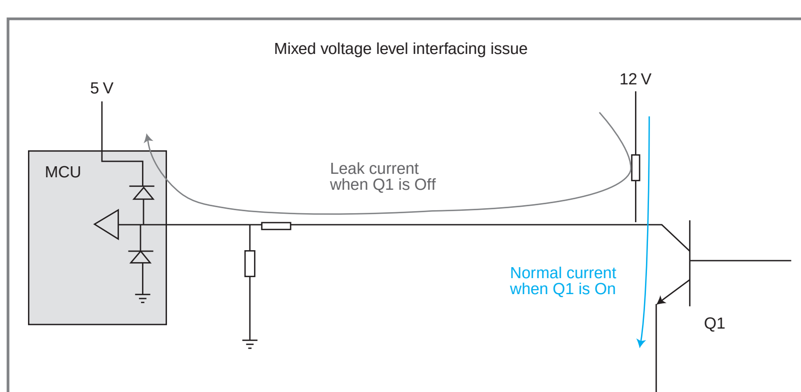


Figure 10—Sneak current that can't be addressed by software

by Aleksander Borysiuk (Poland)

Access Control System Design

Build an MCU-Based PIN Reader

This access control PIN reader system is designed to increase PIN code security. The system displays random keypad patterns. The microcontroller detects touch, computes touch position coordinates, and converts them to key codes that are then transmitted to an access controller.

I do not know whether the world has become more dangerous or security technology has simply entered the mass market, but I have noticed a steady increase in security systems. We may not realize it, but we are almost always surrounded by dozens of alarm, surveillance, and monitoring systems. These include access control (AC) systems. Their main task is to provide authorized users with access to protected premises. There are different user-recognition methods implemented in these systems. Examples include PINs, ID tags, and biometric feature reading. Each method has pros and cons. In Europe, where I live, user-recognition methods applied in AC systems should conform to the European Standard EN 50133-1 (see Table 1).[1]

The simplest and least expensive method of user recognition is a PIN. As you can see in Table 1, PINs are applied in AC systems with low security levels (recognition Class 1) as well as in AC systems with high security levels (recognition Class 3). Nevertheless, PINs have a weakness: they're very susceptible to attack. PINs may be viewed, guessed, or scanned. There are a lot of methods for PIN attacks. (One of the most memorable was used by Nicolas Cage in the movie *National Treasure*. You may have seen it.) But, in many cases, sophisticated methods of PIN interception are unnecessary. If you look at a keypad, it is usually clear which keys are used most often. What usually remains for the attacker is to find the correct sequence of digits. But what would happen if the keypad pattern were different every time you entered your PIN? Of course, the PIN you enter would remain the same, but because key meanings are different at each access, you would have to press different keys every time you entered your PIN. Consequently, the remote monitoring of your hand movements or keypad scanning would not guarantee success. To test this concept, I decided to design an AC PIN reader with a random keypad pattern (see Photo 1).

HARDWARE & DISPLAY

The PIN reader is built around a Texas Instruments DK-LM3S9B96 development kit, which was designed for TI's latest Stellaris microcontroller (see Figure 1). It contains all the blocks you may expect in a PIN reader. There is a user interface, a microcontroller, and an external system interface.

There are many ways you could construct the PIN reader's user interface. It should include a display that shows a keypad pattern and an input device. When I was considering this problem, my first idea was to use NKK Switches SmartSwitches, which are single-pole switches with tiny embedded LCDs. (For more information about this technology, refer to Aubrey Kagan's two-part series "Driving the NKK SmartSwitch," which appeared in *Circuit Cellar* 144 and 145, 2002.) Unfortunately, they seemed too expensive

User recognition class	Recognition method	Example
Class 0	No user recognition	"Press to exit" key
Class 1	User recognition based on remembered information	Password, PIN
Class 2	User recognition based on ID tag or biometric feature	Proximity ID card or fingerprints
Class 3	User recognition based on ID tag and remembered information, biometric feature and remembered information, or ID tag and biometric feature	Proximity ID card and PIN

Table 1—The European Standard EN 50133-1 defines four classes of user recognition in AC systems. The higher the user recognition class, the higher the security level, and the higher the cost for the reader and its maintenance.

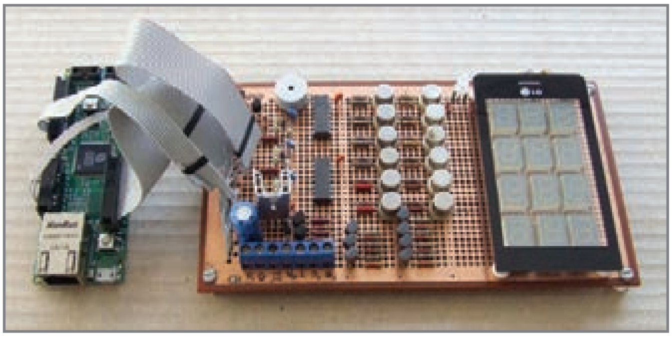


Photo 1—This is a complete AC PIN reader. The keypad pattern display and its control circuits occupy three-fourths of the universal board. The remaining area (on the left-hand side of the PCB) is populated by the Wiegand interface and the voltage regulator.

and were difficult to obtain. My second idea was to build a user interface with an LCD equipped with a touch panel. But using a color TFT display to show only key patterns seemed like overkill and was still too expensive. Nevertheless, a couple months after finishing my project, I found out about Texas Instruments's RDK-IDM-SBC hardware package, which was designed for the Stellaris microcontroller family. An example application in the kit presents a PIN-controlled electronic lock with a random keypad pattern.

For this project, I used seven-segment LED displays and a resistive-touch panel as an input device (see Figure 1). Initially, I had some problems selecting the touch panel. Although they seem ubiquitous, touch panels are usually built in LCD modules and are difficult to purchase separately. I was going to scavenge such panels from a damaged TFT display, but I accidentally discovered another big source of touch panels. It turned out that these components are offered in a wide variety as spare parts for mobile phones. Panels for older mobile phone models are available at an especially reasonable price. I decided on the touch panel originally designed for an LG KU990 phone. It has a working area size of 66 × 40 mm as well as X and Y resistance equal to 280 Ω and 480 Ω, respectively.

I was working on a tight budget when designing the PIN reader. Therefore, I emphasized cost optimization. As a result, the circuit features only generic components that are readily available in electronic surplus stores. You may have these parts at your workbench, too. The DK-LM359B96-based section of the circuit is shown in Figure 2. The most complex part of the design is the keypad pattern display (see Figure 3). The display is formed by a 3 × 4 array of seven-segment numeric LED modules that are driven in multiplex mode. The switching rate equals 1/12, which is a rather high value for LEDs. Therefore, the seven-segment displays have to be high efficiency (preferably ultra bright) to be visible in daylight. My project uses Kingbright's SA39-11EWA LED display modules with segment currents set to 30 mA, which is the maximum DC current for SA39-11 displays. Technically, it is possible to drive display segments in multiplex mode with a much higher current that results in brighter symbols. For example, peak current for SA-11EWA modules is specified as 160 mA at duty cycle and a pulse width equal to 1/10 and 0.1 ms, respectively. But, when implementing software controlled multiplexing of displays, it is not advisable to go that far. Should the display control routine unexpectedly hang or stop, the LED module will be fried. Therefore, it is safer to stay within display DC current limits. The display module anodes and segments are directly controlled by the microcontroller via transistor switches. As Figure 3 shows, the anode switches have extra '04 inverters added.

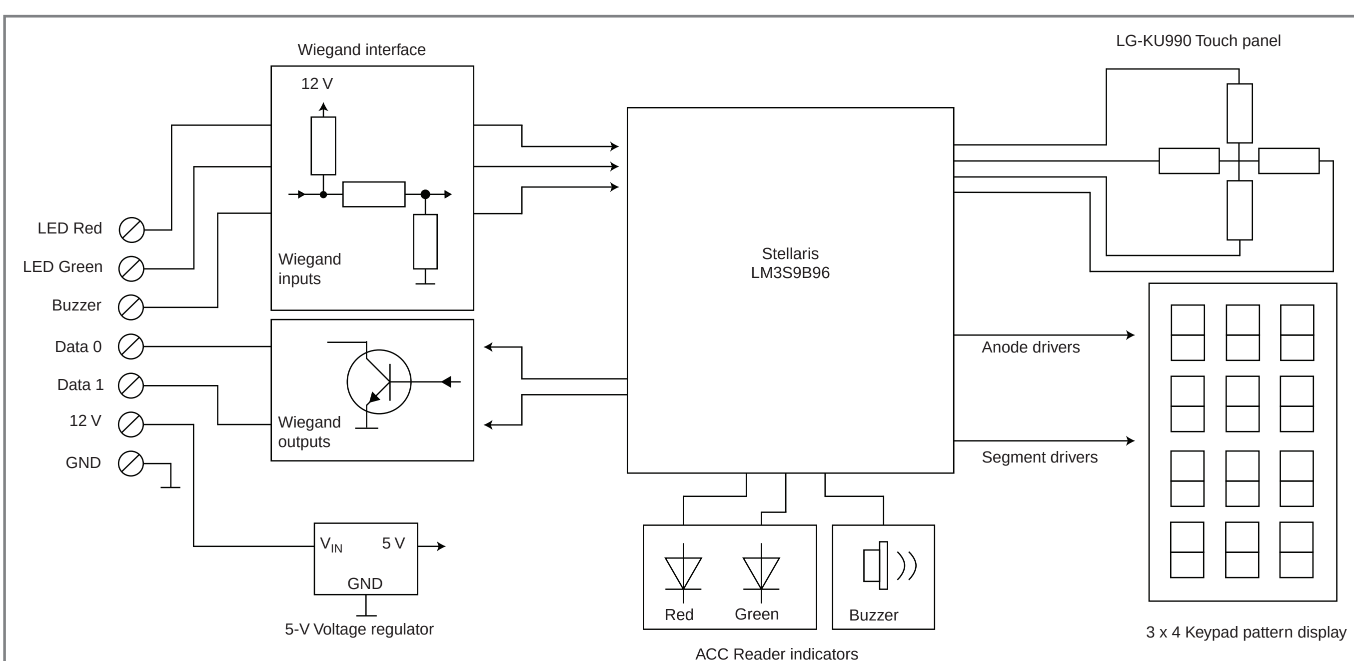


Figure 1—The block diagram of the device reflects the typical structure of an AC PIN reader. The keypad pattern display and the touch panel are, in this case, substituted for a 3 x 4 push button array of a standard PIN reader.

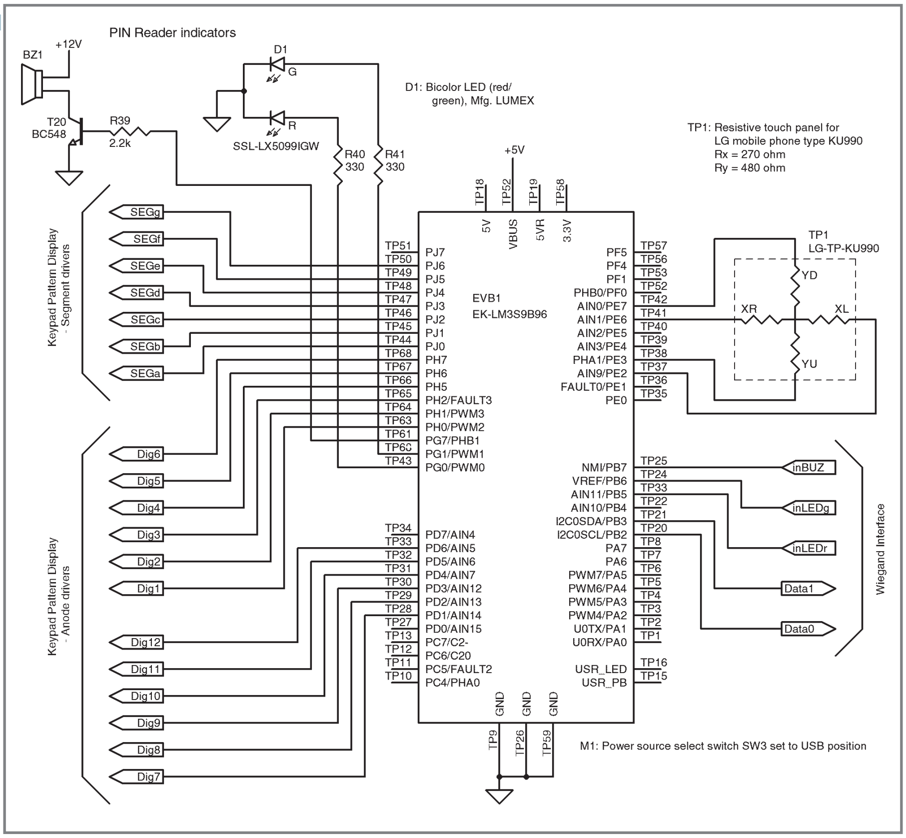


Figure 2—The DK-LM3S9B96 evaluation board is powered via a VBUS pin; therefore, the power select switch on the module has to be set to the USB position. Otherwise, a power supply voltage of 5 V may be shorted to a power supply provided by the in-circuit debug interface board.

Their primary task is voltage conversion between 3-V logic levels of the microcontroller and 5-V supply of the displays. Thus, these inverters have to accept TTL input signal levels. I used 74LS family devices in the prototype, but more modern 74HCT chips make a good choice, too.

The resistive-touch panel is originally serviced in a KU990 mobile phone by Texas Instruments's specialized TSC2007 touch screen controller. This integrated circuit dramatically simplifies the application of the resistive-touch panel but adds extra cost to the design. Luckily, most contemporary microcontrollers can directly control the resistive-touch panel. The Stellaris chip has more than enough

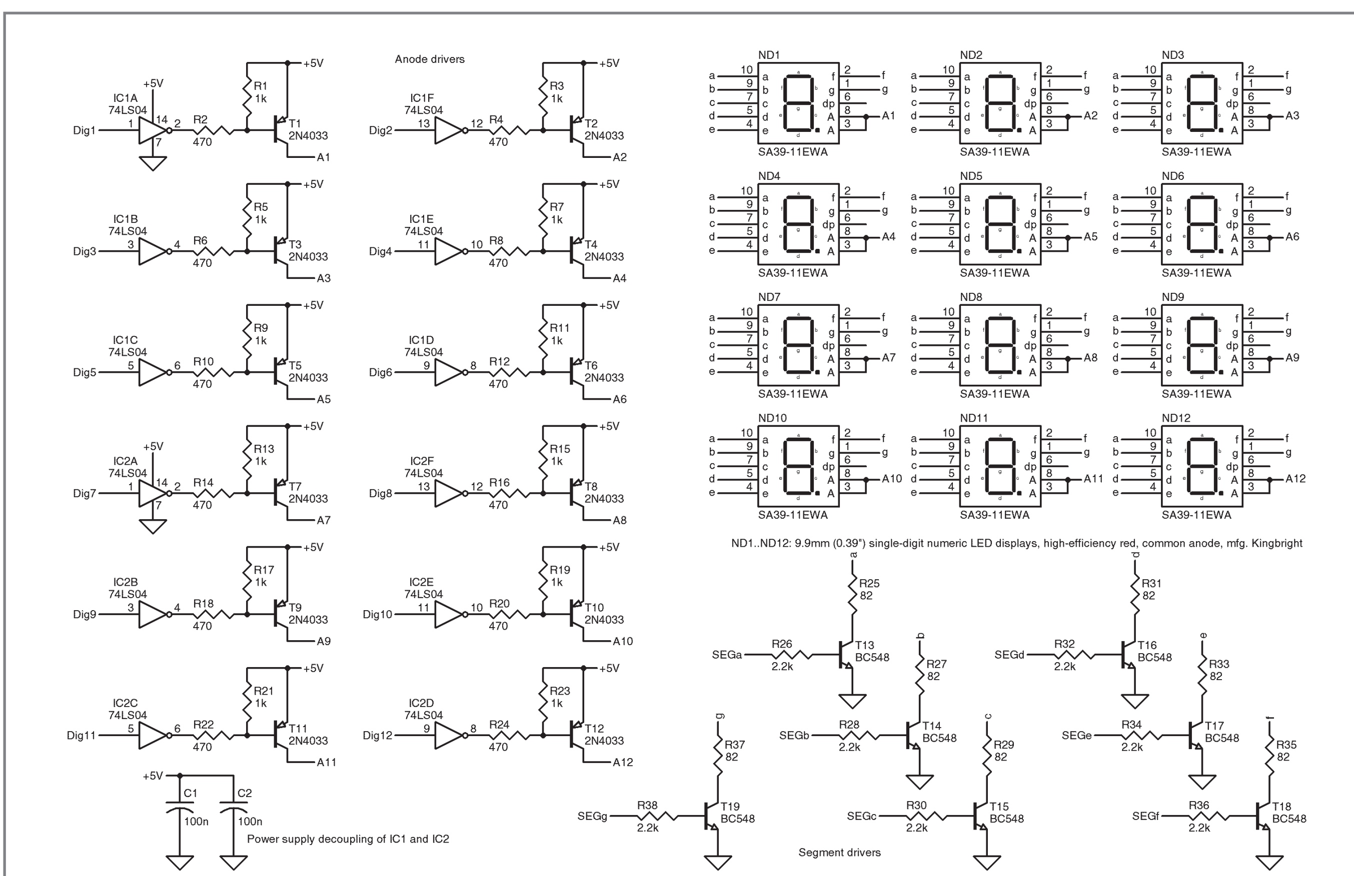


Figure 3—In most cases, the multiplexing of seven-segment LED displays is the best solution when more than one display module has to be controlled. Although the driving circuit looks complicated, it uses fewer components and fewer CPU I/O ports than the static display drive method.

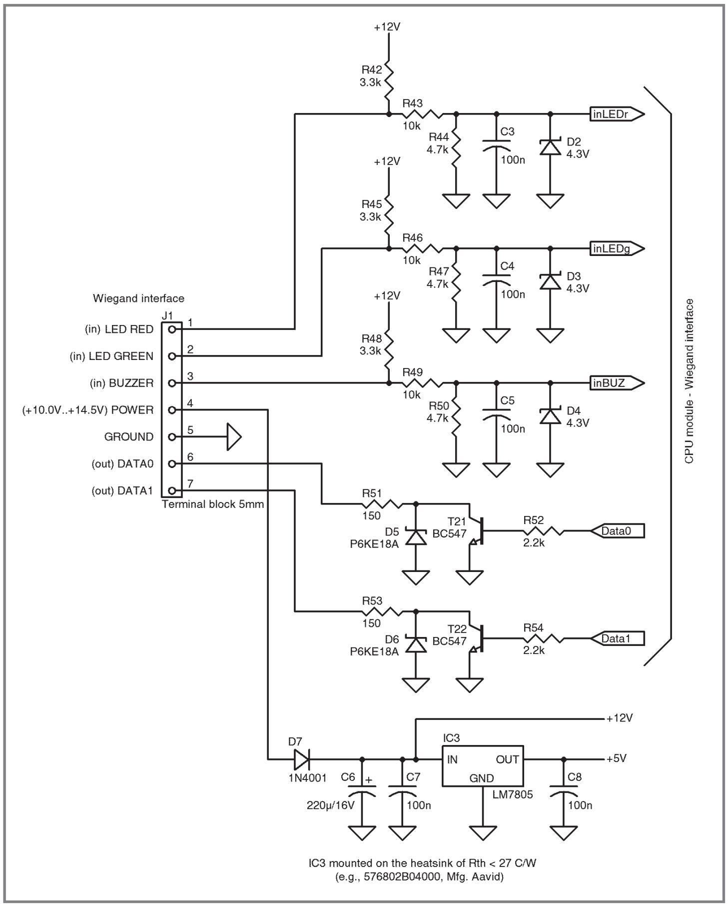


Figure 4—The Wiegand interface is an informal standard for reading devices in the AC industry. This interface is implemented in a majority of controllers although not all signals that control reader indicators (i.e., LED red, LED green, Buzzer) are always present.

resources to perform this task. It should be no surprise (the tight budget) that I applied the second solution in my project. The touch panel is connected to the microcontroller's port E (see Figure 3) and is serviced by its I/O drivers and ADC converter.

The last block of the circuit is the Wiegand interface (see Figure 4). Although this interface isn't formally standardized, it is typically used by most types of AC systems, enabling seamless connecting of AC readers to controllers, regardless of manufacturer. As Figure 4 shows, this interface is very simple. It consists of two data lines (DATA0 and DATA1) that transfer data from the reader to the AC controller and three control lines (LED Red, LED Green, and Buzzer) that are used by the controller to activate reader indicators. The Zener and Transil diodes and capacitors in the interface circuit are there to protect the microcontroller I/O ports and output transistors T21 and T22. This is essential since Wiegand interface wires can be up to 150 m!

All the circuit components were assembled on one universal board that is connected with a microcontroller board with four flat cables and IDC connectors (see Photo 1). During circuit assembly there were some problems with the touch panel. Because it has no fixing holes, it was mounted above LED displays on four spacer bolts and fixed to the bolts with double-sided adhesive tape. But the real problem turned out to be its 0.5-mm pin pitch connector. It is difficult to buy the socket for this connector, and soldering this socket to the universal printed circuit board (PCB) is even more difficult. Therefore, I did not use the socket. I simply soldered 0.2-mm copper wires directly to the connector pads of the touch panel module and then soldered these wires to pads on the PCB (see Photo 2). This work was arduous and required a lot of patience. The key to success was to coat the end of each wire with tiny drop of solder, cover it with flux, touch it to the pad, and then heat the wire with a soldering iron. Remember, you mustn't heat the touch-panel pad directly. Heat only the wire! Otherwise, you will end up with two or even three pads of the touch panel shorted by a solder blob. After assembly, it is a good idea to cover the touch-panel connector and wires with thermal glue for mechanical protection.

RTOS

The DK-LM3S9B96 microcontroller is the first member of the Stellaris family that has embedded both a simple RTOS—Wittenstein High Integrity Systems's SafeRTOS—and the Stellaris Peripheral Driver Library. Using an RTOS in a microcontroller simplifies program writing. In this case, the program has to have special structure. It consists of the set of tasks, which are concurrently performed by the microcontroller, rather than a set of functions called by the main function. The tasks communicate with each other via a queue mechanism that is provided by the SafeRTOS system.

When I started this project, I hadn't applied an RTOS in any of my past projects; therefore, I was overwhelmed by the documentation available for the DK-LM3S9B96 microcontroller. To make things a bit easier, I based a skeleton of the PIN reader program on the qs-safertos example in the DK-LM3S9B96 development kit. The structure of the complete program is shown in Figure 5.

PIN READER TASKS

There are seven tasks in my PIN reader application. Five provide PIN reader functionality, while the two remaining tasks are responsible for program debugging. The main task is the Reader task. In one coherent program, it integrates all other tasks and controls upper-level functionality of the device. This task processes messages sent from the touch-panel task, converts received touch-position coordinates to the pressed key's codes, confirms a key press on the display and buzzer, and sends a pressed key code to an AC controller via a Wiegand interface. Additionally, the same information with some extra debugging data is sent out via an RS interface. Last but not least, the Reader task is responsible for generating random keypad patterns.

As you may know, achieving true randomness in microcontrollers is difficult; therefore I did not fight to do so. Instead, I implemented a simple algorithm of random permutation based on an idea I found at www.rgrig.blogspot.com. The entire concept that lies behind it is simple. If you have random permutation of N consecutive numbers, it is enough to

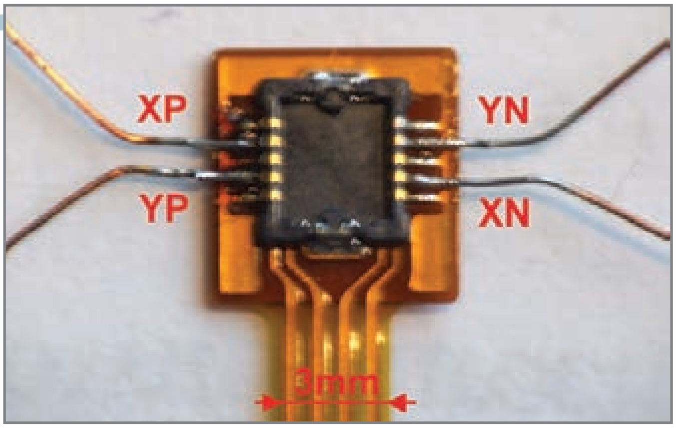


Photo 2—The LG Electronics KU990 touch panel connector has 10 pins, but only four are used. The remaining pins are shorted together and connected to a field of copper that works as a connector shield. This shield is left unconnected in the PIN reader prototype.

pick a random position from 1 to N and place there a number N + 1 moving at the same time as the old number to the end of the set (i.e., to position N + 1). As a result, random permutation of the N + 1 number is achieved. Applying this rule and iterating it from N = 1 to N = 10, I generate random keypad patterns. Random numbers needed for this operation are calculated using the linear feedback shift register (LFSR) method. It is detailed in the program code as well as in the Resources section at the end of this article (see Maxim's application note 1743). My function uses a 31-bit register with feedback taps at bits 2 and 30, which results in a pseudorandom sequence of length 2 147 483 647. Although there are 12 keys in the keypad, random permutation is applied only to digits while function keys # and * stay in the same standard place. (I'll explain why later.) In fact, it reduces the number of keypad patterns by 132 times. Nevertheless, more than 3.6 million keypad patterns (there are exactly 10! factorial permutations—that is, 3 628 800) still remain. This seems more than enough to guarantee the security of entered PINs.

The Display task services the keypad pattern display. The task performs all of the operations needed to display the keypad layout received from the reader task. It transcodes symbol codes to seven-segment display pattern codes, and it multiplexes display modules and drives their segments accordingly. Each seven-segment module is kept on for 2 ms, which means refreshing the entire display takes 24 ms. As you may quickly calculate, the display is refreshed with a frequency of 41.6 Hz, which is enough to make symbol flickering invisible to the human eye.

The Touch Panel task services the touch module. It detects touch panel presses, performs press debouncing, and calculates touch coordinates. I adapted this task code from the touchscreen driver that is available as a part of the DK-LM3S9B96 development kit.

If you're interested in how a resistive-touch panel works and how touch coordinates are calculated, I recommend studying Texas Instruments's TS2007 controller datasheet. (Refer to the Resources section at the end of this article.) This document provides a detailed description of resistive-touch panel service. Although the task in the PIN reader program performs all operations using microcontroller peripherals, the operation concept stays the same. The TS2007 datasheet gives one additional clue. It shows that in case of a resistive-touch panel, it is possible to detect not only the X and Y coordinates of the touch position, but also the strength of the press (i.e., Z coordinate). The latter feature is not implemented in the Touch Panel task code.

The Wiegand interface is serviced by two tasks: the Wiegand Inputs and Reader Indicators task and the Wiegand Output task. The first task controls interface input lines and drives the PIN reader indicators accordingly while the second one is responsible for sending codes of the pressed key via the Data0 and Data1 interface output lines. Format of transmission is compatible with HID Global's ProxPro PIN reader data format, which is another informal standard in the AC industry. In this case, each key code is sent as 6-bit frame: E b3 b2 b1 b0 O, where E is the even parity bit for data bits, b3 b2, b3 b2 b1 b0 is binary code of the pressed key, and O is the odd parity bit for data bits b1 b0.

The remaining two tasks in the PIN reader program, the LED task and the UART task, are original tasks from the qs-safertos example. They are not needed for device operation, but I left them in for debugging purposes. The LED task blinks the user LED on the DK-LM3S9B96 board to indicate that the SafeRTOS kernel is up and running. The UART task is used for sending debugging information about starting the kernel and tasks, as well as touch coordinates and decoded key codes. This data may be received by any RS terminal connected to the DK-LM3S9B96 board. The transmission speed and data format are equal to 115,200 bps and 8N1, respectively.

To write the software, I used the evaluation version of Keil's µVision integrated development environment (IDE). I tried to use many of the embedded SafeRTOS functions and the Stellaris Peripheral Library, which had excellent results with regard to the program code size.

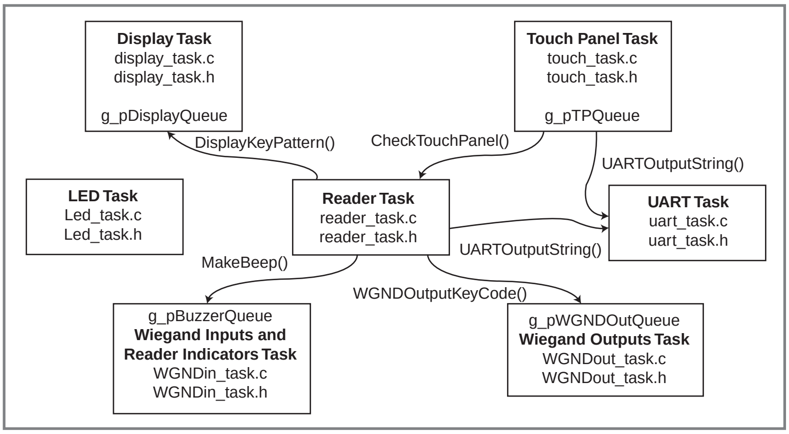


Figure 5—This diagram shows the AC PIN reader's program structure. Each task is written as a separate *.c module with a corresponding *.h header file. Function names alongside arrows depict functions designated for reading from and writing to the task input/output queue.

Photo 3—The standard keypad layout shows up during system start-up (**a**). Later, only random keypad patterns are displayed (**b**, **c**, and **d**). Because seven-segment displays can't present * or # symbols, these symbols are substituted with "c" (for Clear) and "o" (for OK), respectively.

The PIN reader's complete binary code does not exceed 7 KB, which is far below the limits of the µVision IDE evaluation version, not to mention the size of the microcontroller's program memory. Consequently, there should be no problem with further development of the project.

DEVICE OPERATION

Most of the time, the PIN reader is in Idle mode with its keypad pattern display switched off. This reduces power consumption and helps the design to be more "green." But, more importantly, it prevents a user from using a known (and thus unsafe) keypad pattern. After the first touch, the new keypad pattern is generated and displayed. Some sample keypad layouts are shown in Photo 3. Now a PIN code can be entered by touching the display. As user feedback, the pressed symbol is dimmed, which provides a nice impression of the retracting key and shows you which key was recognized by the controller. Additionally, each touch is confirmed by a short beep as any keypad does.

You may generate a new keypad pattern at any time by selecting the asterisk symbol. Most of the access controllers interpret the asterisk symbol as a Clear Buffer command. In such a case, the PIN has to be reentered. For ease of access, asterisk and hash keys are always displayed in the same place that is standard for all keypads. Although it reduces the number of possible keypad patterns, I found that leaving these two keys at standard positions makes operating the random keypad less confusing and simplifies correcting errors. Nevertheless, I have to admit that entering PINs is somewhat slower than on a keypad with a typical layout.

IT ALL MAKES SENSE

I spent some time playing with the device and found it useful in an AC system. It performs its task well, and proves that the idea of random keypad layout makes sense. Maybe you don't need such a sophisticated PIN reader, but you may find some of its design concepts useful in your future projects. The application of the touch panel taken from a mobile phone may be especially useful, as the user input interface seems attractive these days and worth adding to your workbench. After all, it looks like we'll soon have to change the "To press or not to press" dilemma to the "To touch or not to touch" dilemma. ■

Aleksander Borysiuk (alex_priv@wp.pl) holds an MSEE from the Warsaw University of Technology. He also completed post-graduate studies in Physical and Technical Security at the Military Technical Academy in Warsaw. Aleksander works in the security industry designing devices for alarm detection and monitoring systems. His primary area of interest is applications for embedded microcontrollers.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/253.

REFERENCES

[1] Joint Security Industry Council, "European Standard EN 50133-1: Alarm systems. Access Control Systems for Use in Security Applications. Part 1: System Requirements," 2006.

RESOURCES

Maxim Integrated Products, "Pseudo-Random Number Generation Routine for the MAX765*x* Microprocessor," Application Note 1743, 2002.

Texas Instruments, Inc., "Nano-Power Touch Screen Controller with I²C Serial Interface," TSC2007, 2011.

———, "Stellaris Peripheral Driver Library User's Guide," SW-DRL-UG-7243, 2011.

RGrig's Thoughts, Theory and Practice, "Random Permutation," 2005, www.rgrig.blogspot.com.

SOURCES

ProxPro Reader
HID Global | www.hidglobal.com

µVision IDE
Keil – An ARM Company | www.keil.com/arm

SA39-11EWA LED Display
Kingbright Corp. | www.kingbright.com

LG KU990 Mobile phones
LG Electronics, Inc. | www.lg.com

SmartSwitches
NKK Switches | www.nkkswitches.com

DK-LM3S9B96 Microcontroller and development kit, RDK-IDM-SBC Hardware package, and TSC2007 Touch screen controller
Texas Instruments, Inc. | www.ti.com

SafeRTOS Real Time Operating System
Wittenstein High Integrity Systems | www.highintegritysystems.com

ICbank
Gateway to Electronics

MP3P DIY KIT, Do it yourself

(Include Firmware Full source Code, Schematic)

· myPIC

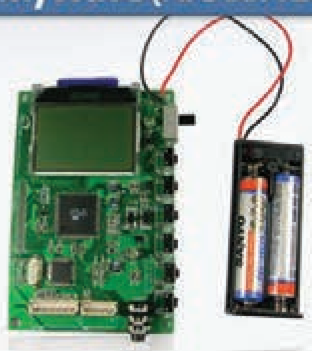
Only
$160 qty 100
$220 qty 1

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)



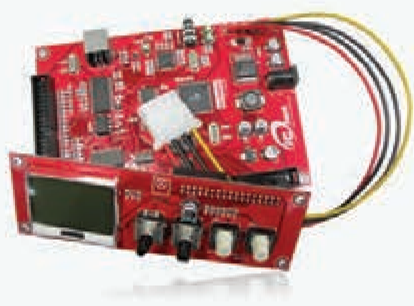
Only
$150 qty 100
$200 qty 1

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

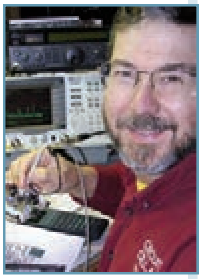
Specification

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

ICbank
Gateway to Electronics

www.icbank.com

by Ed Nisley (USA)



Stepper Motor Drive (Part 1)

The Basics

Stepper motors drive many every-day mechanical systems. Thus, if you're actively designing mechanical systems at the office, your workbench, or both, you're bound to work with stepper motors in the near future. Before you start your next motor control application, take some time to learn about—or reacquaint yourself with—basic stepper motor configurations and a classic motor drive system.

Stepper motors provide simple, inexpensive, reliable rotary motion in mechanical systems with digital controllers, a description that covers everything from factory equipment to desktop scanners. Most *Circuit Cellar* readers have a stepper motor within arm's reach!

Stepper motors seem to be digital devices because their rotors move in discrete steps. However, they're really analog at heart, which means understanding how to use them requires more than just the zeros and ones found on the microcontroller side of the driver circuits.

In this column I'll describe some common configurations and explore a classic motor drive system. Next time, I'll cover contemporary microstepping drivers.

Photo 1—This assortment of four-, five-, and six-wire stepper motors from my collection represents only a small fraction of the available designs. The largest motor is a NEMA 23 size, the two on the right are NEMA 17, and the smallest one, from my surplus heap, has a custom case.

STEPS & WIRES

You'll encounter the National Electrical Manufacturers Association (NEMA) puzzle as soon as you start using steppers. Simply put, the numeric part of an NEMA stepper motor designation gives the approximate diameter of the mounting bolt hole circle in tenths of an inch. For example, an NEMA 17 motor has mounting bolts on a 1.725″ diameter circle, with adjacent bolts separated by:

$$1.22'' = \frac{1.725''}{\sqrt{2}}$$

Although the NEMA designation classifies a motor's size without the burden of too many decimal places, you must consult the motor's datasheet for the actual measurements and bolt sizes before you begin drilling. Note that the NEMA designation does *not* specify the motor's shaft diameter, shaft length, or case length. In fact, it doesn't specify *anything* else about the motor!

With a datasheet on your bench and a (disconnected!) stepper motor in hand, twist the

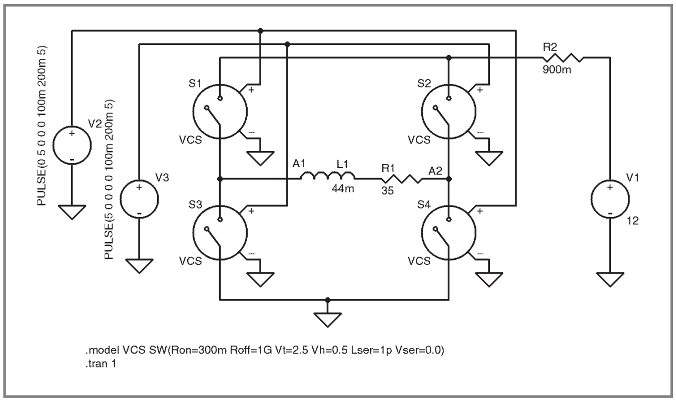


Figure 1—This simulation model of an H-bridge bipolar motor driver can push current in either direction through the motor winding represented by L1 and R1. A bipolar four-wire motor requires one of these H bridges for each winding. The four voltage controlled switches represent simple models of the MOSFET power transistors used in an actual circuit, with their on-state resistances combined in R2.

shaft between your thumb and forefinger. The periodic hesitation is *cogging* due to the rotor's permanent magnetic field aligning with the prominent pole teeth on the stator. The motor's datasheet specifies the *Detent Torque* required to turn the rotor from one cog position to the next, although that value has no significance in most designs.

The number of cogs will be an even fraction of the motor's *Steps Per Revolution*, with both the ratio and torque depending on the motor's mechanical design. For example, the small 48 step/revolution motor in the front left corner of Photo 1 has 24 firm cog positions, the 200 step/revolution NEMA 17 motors on the right have 50 much softer cogs, and the large 200 step/revolution NEMA 24 motor has 100 barely perceptible ticks.

Common stepper motors may have four, five, six, or eight wires or connector pins emerging from the stator's armature windings. Beneath that confusion, however, lies one simple truth: the armature has only two distinct windings, each capable of attracting the rotor's permanent magnetic poles. Current can flow in either direction in each winding and the four possible combinations of current direction define four adjacent *full step* positions. One complete rotation consists of multiples of those four positions and, after you understand how that works, everything else makes perfect sense.

Stepper motors with four wires, the simplest configuration, operate in *bipolar* mode: each winding has two wires and the driver must reverse the current in the windings. Typically, the driver uses four transistors arranged in the H-bridge configuration with a separate H bridge for each winding (see Figure 1). I used voltage controlled switches in the schematic, rather than transistors, to simplify the simulations you'll see later.

Six wire motors bring out a center tap from each winding and operate in *unipolar* mode: the current in each half-winding always flows from the center tap to the outer wire, but the two half-windings induce magnetic flux in opposite directions in the stator. The driver for each complete winding requires only two transistors with the winding's center tap connected to the positive power supply (see Figure 2).

You can run a six wire motor in bipolar mode by disconnecting the center taps and using H-bridge drivers. You must reduce the winding current by a factor of two to avoid magnetic saturation, because energizing the entire winding in bipolar mode produces twice the magnetic flux for a given current.

The manufacturer can connect the two center tap wires together inside the motor and bring out a single common lead to produce a five wire stepper. The combined common wire must carry twice the current of each half-winding.

Splitting the two center tap wires of a six wire motor into four external wires, one for each half-winding, produces an eight wire motor. You must ensure proper polarity among the octopus of leads: a reversed half-winding will attempt to drive the rotor in the wrong direction and, more likely than not, destroy a driver transistor.

As with a six wire motor, you can drive an eight wire motor in unipolar or bipolar mode, with the half-windings in parallel or series. Pay proper attention to currents, magnetic flux, and polarity to keep the motor within its specifications.

You can find motors that superficially resemble steppers with completely different internal construction and drive requirements. As with the NEMA size designation, the number of leads does not mandate anything about a motor, so verify what you have on hand before connecting it to a drive circuit.

For the purposes of this column, I'll ignore some obvious complexities: back EMF due to the motor rotation, magnetic coupling between the half-windings in unipolar motors, dynamic braking, and many others. Understanding the fundamentals will get you most of the way to the goal, but you must also consult the documentation for the motor you're using.

The cost of digital electronics and power transistors was sufficiently high in the "bad old days" to make unipolar steppers the preferred choice. Cheap microcontrollers and low-resistance MOSFETs have eliminated that restriction, so I'll discuss the drive conditions for the simplest case of a four wire bipolar motor.

Homework: generalize the results to the other motor types.

As noted earlier, stepper motors will have multiples of

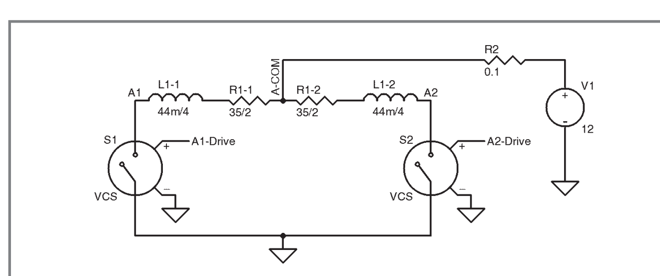


Figure 2—A five or six wire unipolar motor driver requires only one transistor switch for each half-winding: a total of four switches for both windings. An eight wire motor brings out connections to each point of each half-winding.

Step Position	Winding A	Winding B
1	+	+
2	–	+
3	–	–
4	+	–
5	+	+
6	–	+
7	–	–
8	+	–

Table 1—A quadrature sequence of winding currents turns a bipolar stepper motor's rotor in one direction; reversing the sequence turns it in the other direction. Both windings carry current at all times.

four *Full Steps* per revolution, ranging from a very coarse 20 step/revolution on low-end motors to 400 step/revolution for fine positioning, with 200 step/revolution being fairly common. Each full step corresponds to one of the four possible combinations of current direction in the windings. Table 1 shows how the combinations vary during eight successive full steps.

Applying the currents in top-to-bottom sequence turns the rotor eight steps in one direction, while traversing the table upward spins it in the other direction. The patterns repeat in groups of four in each direction.

Homework: plot the two winding currents against the full step numbers, then squint at the graph to recognize the familiar quadrature relationship of sine and cosine waves.

CURRENT & POWER

The current in a stepper motor's winding generates a magnetic field in the armature that attracts the corresponding magnetic poles in the rotor. The force of that attraction produces the torque that drives the motor shaft and, thus, whatever's attached to the shaft.

To a good first approximation, the strength of the magnetic field varies linearly with the winding current, so the motor torque also varies directly with the winding current. The datasheet's *Holding Torque* value gives the minimum torque required to begin turning the rotor when one winding carries its rated current. The rotor will rotate continuously with that torque applied to the shaft.

The datasheet also provides the *Rated Current* and *Resistance* for each winding. Multiplying those two gives the winding's *Rated Voltage*, the DC value that will produce the Rated Current, although that value may not appear in the datasheet.

The Rated Voltage for each winding will be far less than the winding insulation's *Dielectric Strength*, which represents the maximum voltage that may be applied to any wire. Most applications mount the motor on a metallic machine frame that connects to electrical ground, so the Dielectric Strength limits each winding's maximum voltage above ground. Uncontrolled back EMF from a rapidly spinning motor may come surprisingly close to the Dielectric Strength.

The datasheets generally don't list the motor's maximum power dissipation, because the familiar I^2R equation combines the motor's Rated Current and Resistance into that value. Most modern driver chips will control the total current in both windings to keep the power dissipation within the overall limit.

For example, each winding must carry only 0.707 (i.e., $1/\sqrt{2}$) of the Rated Current when the motor is in any of the full-step positions shown in Table 1. As we'll see later, microstepping drivers can fully energize a single winding at the Rated Current when the other winding carries no current.

Homework: Verify that the motor dissipates its maximum rated power under those conditions.

Easy bonus: How much power will it dissipate with both windings carrying their full Rated Current?

As you should expect, the motor's temperature will rise due to its power dissipation. The datasheet provides

Size	NEMA 17
Shaft	3/16″ = 4.78 mm
Current	400 mA
Resistance	35 Ω
Voltage	14 V
Inductance	44 mH
Step size	1.8 degree
Holding torque	2.3 kg·cm maximum
Rotor inertia	20 g·cm^2

Table 2—The datasheet for the Kysan Electronics 1123029 stepper motor (seen in the front-right corner of Photo 1) lists these properties.

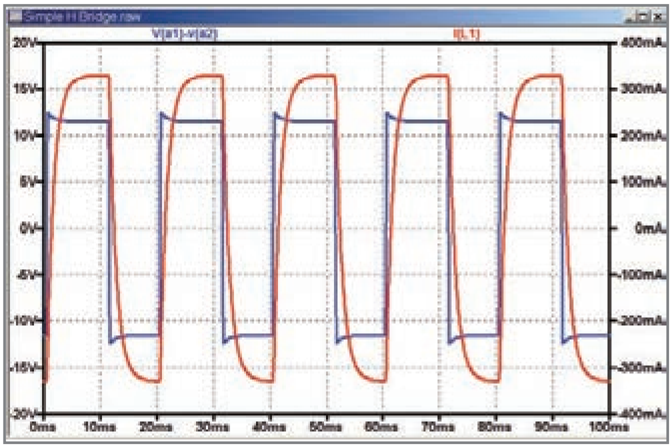


Photo 2—Applying 12 V to a 35 Ω winding eventually produces the expected current, but with a rise time controlled by the winding inductance and resistance.

the maximum allowed temperature rise above a specific ambient temperature range; the sum of the highest ambient temperature and maximum rise can be astonishingly large. Stepper motors can operate with boiling-water frame temperatures, although you shouldn't treat a motor like that unless absolutely necessary.

Stepper motor datasheets generally do not give the thermal coefficient relating temperature rise to power dissipation, probably because motor mounts can range from wooden frames to sheet metal stampings to machined steel frames, all with wildly differing thermal properties. You must verify the motor's temperature rise under operating conditions in your equipment to ensure that the motor runs within specifications.

Now that you understand the basic configurations, it's time to see what happens when current hits the windings.

CLASSIC DRIVE CIRCUITS

Table 2 lists the datasheet values for the Kysan Electronic 42BYG034-4.78 four wire bipolar stepper motor which appears in the front right corner of Photo 1. I'll use this motor as an example because I'm intimately familiar with its performance: it previously powered the Y axis in my MakerBot Industries Thing-O-Matic 3-D printer.

Although the motor has a 14 V Rated Voltage, the MakerBot Industries stepper driver board applied 12 V from a standard PC ATX power supply and I'll continue that tradition here. The values of R1 and L1 in Figure 1 correspond to the winding resistance and inductance given in Table 2, with R2 modeling the driver's Allegro MicroSystems A3977 MOSFET on-state resistances. The pulse generators and ideal voltage-controlled switches simplify the simulation without losing too much detail.

The MBI driver board operates in 1/8 microstep mode and controls the current based on a trimpot setting. The simulations I use here mimic full step mode with no current control to show how a classic voltage drive system works, so these results are not typical of the printer's operation.

Homework: adjust the voltage to suit and evaluate the resulting torque.

The motor shaft

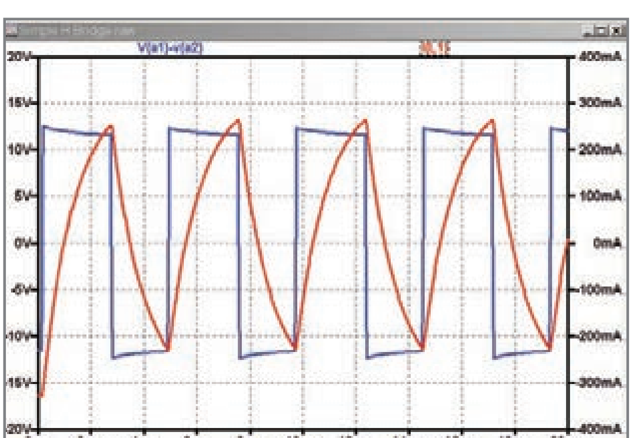


Photo 3—Rotating the motor faster reduces the time available for each drive pulse, but the L/R time constant remains unchanged and the available current drops dramatically.

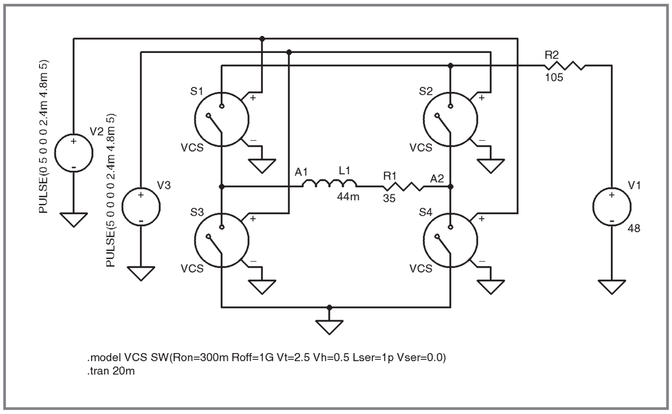


Figure 3—Quadrupling both the supply voltage and the total circuit resistance produces an L/4R drive with faster transitions and terrible power efficiency. Modern semiconductors rendered L/R and L/4R drive circuits obsolete many years ago: good riddance!

turns a 17 tooth pulley carrying a 2 mm pitch timing belt that moves the stage carrying the build surface, so one revolution moves the stage 34 mm. In round numbers, a stock Thing-O-Matic can deposit plastic filament at 30 to 40 mm/s, which means the pulley turns at about 1 revolution/s = 60 RPM. The motor has 200 full step/revolution and each winding reverses polarity on every other step, so each winding sees 100 reversal/s. Under those conditions, the winding current waveform has a 10 ms pulse width and a 20 ms period.

Photo 2 shows the winding current and voltage resulting from running the circuit in Figure 1 with a square wave simulating those drive conditions. Although the winding current eventually reaches the expected DC value, it changes with a time constant τ determined by the winding inductance and the total series resistance:

$$\tau = \frac{L}{R} = \frac{44 \text{ mH}}{35.9\ \Omega} = 1.2 \text{ ms}$$

The current settles within 5% of its final value in three time constants, which accounts for nearly half of the 10 ms pulse width. Because the pulse for a single winding lasts for two full steps, the current does not reach its final value before the next step begins and the motor runs with reduced torque.

However, Photo 3 shows that the same motor will have trouble running at 250 revolution/min, where each current reversal takes only 2.4 ms. The winding current waveform becomes roughly triangular, with the 250 mA peaks occurring near the end of the drive pulses. Because the peak current lags the applied drive pulse, the rotor falls behind its intended position.

The current peaks become even smaller at higher speeds and the motor eventually stalls when its torque drops below the level required to move the load. Motor datasheets often include torque-versus-speed curves that show the torque dropping sharply as the speed increases, so your motor selection process must take into account both the load torque and the speed at which the motor must move that load.

Homework: Superimpose two copies of Photo 3 in quadrature and sketch the actual current in each winding. Notice where the peaks occur in relation to the drive pulses.

That situation doesn't arise in a Thing-O-Matic because other factors limit the printing speed to 30 to 40 mm/s. Other 3-D printers can attain higher speeds, but only with much smaller moving masses and different motors.

The classic, pre-cheap-electronics solution to obtaining fast stepping reduced the circuit's time constant by increasing the overall resistance by adding an external-series resistor of three times the winding resistance. The time constant thus decreases by a factor of four, hence the circuit's name: an L/4R driver.

However, the supply voltage must increase by a factor of four to maintain the same winding current and produce the same motor torque, with the external resistor dropping 3/4 of the higher voltage. The schematic in Figure 3 shows that the resulting L/4R driver is essentially identical to the L/R driver in Figure 1. Because R2 is now so large, the MOSFET resistances become negligible by comparison.

The red current waveform in Photo 4 now resembles the waveform in Photo 2. The first full step after each transition remains starved, but the motor could run at nearly the same torque at the higher speed.

Pop Quiz: Determine how much R2 must increase to drive the Rated Current into the winding in 10% of a single full-step time. Use that resistance and the corresponding supply voltage in the ensuing discussion.

The blue trace in Photo 4 shows that the voltage across the winding now peaks at nearly twice the power supply voltage, much higher than the small peaks you probably ignored in Photo 2.

To see where those peaks come from, consider the situation in Figure 3 just before the driver switches the applied voltage at t = 4.8 ms. Ignoring the transistor resistances, terminal A1 sits at 0 V and A2 is connected to R2, so the winding carries a constant –340 mA (+ defined left-to-right), the

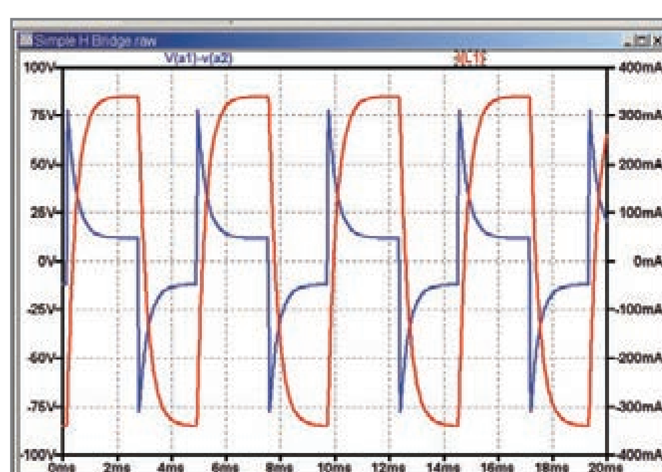


Photo 4—The large external resistance reduces the 48 V supply as the winding current increases, so the steady-state condition doesn't exceed the motor ratings. The external resistor will require a huge heatsink.

voltage across L1 is 0, and the voltage across R1 is –12 V (+ defined on the left). R2 carries 340 mA and drops 36 V.

At the transition, the switches connect A1 to R2 and A2 to ground. The winding current cannot change instantly, but the current in R2 can: the winding current now flows backwards through R2 *into* the power supply, which must be able to absorb reverse current. The voltage at A1 becomes 84 V (i.e., 48 + 36 V) with R1 holding the right side of L1 at –12 V (ignoring the transistor resistance), however, the voltage across L1 suddenly becomes 96 V (i.e., 84 – (–12) V).

That's exactly twice the full power supply voltage! The voltage across an inductor is related to its rate of current change, so the winding current begins changing at:

$$2200\ \frac{A}{s} = \frac{di}{dt} = \frac{V}{L1} = \frac{96\ V}{44\ mH}$$

After a few time constants everything settles into a steady state with the winding current flowing in the opposite direction.

Pop Quiz: Figure the dissipation for the resistance and voltage you calculated earlier.

Homework: Design an L/4R drive for a motor with winding resistance of 1.4 Ω and a current of 6 A. Estimate the physical size and cost of R2.

The reason nobody uses L/4R drives these days should be obvious: power efficiency. Because R2 has three times the resistance of R1 (the winding resistance), it will dissipate three times as much power. In this example, a 5 W winding dissipation dumps 15 W in R2 and gives an overall power efficiency of 25%. Obviously, there must be a better way!

CONTACT RELEASE

Contemporary stepper-motor drivers regulate winding current by turning the H-bridge MOSFETs into switching regulators with energy storage in the winding inductances. Exploring those circuits requires both simulation and actual measurements, which I'll have room for in the next column. ◼

Ed Nisley is an EE and author in Poughkeepsie, NY. Contact him at ed.nisley@ieee.org with "Circuit Cellar" in the subject to avoid spam filters.

PROJECT FILES

To download the LTspice IV simulation modules, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/253.

RESOURCES

Linear Technology, "Design Simulation and Device Models," www.linear.com/designtools/software.

E. Nisley, "Stepper Drive Part 1: Analog," *Circuit Cellar,* 169, 2004.

———, "Stepper Drive Part 2: Digital," *Circuit Cellar,* 171, 2004.

———, "Stepper Failure: Death By Disconnection," *Circuit Cellar,* 191, 2006.

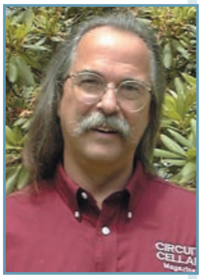
SOURCES

Allegro A3977 Microstepping DMOS Driver with Translator
Allegro MicroSystems, Inc. | www.allegromicro.com/en/Products/Part_Numbers/3977

42BYG034-4.78 and 1123029 Stepper motors
Kysan Electronics | www.kysanelectronics.com

Thing-O-Matic 3-D printer and Stepper driver
MakerBot Industries | http://wiki.makerbot.com/thingomatic

by Jeff Bachiochi (USA)



Vehicle Diagnostics (Part 3)

ISO 15765-4 (The CAN Protocol)

The first and second parts of this series introduced you to OBD-II and the pre-CAN protocol. This final installment focuses on ISO 15765-4 (the CAN protocol) and wraps up with some speculation about the future of OBD-III.

Two guys are standing in a driveway in front of a parked car. One dials up his wife on his cell phone. She is about to board a plane and the husband says, "Do it again, Honey." She replies, "Last time." As the scene flips back to the driveway, the car doors' locks are popping up and down and the horn is beeping. I can't even tell you what type of car this was a commercial for! And it's debatable just how handy this little trick is, but it tells you a lot about the future of the automobile.

If you have purchased a new car in the last few years, your vehicle uses the industry-wide automotive networking standard ISO 15765-4. A couple of months ago, I began this series by explaining the ODB-II protocols developed for this industry. At the time, each manufacturer had its own proprietary network. OBD-II gathered these together in a single connector and set its sights on 2008, when all manufacturers would phase out their old protocol in favor of one standard. Because so many pre-2008 cars are still on the road today, last month I continued by explaining how you can talk with your car via one of the protocols in use prior to 2008. General Motors, Chrysler, Ford, and many imports used either one of the two SAE J1850 protocols, VPM and PWM, or ISO 9141-2, a LIN-type UART-based protocol (or its offshoot, ISO 14230).

The circuitry presented in the first part of this series can be used to speak through the OBD-II connector to any of the early dialects. I am presenting this series to help you learn about each protocol so you can write your own code to interface with your vehicle. I have emulated many of the commands handled by the ELM327 OBD-II microcontroller. This microcontroller has been around for more than 15 years and has become a standard on its own.[1] This circuitry is compatible with the ELM327. In a standalone application, you may not need user I/O, but, for experimenting, a user interface is a necessity. Therefore, I chose to emulate the ELM interpreter. You can connect up your favorite terminal program and begin a conversation with your car.

CAN

This month I will finish up with the CAN protocol. While the CAN format can be produced like the previous protocols, without any special hardware, many microcontrollers offer a CAN peripheral as part of their standard lineup. Originating in 1983 by Robert Bosch, this protocol has

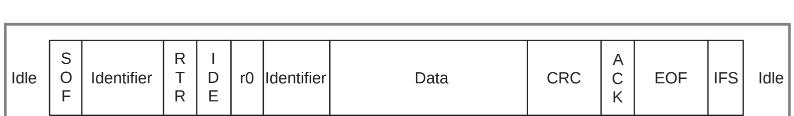


Figure 1—The CAN packet format. Note that the CAN packet frame is more complicated than just a header, data, and checksum. The microcontroller's CAN peripheral handles all the packet assembly.

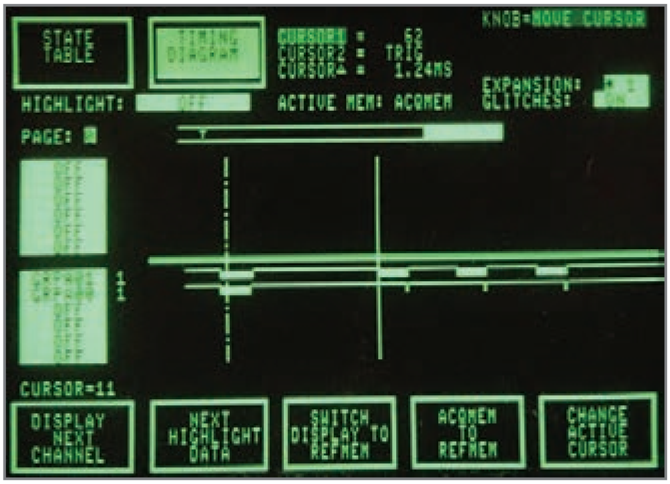


Photo 1—This sample shows a command and responses as seen from the RX and TX lines of the CAN driver chip used to interface with the OBD-II CAN bus.

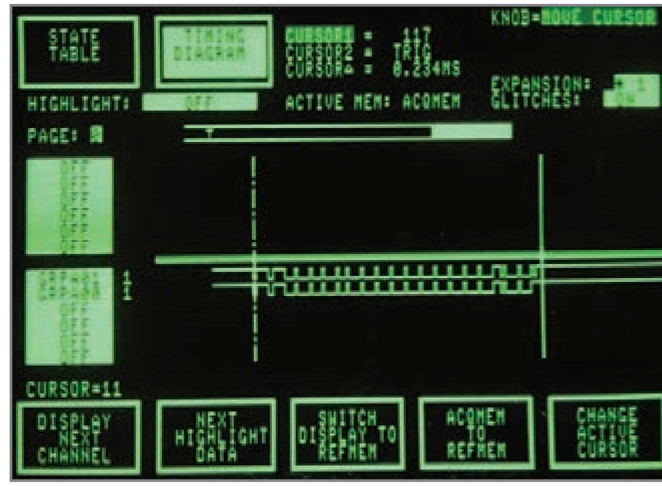


Photo 2—This view of just the command packet shows the active and passive states of the transmission.

been in use for many years. The CAN physical layer has lent itself to a number of application-layer protocols, such as DeviceNet, CANopen, CAN Kingdom, SafetyBUS p, MILcan, and many others.

The packet format for CAN contains three basic parts, a header, data, and a checksum. As shown in Figure 1, it is a bit more complicated than that, but the CAN peripheral does all of the formatting for us, so we only need to deal with the identifier and the data. Even the CRC checksum is calculated as necessary.

The synchronous bit timing uses non-return-to-zero (NRZ) coding and decoding with each bit broken down into multiple segments, sync, propagation delay, and phase buffer 1 and 2. This enables the receiver's sampling point to be adjusted within each bit. Because there is no clock transmitted, bit stuffing ensures there is a transition at least every 5 bits (as a way for receivers to keep in sync

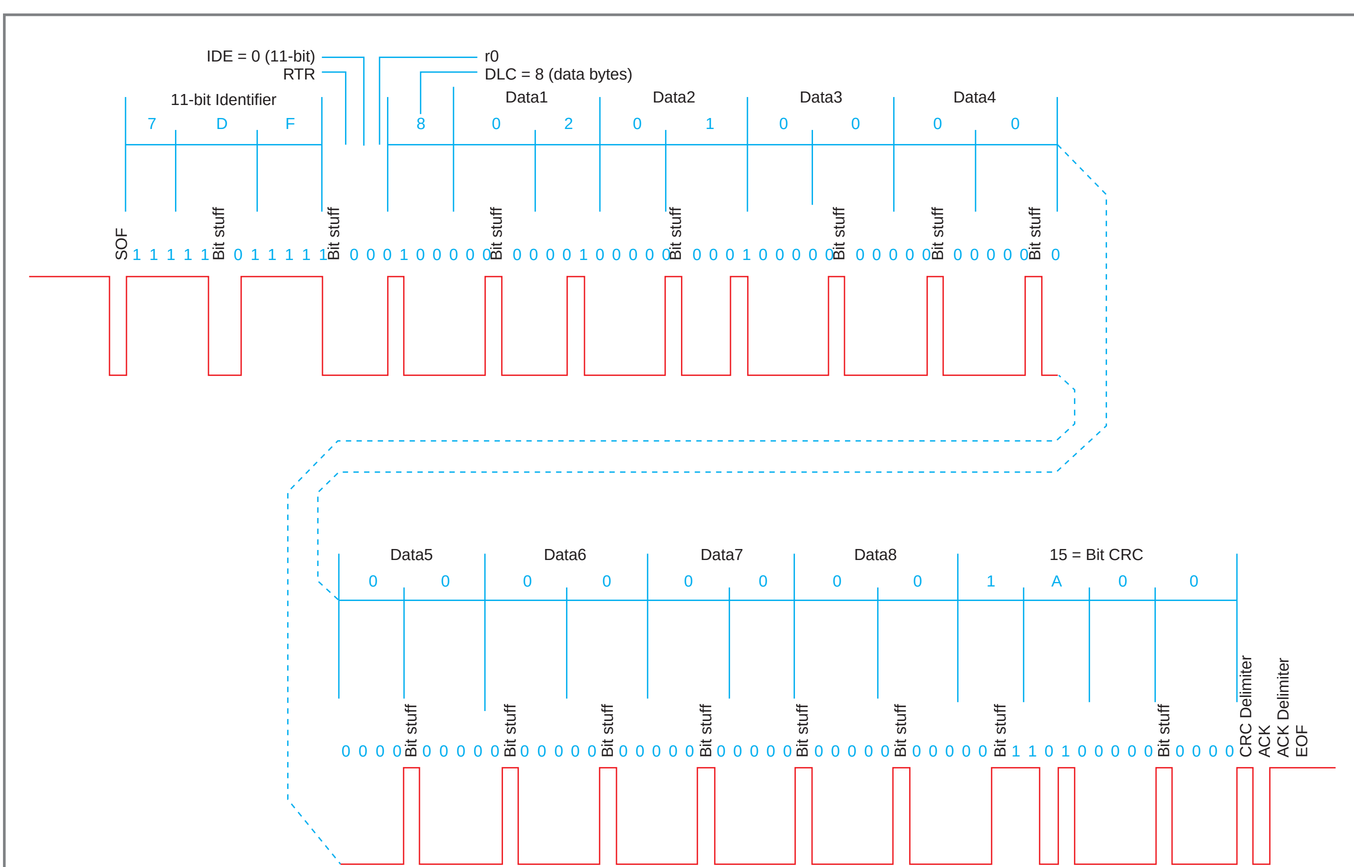


Figure 2—This diagram illustrates the breakdown of the command packet which includes bit stuffing used to keep receivers in sync with the transmitter.

with the transmitter.)

We tend to be conservative. In the interest of saving space and time, we often end up with complications due to our short sightedness. After the initial implementation of CAN the 11-bit address was found to be lacking. Modifications to the format allow for an expanded address of an additional 18 bits. So, we end up with two possible CAN formats. Again, the hardware takes care of all of that if we tell it which format to use. Although CAN can operate up to 1 mbps, the OBD-II specifies either 250 Kbps or 500 Kbps. This then defines the ODB-II CAN specifications as either an 11- or 29-bit ID running at either 250 or 500 Kbps.

The differential 5-V CAN bus driver idles with both outputs undriven. An active transmitter pulls the CANH node high and the CANL node low. Receivers determine an active state by the differential voltage between CANH and CANL. Therefore, like all the other formats discussed, an active state takes precedence over a passive state. All transmitters must monitor the bus during their transmission and stop transmitting if they see an active state while transmitting a passive state.

Photo 1 shows a CAN receiver (top trace) and a CAN transmitter (bottom trace) requesting a Mode 1, PID 0 command. You can see the transmitter's request and the three engine control units (ECUs) responding with data. Note the requestor ACKing each response. You cannot see the ACK on

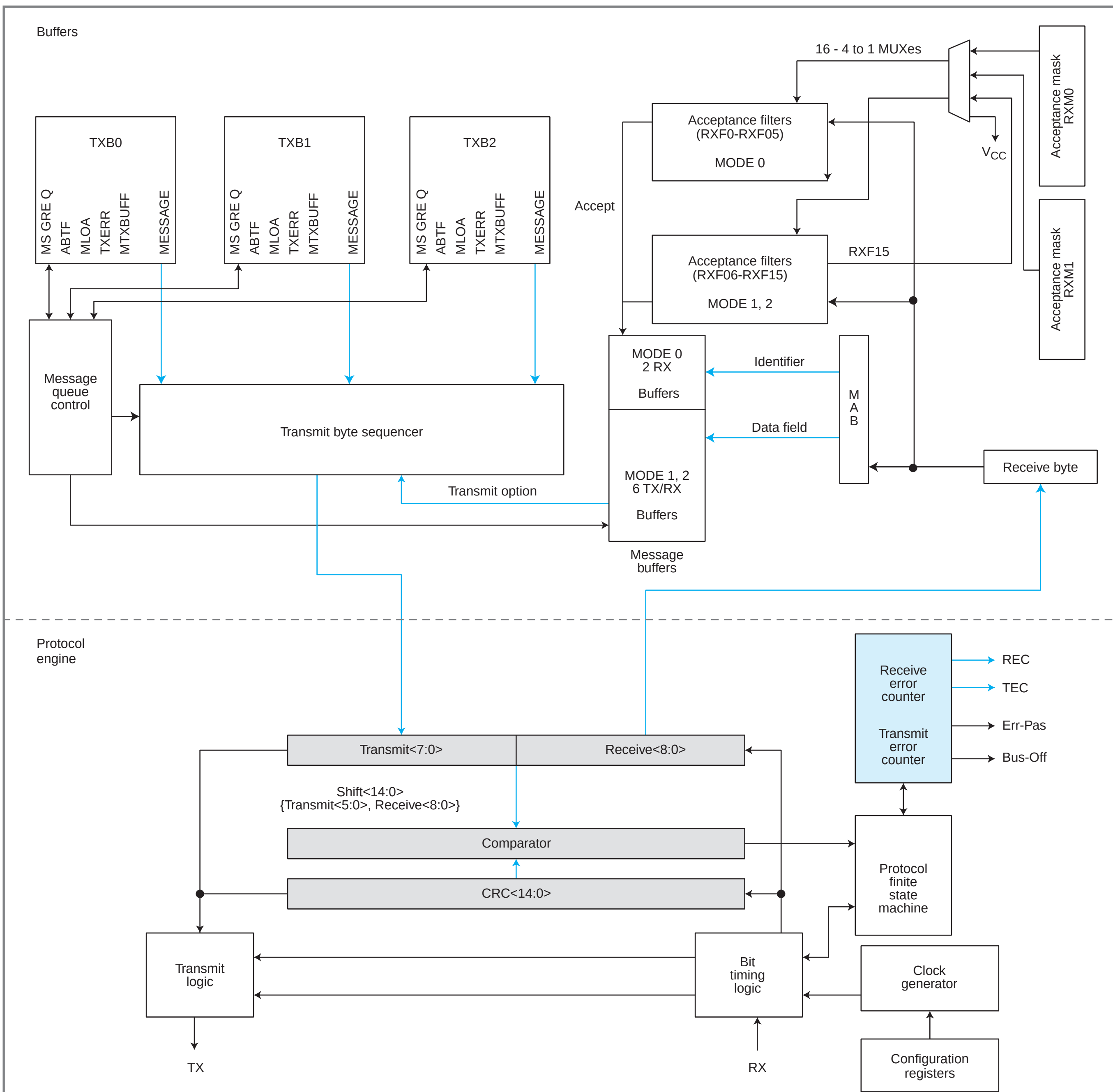


Figure 3—This block diagram shows many parts of the CAN peripheral required to transmit and receive packets.

the request because of the resolution chosen to enable the display of both the command and responses. However, you can see the ACK at the end of the request in Photo 2. This photo shows just the request and you can clearly see each active and passive state.

I briefly discussed bit stuffing, which forces a periodic data edge to keep the receivers in sync with the transmitter. Since there is no clock transmitted, let's see just how that works in this command request. In Figure 2 I've reproduced the signal from Photo 2 and labeled each part of the CAN packet. Note that after five consecutive unchanged states a state change is added by the transmitter, I've labeled these "bit stuff." All receivers know this rule and automatically throw out the "bit stuff" bit. Note that the bit stuff can be either an active or passive state. If you missed the ACK, take a look at the end of the packet. This is what a receiver is seeing. Note back in Photo 2 that the transmitter is requesting a passive state and wants to see an active state (from a responder). Indeed, a responder added this.

PLUG IT IN

Last month I explained the interface and we got started with an example of how to ask for the VIN of my 2000 Dodge Caravan. An initial request of "0100" was sent to the functional address 0x6A. Every ECU in the car will respond to this functional address as a general call. Each will respond with the requested information as it pertains to itself. The first byte, 01 (0x01), the Mode byte, is asking for some general information. The second byte, 00 (0x00), the PID byte, is requesting which other PIDs are supported. Each mode, 1 to 10 (0x01 to 0x0A), has specific PIDs associated with it. (You can find an extensive list of PIDs used with various modes at en.wikipedia.org/wiki/OBD-II_PIDs.)

To get the VIN, we use Mode 9. As in the previous example, 0100, we can use "0900" to request the supported PIDs for Mode 9. When "0900" was requested from my Caravan, the return "3C 00 00 00" indicated that the VIN (PID 0x02) was not supported. Starting from the left, the binary equivalent of 3C = 00111100. Again, from the left, zeros indicate PIDs 1, 2 ,7, and 8 are not supported, but 3, 4, 5, and 6 are supported. The following 3 "00" bytes indicate PIDs 0x09 to 0x20 are also not supported. Note: PID 0x0A (if supported) is the name of the responding ECU.

As mandated, my wife's 2009 Prius uses the CAN protocol. Let's try the same example on this vehicle:

```
>0900

7EA 06 49 00 FC 00 00 00 00
7EB 06 49 00 30 00 00 00 00
7E8 06 49 00 3F 00 00 00 00
```

Three ECUs have responded. From online searches I found out that these are identified as HV ECU 0x7E2, HV Battery 0x7E1, and Engine ECU 0x7E0 (each responds with most-significant bit of the least-significant byte set). The request, Mode 9 PID 0, has asked the general population if they support any of the PIDs, 1 to 32. ECU 0x7E2 has responded with "06" significant bytes "49 00 FC 00 00 00." The last four of these bytes indicate PIDs 1 to 32. The "FC" (0b11111100) indicates the first six PIDs are supported. We are interested in PID 2, the VIN. So we issue the VIN PID command:

```
>0902

7EA 10 14 49 02 01 4A 54 44
7EA 21 4B 42 32 30 55 39 39
7EA 22 37 38 33 39 30 30 35
```

The HV ECU 0x7E2 has responded with three packets. The first byte in these packets is an indication of the proper order of the packets. The high nibble indicates the frame type (1 = first, 2 = consecutive frame) and the low nibble indicates the proper order. Actually, the zero in the low nibble of the first frame is part of the next byte (bits 8 to 11). This second byte in the first packet is the number of significant data bytes, in this case 0x014 or 20 (it could up to 4,095!). The first and second significant data bytes "49 02" are a reflection of the command sent (bit 6 of the Mode byte is always set). The next significant data byte, 0x01, indicates the occurrence of the data (should there be more than one available). Seventeen significant data bytes follow, "4A 54 44 4B 42 32 30 55 39 39 37 38 33 39 30 30 35."

Configuration		Transmit			Receive		
Registers	Total	Registers	Groups	Total	Registers	Groups	Total
ECANCON	1	TXERRCNT		1	RXERRCNT		1
COMSTAT	1	TXBxD0:7	x=0:2	24	RXBxD0:7	x=0:1	16
CIOCON	1	TXBxDLC	x=0:2	3	RXBxDLC	x=0:1	2
BRGCON1:3	3	TXBxEIDH:L	x=0:2	6	RXBxEIDH:L	x=0:1	4
BIE0	1	TXBxSIDH:L	x=0:2	6	RXBxSIDH:L	x=0:1	4
BSEL0	1	TXBxCON	x=0:2	3	RXBxCON	x=0:1	2
MSEL0:3	4	TXBIE		1	RXMxEIDH:L	x=0:1	4
SDFLC	1				RXMxSIDH:L	x=0:1	4
					RXF0EIDH:L	x=0:5	12
					RXF0SIDH:L	x=0:5	12
					RXFCON0:1		2
					RXFBCON0:7		8
					RXFxEIDH:L	x=6:15	24
					RXFBSIDH:L	x=6:15	24
		Shared registers					
		BxD0:7	x=0:5	48	BxD0:7	x=0:5	48
		BxDLC	x=0:5	6	BxDLC	x=0:5	6
		BxEIDH:L	x=0:5	12	BxEIDH:L	x=0:5	12
		BxSIDH:L	x=0:5	12	BxSIDH:L	x=0:5	12
		BxCON	x=0:5	6	BxCON	x=0:5	6

Table 1—The registers associated with the CAN hardware peripheral can be divided into three categories: Configuration, Transmit, and Receive. Multiple groups of Transmit and Receive buffers add flexibility but muddy the waters for those requiring only basic operations.

a)

S(tandard) ID(entifier) H(igh byte)								S(tandard) ID(entifier) L(ow byte)							
10	9	8	7	6	5	4	3	2	1	0	x	IDE=0	x	x	x

b)

S(tandard) ID(entifier) H(igh byte)								S(tandard) ID(entifier) L(ow byte)							
28	27	26	25	24	23	22	21	20	19	18	x	IDE=1	x	17	16

E(xtended) ID(entifier) H(igh byte)								E(xtended) ID(entifier) L(ow byte)							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 4a—The CAN 11-bit address is left-justified in the SIDH:L 16-bit register with an IDE bit=0. b—To extend the format to 29 bits, the IDE bit=1 and the most significant two bits of the extended address are found in the least significant two bits of the SIDH:L register, while the remaining extra 16 bits are in the EIDH:L register.

These are ASCII representations of the VIN of Beverly's Prius, "JTDKB20U997839005."

CAN REGISTERS

Last month's protocols were all handled in software. These require a large portion of execution time to support transmitting and receiving data. When you can take advantage of a peripheral that will handle the protocol, it really frees up the microcontroller. While a peripheral can initially seem complicated, much of the complexity is often added to increase flexibility. The block diagram of the CAN protocol engine available in a Microchip Technology PIC18F2480 microcontroller and its hundreds of registers looks quite intimidating at first glance (see Figure 3). Divide and conquer to make things more manageable.

The CAN registers can be divided into three categories, Configuration, Transmit, and Receive registers (see Table 1). Configuration (and Status) registers are used to set up operational features and monitor any interrupts that occur. The Transmit registers include three standard groups of registers and six shared groups. Any group can be used to send a CAN packet. You may need to use only one. Receive registers include two standard groups of registers and six shared groups. Multiple receive registers are handy when bus activity is high. This enables you to handle one packet as the next is being received. There are a number of other registers associated with the receive buffers that have not been discussed prior to now. These are the Filter and Mask registers.

The Filter and Mask registers are applied to all received bus packets enabling unwanted packets to be tossed out. This eliminates having to handle packets that are of no interest. Masks and Filters look at only the header, which can be either 11 bits or the 29-bit identifier (the IDE bit determines which format is being used). This means that each Mask and Filter register group consists of four registers, EIDH:L and SIDH:L. This format enables them to be easily compared with the EID and SID registers in the message assembly buffer (incoming packet). If all Mask bits = 0, then the message is automatically accepted. Any Mask bit that = 1 requires further comparison with the associated bit in the Filter register. If all associated bits in the Filter register match those in the MAB the message is accepted.

The most difficult part of using the CAN peripheral might be translating the chosen 11- or 29-bit identifier. The SIDH:L and EIDH:L registers are configured in the format of the packet. For an 11-bit identifier, the SIDH:L holds the 12 bits of the header in the format (see Figure 4a). The 29-bit format adds an additional 19-bits in the format (see Figure 4b). For CAN 11-bit transmissions to address 0x7DF, the SIDH:L become 0xFBE0. For 29-bit transmission to address 0x(18)DB33F1, the SIDH:L EIDH:L becomes 0xC6CB33F1.

THE MECHANIC'S FRIEND

No auto mechanic I know cares what the engine control modules address is. They use OBD-II to regurgitate diagnostic trouble codes (DTCs). These Mode 3 DTCs indicate what part of the vehicle is out of specification. It is important to remember that they are an indication of some area where a measurement is not what it should be. It is still left up to the mechanical detective to determine the actual cause. It could be a sensor, the electrical connection, or an ancillary part, such as a cracked hose, all of which could provide the same DTC.

When a repair has been completed, the DTC can be eliminated from short-term memory using Mode 4. The problem is considered fixed if it has been cleared using Mode 4 and, after performing a verification test, the DTC does not return.

Every DTC has a procedure that will enable the vehicle to retest itself for correct operations. If all remains within specifications, no DTC will result.

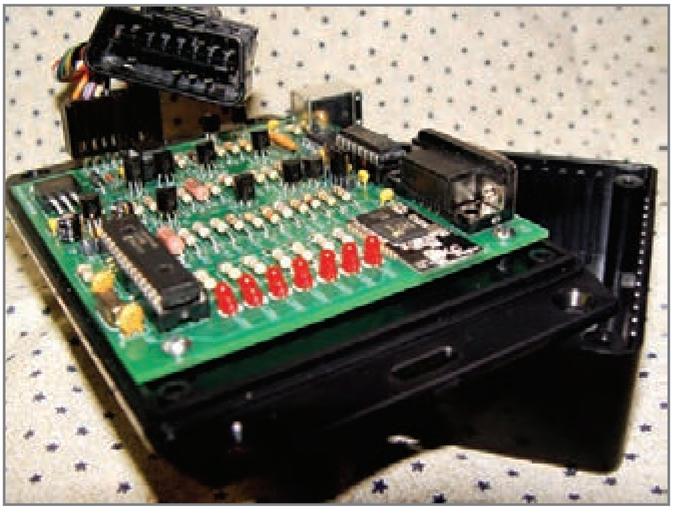


Photo 3—This project offers experimenters access to their vehicle's private diary via the industry-mandated OBD-II connector.

Repair and maintenance manuals are available for all vehicles from your dealer. Third-party manuals can be purchased at bookstores and online. These manuals have extensive DTC lists and procedures to follow to help track down, repair, and retest each.

SERIAL PORT

The circuit presented in Figure 4 of my *Circuit Cellar* 252 article implied a user connection via a serial port (see Photo 3). However, I purposely left out this part of the schematics until now. Figure 5 shows the standard RS-232 port and a couple of alternatives.

Since many computers have now eliminated the DB25/DB9 serial ports, I've included a USB connection that looks like a virtual COM port to any PC application. Microchip Technology's MCP2200 USB-to-UART serial converter is a great interfacing chip that can give your device a USB connection using just three lines, TX, RX, and ground, from your microcontroller. This chip requires no special USB driver and no need for hardware handshaking.

For those of you who are going completely wireless, I've added a small Bluetooth module that can also look like a virtual COM port. This module from KC Wirefree is a 3-V part so there is some circuitry added to level shift between 3 and 5 V. This great little module is $35 and has its own AT command set. You can use the AT commands from the PC end through the Bluetooth connection to configure the KC module for the proper baud rate so it can properly connect to the microcontroller. Be advised that many times Bluetooth and USB virtual ports are created with higher COM numbers and some PC applications have limits on the COM numbers available to the application. You may need to reassign the COM to a lower number using the advanced option in the properties of the COM port via the device manager.

Jumper JP1 selects which serial device can talk to the microcontroller (see Figure 5). All serial devices listen to the microcontroller. This means you can connect to all three interfaces at the same time and see the microcontroller's response to the commands sent by the selected interface.

This is more of an interface-debugging tool than a useful function.

PLAYS WELL WITH OTHERS

Unless you have your design lab in your garage, you might find it annoying to run back and forth between your code changes and the OBD-II connector of your vehicle. You might try to develop code on your laptop while sitting behind the wheel in your car, or you can bring the car into your lab. While this second option might sound foolish, how about just bringing in the ECU from under the hood? Or, get one from a junkyard. Or, purchase an OBD-II simulator.

ScanTool.net makes ECUsim 2000, a small simulator for around $300 including a single protocol. Additional protocols can be enabled (purchased) for the same device. This product will look like a vehicle with three ECUs, an engine control module (ECM), a transmission control module (TCM), and an antilock braking system (ABS) module.

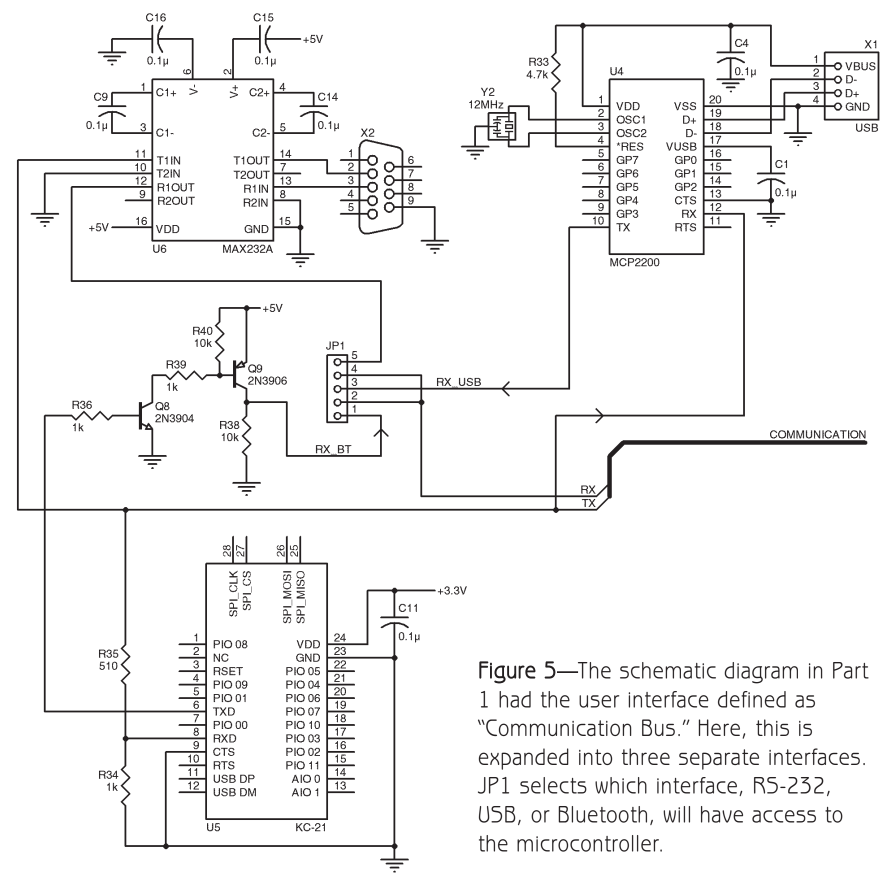


Figure 5—The schematic diagram in Part 1 had the user interface defined as "Communication Bus." Here, this is expanded into three separate interfaces. JP1 selects which interface, RS-232, USB, or Bluetooth, will have access to the microcontroller.

The emulator has three analog pots that can vary the values read by certain PIDs. Other PID values have fixed values. You can even emulate a fault condition via a push button.

ScanTool.net has some other unique and handy components that you will find helpful. You can get cables and entire ODB-II interfaces on small plug-in modules that can make experimenting less of a hassle for those unwilling to make their own PCBs and collect all the interfacing parts. I suggest you visit the Resources at the end of this column to increase your knowledge of OBD-II beyond what I've been able to provide here in this series.

INTO THE FUTURE

I suspect it will be a while until we have a people delivery system in place, similar to those pneumatic tubes featured on the science fiction sitcom, *Futurama*. Until then, we will need to rely on vehicles to move us around. Are electric- or hydrogen-powered vehicles the answer to higher oil prices or, worse yet, embargos? I hope we are able to obtain some level of independence before we are brought to our knees.

What about the future of OBD-III? It's obvious that wireless is already being flaunted. I hope OBD-III isn't bogged down by Big Brother surveillance issues. As in the past, the EPA is interested in how OBD-III can minimize the delay between the detection of an emissions malfunction by OBD and the actual repair of the vehicle. One thought is to include this monitoring into the Fast Lane toll collection system that is already in place across many states in the Northeast. Alternative cellular- or satellite-based communications open the door for services that can save your life, as in automatic response to crash data and GPS location. On the negative side, our driving patterns could be tracked or our vehicle could be disabled at any point in time.

For now, the required emission testing on each vehicle every two years will continue to be a hassle for me. That's more from the inconvenience than the $20 fee. I would welcome technology that would enable emission data to be collected and verified automatically, eliminating the time wasted for testing. Now, if only this could be applied to other areas of preventative maintenance, like preventing annual visits to the dentist! ■

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

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/253.

REFERENCES

[1] Elm Electronics, Inc., "ELM327 OBD to RS232 Interpreter," ELM327DSH, 2010, www.elmelectronics.com/DSheets/ELM327DS.pdf.

RESOURCES

Elm Electronics, Inc., "AN02 – ELM327 Circuit Construction," AN02 rev A, www.elmelectronics.com/AppNotes/AppNote02.pdf.

ScanTool.net, OBD-II Scantools, www.scantool.net/scantool/downloads/101/ecusim_2000-ug.pdf.

SOURCES

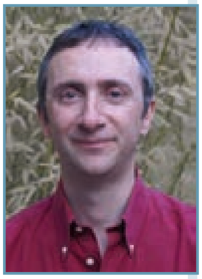
ELM327 OBD-to-RS-232 Interpreter
Elm Electronics, Inc. | www.elmelectronics.com

Bluetooth modules
KC Wirefree Corp. | http://kcwirefree.com

PIC18F2480 Microcontrollers and MCP2200 USB-to-UART Serial converter
Microchip Technology, Inc. | www.microchip.com

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

by Robert Lacoste (France)



RF and High-Speed PCBs

How to Avoid Basic Mistakes

The principles of RF and high-speed PCB design are fairly simple, but there are some common mistakes that can ruin a project. This article details proper PCB routing practices and topics, including ground structure, impendence matching, and component placement.

When consulting, I occasionally encounter a customer who has a "99% finished product" and needs help fixing a few remaining problems. Unfortunately, it is not unusual that the issues hidden in the last percent of the "finished prototype" require a full redesign. In particular, one of the most common mistakes is hiring a printed circuit board (PCB) designer with no high frequency experience to route a PCB for either a wireless device or a high-speed digital board. If you are lucky, the results will be bad performance, random reliability, or regulation infringement. But, more often than not, you will end up with a product that does not work.

The basic rules for rock-solid RF PCB design are easy to understand. Moreover, the same rules apply for high-speed logic designs: "High speed" implies fast signal edges, which means high frequencies everywhere. It's always better to avoid doing a job twice. So, with this column, I plan to make your life easier by presenting some examples of good and bad PCB routing practices.

GROUND, GROUND, GROUND

One of the most common problems in PCB design, and also system design, is the lack of a good ground structure. Just take your last project's schematic, select any grounded node, and imagine that it is no longer actually connected to ground but connected to a voltage source of a few volts. I bet your project wouldn't work anymore, right? Such a situation is unfortunately common for poorly designed PCBs. When you manipulate high-frequency

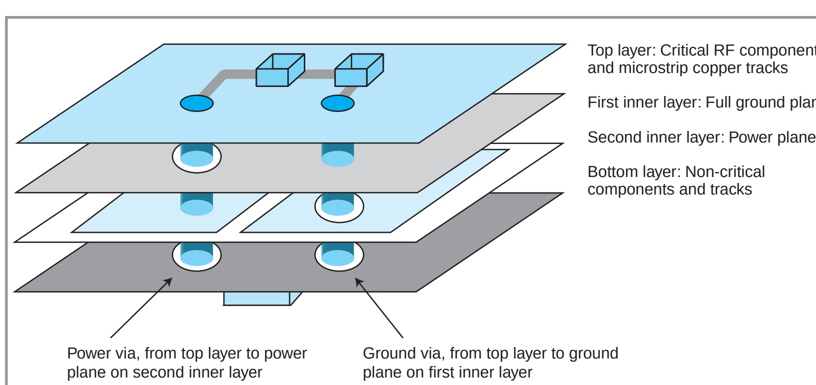


Figure 1—Using a four-layer PCB is the easiest solution for a good grounding structure. Usually, the first inner layer is devoted to a full ground plane. The top layer, just above the ground plane, is used for all critical RF components and microstrip tracks. The second inner layer is used for power supplies. And lastly, the bottom layer can be used for less critical components and tracks.

signals, any conductor acts as a transmission line. The voltage between its ends is no longer null because its impedance, at the working frequency, is not very low.

So, as a general rule, as soon as the ground level is not well defined, things become difficult. There is only one solution: do your best to have a rock-solid common ground. And, on the PCB, spend a couple of extra dollars and use at least a four-layer PCB. A four-layer PCB enables you to devote one of the inner layers to a full ground plane (see Figure 1). This is the best ground you can imagine: a full, plain sheet of copper, as large as your PCB, ensuring minimal impedance between a couple of grounded points. Never break this ground plane by routing any small track on this ground layer. There should only be a ground plane—period. As soon as you remove copper on this ground plane, you introduce parasitic impedances on neighbor tracks. On such a four-layer design, usually the side of the PCB closest to the ground plane is used to place all RF components using microstrip techniques. More on that later. The opposite side is used for less critical components. And lastly, the second internal layer is used for power supplies, using power planes as large as possible to minimize their impedance as well. This has the added advantage of implementing zero-cost filtering capacitors because the power planes and the ground plane are facing each other.

To reduce costs, or if you plan to manufacture your PCB in your kitchen, you might want to stick with a double-sided PCB. This is possible, of course, but it will make your life more difficult. When you need to route tracks on both sides of the PCB in the same area, you cannot ensure a good ground plane. The only solution is to implement ground planes as large as possible on both sides, interconnected by plenty of vias. This is a nontrivial task, and usually the best solution is to try to avoid any bottom-layer tracks below the most critical RF sections. An example is shown in Photo 1.

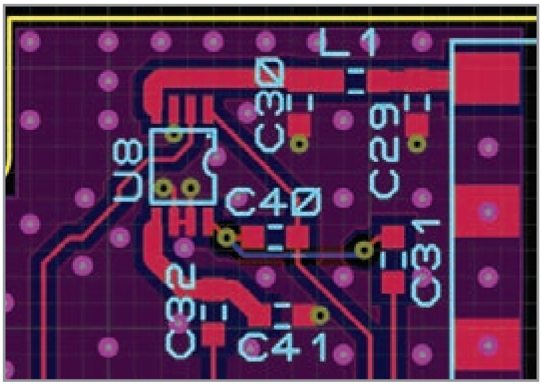


Photo 1—If you must use a double-sided PCB, your life will be more complex, as the ground plane will be shared between top and bottom layers. You need to ensure that there is at least a full ground plane under the most critical sections, using top-side routing as much as possible and only a couple of tracks on the bottom side. Plenty of interconnecting vias are needed to connect the top and bottom grounds. Bottom tracks (shown in blue) should never cross the wider RF tracks on the opposite side.

One more point on ground planes. From time to time, you will find articles recommending the use of split ground planes—for example, with a ground plane for the logic sections and a ground plane for the analog components, interconnected at a single point. The idea is to minimize noise through the ground planes. Unfortunately, it is difficult to properly implement such a concept. In particular, it is mandatory to route all tracks going from one region to the other exclusively above this interconnecting point. If not, you are constructing a good antenna that will either transmit or receive spurious signals (see Figure 2). Personally, I am convinced that 99% of the time, a full unique ground is more reliable and gives better results than split grounds, as long as the placement of the components is adequate—except, maybe, for some audio projects. More on that later.

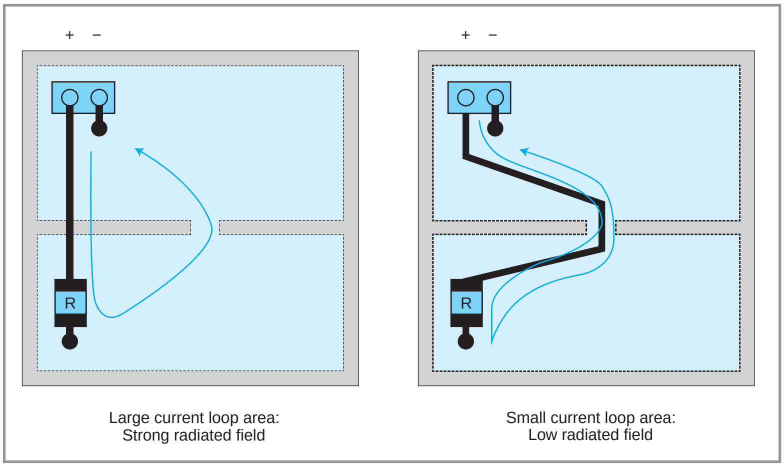


Figure 2—A split ground plane is usually a bad idea, except if there is a specific need for one, such as strong ESD risks. Any track going from one ground area to the other should cross the boundary just under the interconnecting point (right). If not (left), you will have a current loop, which means EMC problems.

TRACK IMPEDANCE & REFLECTIONS

I have already devoted a full column to microstrip techniques ("Microstrip Techniques," *Circuit Cellar* 223, 2009), so I will not expound on that topic again. Remember the basics: If you route a high-frequency or high-speed signal, you must take care of impedance matching. If not, you will get signal strength losses, and more importantly nasty reflection phenomena. (Refer to my column, "Time Domain Reflectometry: Detect and Measure Impedance Mismatches," *Circuit Cellar* 225, 2009.)

In order to have impedance matching, the receiver impedance must be the complex conjugate of the source impedance. They must be identical if the impedances are purely resistive and the characteristic impedance of the interconnecting line must be well selected. In the vast majority of RF designs, all subsystems have a 50-Ω impedance; therefore, the PCB tracks must have a 50-Ω characteristic impedance if you want to avoid reflection. This translates to a precise PCB track width, depending on your PCB technology. The two most standard solutions are either the microstrip technique, meaning routing the RF track just above a full ground plane, or stripline, where the RF track is sandwiched between two ground planes. The most common track widths in both techniques when using a standard FR4 laminate are shown in Figure 3.

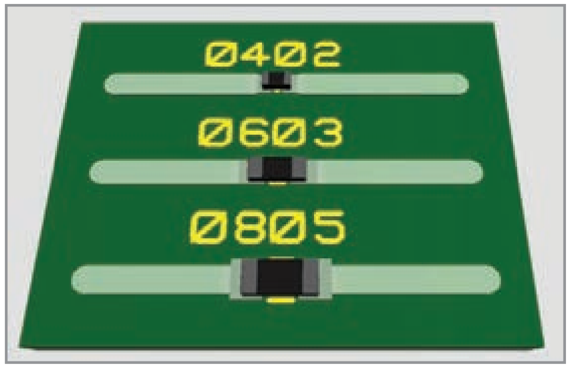


Photo 2—Impedance mismatches will be reduced if you select components with packages roughly the same size as the calculated track width. Here a 0603 package will be a good choice for this 0.9-mm microstrip track.

These widths are usually not very critical, meaning that a 5% error will not change your life, but you must at least have the good order of magnitude in mind, and you must double check the height between the tracks and the ground plane with your PCB supplier. Lastly, track width is only a concern when the track length is longer than a tenth of the wavelength at the working frequency. For example, at 2.4 GHz the wavelength in the air is c/f, with c equal to the speed of the light in free space, which is:

3.10^8 m/s and f = $2.4.10^9$ Hz giving L = 125 mm.

A tenth is 12.5 mm. However, here the signal is not in the air but on a PCB, and you should take into account that the signal speed will be lower than the speed of the light. It is reduced by the square root of the relative dielectric constant of the material, which is 4.3 for FR4. Therefore, the final critical length is:

$$\frac{12.5\text{ mm}}{\sqrt{(\Sigma_R)}} = 6\text{ mm}$$

At 2.4 GHz at least, any track longer than 6 mm must have the correct width.

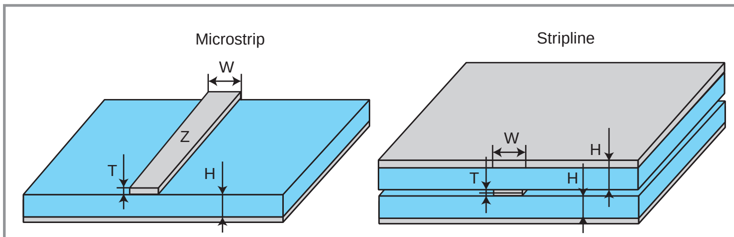
SCHEMATIC & COMPONENTS SELECTION

The PCB design starts with the schematic and, more specifically, with the selection of the components. Of course you should use surface-mount devices (SMDs), since smaller components and shorter wires mean better RF performances. But, which package size should you select, for example, for all the passive resistors and capacitors? You may select the largest ones, as they are easier to solder, or the smallest ones, say 0201, to try and get the best performances. Usually, other criteria will help you make the best choice. You have just calculated the proper track width for a 50-Ω impedance. Intuitively, you will get the best performances if the width of the component is close to the track width (see Photo 2). This will reduce impedance matching problems between the track and the component pad.

You also need to add test points to your circuit if you want to test a subpart of the design—say, an amplifier or filter stage. This must be planned at the schematic phase, too. Usually, you will need to add extra components, such as test connectors, which might not be populated if not needed for production batches, or coupling capacitors whose removal enables you to open the circuit during testing. Just think twice about the resources you will need to test the board before drawing the PCB. Writing the validation plan as early as possible also helps.

You should also double check the power suppliers' schematics. It's a good idea to implement separate digital and analog power supplies, even just using a small ferrite bead to isolate them. If you have successive high-frequency amplifying stages, take care to decouple the power supply of these successive circuits. If not, you will probably get an oscillation somewhere.

Lastly, if you are assembling your PCB at home, you may have a problem with some RF integrated circuits that have a ground pad just under the chip. It would be a bad idea to have it unsoldered, as you will probably get disappointing performances or spurious oscillations. You can try to avoid using such chips, but it may be difficult. If you have no other solution, you can still solder it. The first solution is to use a homemade reflow oven (see my column, "Easy Reflow: Build an SMT Reflow Oven Controller," *Circuit Cellar* 168, July 2004). The alternative is to add a large plated through ground hole via under the chip. Put solder flux on the pads, place the component, and solder the top pads of the chip. Then flip the board



Microstrip: Z = 50 , T = 35 µm, FR4 substrate (ε_r = 4.6)

H	W
1.6 mm	2.95 mm
0.8 mm	1.45 mm
0.5 mm	0.9 mm

Stripline: Z = 50 , T = 35 µm, FR4 substrate (ε_r = 4.6)

H	W
0.8 mm	0.65 mm
0.5 mm	0.4 mm
0.35 mm	0.26 mm

Figure 3—A microstrip is a copper track just above a plain ground plane, whereas a stripline is sandwiched between two ground planes. The tables give you the track width for a 50-Ω characteristic impedance for classical FR4 PCB thicknesses, calculated with AppCad.

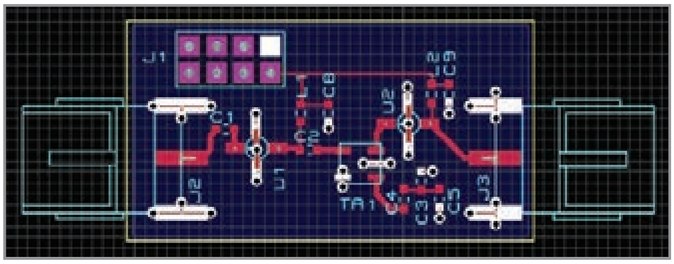


Photo 3—A CAD tool, such as Labcenter Electronics's Proteus PCB Design Software, can highlight all the pads connected to ground. Start by interconnecting each of them to the underlying ground plane with dedicated ground vias as close as possible to the component. A via costs nothing, and good grounding is priceless.

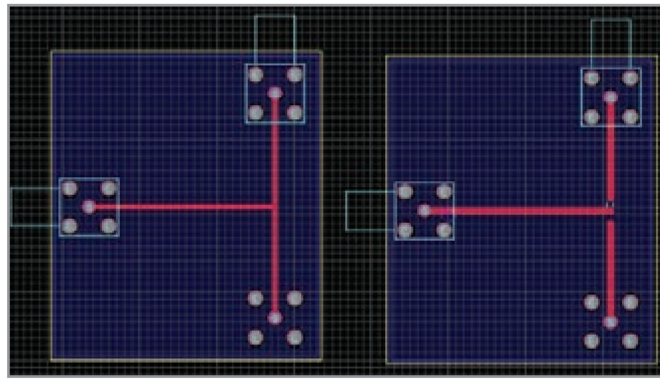


Photo 4—In this example, the same PCB can be populated with a connector either in the top or bottom position. The layout shown on the left is catastrophic because the unused open-ended track will act as a band-stop filter at any frequency for which it is an odd number of quarter wavelengths long. On the right, this improved design solves the issue with a small coupling capacitor or 0-Ω resistor installed in either the top or bottom position.

and put solder inside the ground hole using a powerful iron. If you are lucky, and if you have sufficiently heated the chip, the ground pad will be soldered and you will avoid short circuits.

THE PLACEMENT PHASE

High frequency problems are drastically minimized when the tracks are short and straightforward. Therefore, the placement of the components on the PCB is really the most critical phase of PCB design. Experience shows that you must spend as much time on placement as on routing. Personally, I always keep a printed version of the schematics in front of me when placing the components. You should be able to "read" the schematic when looking at the finished PCB.

Start by segmenting the different areas of the PCB. Where are you going to place the digital sections? The critical RF sections? The power supplies? If possible, avoid any loopback risks. For example, don't place an RF output connector close to an input because this causes problems (see Figure 4).

When the placement strategy is defined you can start by placing the most critical components first. Try to simplify the routing of the RF or high-speed signals as much as possible. Don't forget to place their associated decoupling and peripheral components as close as possible. This is where looking at the schematic helps. Always try to rotate components, or to swap pins where possible, in order to limit the number of signal crossings. Each crossing will mean vias, which means potential impedance matching problems.

After all critical parts are placed, you can then place the less critical sections (e.g., digital control and power supplies). If you are using a four-layer PCB, a solution is to put all critical components on the top side and all decoupling and power supply components on the bottom. This ensures that the decoupling is as close as possible to the destination circuits and simplifies the routing. You can also put all components on the same side, but the PCB dimensions will be significantly larger.

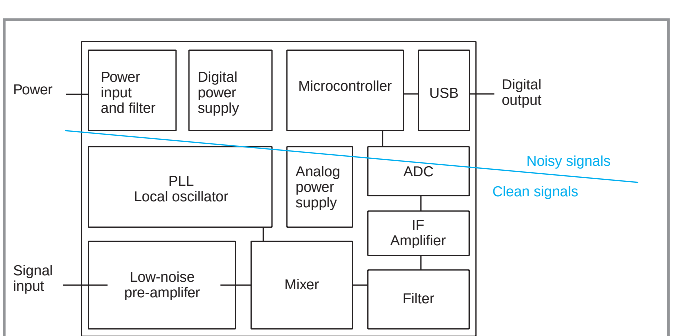


Figure 4—The first step in placement is to define a clear placement strategy based on the constraints of your project. Separating "dirty" areas and sensitive parts is key, as well as avoiding loopbacks between low-level inputs and high-power outputs.

GROUND CONNECTIONS & COMPONENTS

Now all your components are placed on the PCB and you are ready to start the routing. Even if you have high-end PCB auto-routing software, never click on the Autoroute button (at least not at that stage)! You are a good PCB designer and you know which tracks are critical. The tool doesn't, except if you have spent a lot of time to tell it what to do. Start by connecting all grounded pads to the ground layer. Your CAD tool may have a feature to highlight all pads connected to a given net. If so, use it. Refer to Photo 3 for an example using Labcenter Electronics's Proteus PCB Design Software. For each grounded pad, add a ground via as closely as possible to the pad, going directly to the inner ground plane for a four-layer design or to a grounded copper area on the other side for double-sided design. Never use tracks to route a ground: remember that the inductance of a 10-mm track is around 1 nH. This may not seem like much, but the impedance of a 1-nH inductance is 12 Ω at 2 GHz, which is not a negligible value compared to 50 Ω. When using RF ICs with several grounded pins, avoid using only one ground via. Instead, use a separate via for each pin as well as some ground vias under the chip itself, as shown in Photo 3. This will give the lowest impedance to ground. In the same spirit, never use thermal relief when a component grounded pad is inside a ground plane: these

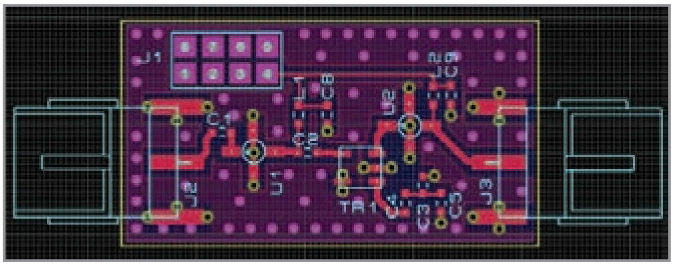


Photo 5—An additional ground plane can be added on the top and bottom layers, filling all free space. However, the clearance must stay large enough not to perturb the RF tracks, and it is mandatory to use regularly spaced ground vias to ensure a proper grounding structure. Densely spaced ground vias at the board outline avoid some RF leakage.

small tracks going between the pad and the ground plane facilitate the soldering but are marvelous small inductors in RF designs and should be avoided.

Next, route all critical RF or high-speed tracks, using the calculated track width as much as possible for good impedance matching, except if the track is very small (shorter than a tenth of the wavelength, as discussed). Usually, I then route the power supplies using power planes where possible and finish up the board with all the noncritical tracks.

Another common error is the routing of "tee" structures on the RF signals. Imagine that you have a PCB that could be populated in two versions, one where the RF signal goes to connector A and one where the RF signal goes to connector B (see Photo 4). If you route them without any specifications, the RF signal will be split into two tracks, one going to a connector and one that is open-ended. However, an open-ended wire is not at all negligible in RF. Transmission line theory shows that such an open circuit is in fact equivalent to a short circuit to ground at a precise frequency, for which the line length is a quarter of the wavelength, making it a resonator. This topic would require another column, but this shows you that it should be avoided. For example, a 40-mm open wire on an FR4 substrate will resonate at about 2.7 GHz. So, open-ended lines or tee structures should be avoided. The right side of Photo 4 shows you how.

ANCILLARY GROUND PLANES

Let's say you have a four-layer PCB with a full ground plane on one of the inner layers. Should you also use copper pour on the two external layers for additional ground planes? The answer is usually yes, as a grounded copper area between two tracks will help to decouple the signals. However, it must be handled with care. First, the clearance between the grounded area and the RF tracks must not be too small. If so, the proximity of the copper will change the impedance of the track. You will, in fact, move from a microstrip structure to a so-called coplanar waveguide (CPW). You could, but then you would need to recalculate the track width for a 50-Ω impedance. However, if your clearance is at least equal to the track width, you shouldn't have any problem.

The second issue is the interconnection of the different ground planes. It is not enough to interconnect them in a couple of points; you must spread ground interconnecting vias all around the ground planes for a proper equipotential structure (see Photo 5). As a rule, the distance between two ground vias must not exceed a tenth of a wavelength at the highest working frequency. If the highest working frequency is 2.4 GHz, you should have a via every 6 mm or so, as discussed in the beginning of this column. It is also fundamental to put ground vias all around the PCB edges. These vias, if closely spaced, will limit the RF leakage through the PCB laminate, especially if you take care to stop the power planes before this via line.

So, I've explained that even on a four-layer PCB the critical RF tracks must all be routed on the same player, above a full ground plane. But what should be done if two RF tracks need to cross each other? First, check twice. Is there another solution? Could you move a component or add a component to avoid this situation? If not, then you need to use vias. There are two solutions: either move one of the RF tracks to the opposite side, still using microstrip above the power plane and back, or use stripline. For the latter solution, the idea is to temporarily use the power plane as an RF plane and to route one of the two RF signals between two ground planes. Figure 5 shows you how to do that. With proper track widths and with plenty of ground vias around to limit the RF leakage through the laminate, you should get satisfactory results. The same technique can be used when using shield cans: when an RF track needs to exit a SMT-soldered shield can it can be routed through either the bottom layer or through an inner layer.

WRAPPING UP

Here we are. I know that you probably won't have to route a microwave PCB like the one shown in Photo 6

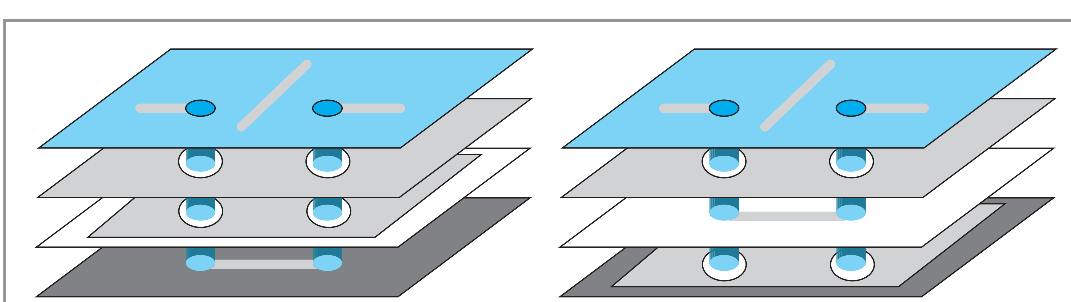


Figure 5—These figures show you two alternatives when you absolutely must have two RF tracks crossing each other: either through the bottom layer (left) or through a stripline on the second inner layer (right). In both cases, plenty of vias should be used to avoid RF leakages.

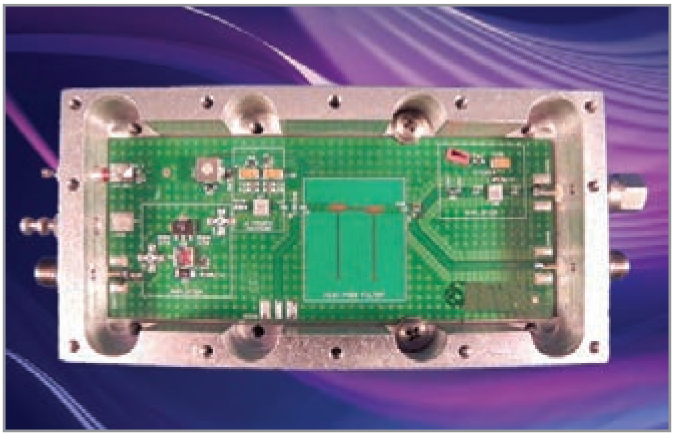


Photo 6—This is an example of a 10-GHz microwave board designed by my company for this column. Look at the density of the ground vias everywhere as well as the ancillary test point structure. Here the substrate is not FR4 but a higher-performance Rogers RO4000.

every week, but using these routing techniques never hurts. Moreover, you may need them more often than you think. Here's a recent example. One of my colleagues just finished a great high-speed FPGA-based project. All the critical parts—including amplifiers, ultra-high-speed ADCs, and buses going between no less than four Xilinx Spartan-6 FPGAs—were routed using controlled impedance techniques. The noncritical signals were routed with fewer constraints, including the JTAG programming lines. Everything worked, but there was no way to program the first FPGA of the JTAG chain even if we had no problem for the next three. What happened? A speed problem on the JTAG? We tried to reduce the JTAG speed to 100 KHz, but there was no improvement. We had to connect a high-speed sampling scope on the JTAG clock line using a high-impedance FET probe to discover the root cause. The clock signal generated by the JTAG probe had fast edges, even at low clock rates. Due to the complex board structure and improper impedance control, the JTAG clock signal got reflected back close to the last FPGA. This reflection was more and more visible the farther it went from the endpoint, meaning closer to the JTAG input. And, the first FPGA—and only that one—saw two clock pulses spaced by 1 ns for each JTAG clock transition. We didn't want to reroute the board, so we lowered the rise time of the JTAG clock with a small RC network and the problem was solved. We knew it, but it once again proves that good design rules must always be followed! Now, it's your turn!

Robert Lacoste lives near Paris, France. He has 20 years of experience working on embedded systems, analog designs, and wireless telecommunications. He has won prizes in more than 15 international design contests. In 2003, Robert started a consulting company, ALCIOM, to share his passion for innovative mixed-signal designs. You can reach him at rlacoste@alciom.com. Don't forget to write "Darker Side" in the subject line to bypass his spam filters.

RESOURCES

Altera Corp., "High-Speed Board Layout Guidelines," 2009, www.altera.com/literature/an/an224.pdf.

J. Ardizzoni, "A Practical Guide to High-Speed Printed-Circuit-Board Layout," Analog Devices, 2005, www.analog.com/library/analogdialogue/archives/39-09/layout.html.

R. Hartley, "RF/Microwave PC Board Design and Layout," L-3 Avionics Systems,www.jlab.org/accel/eecad/pdf/050rfdesign.pdf.

R. Lacoste, "Easy Reflow: Build an SMT Reflow Oven Controller," *Circuit Cellar* 168, 2004.

———, "Microstrip Techniques," *Circuit Cellar* 223, 2009.

———, "Time Domain Reflectometry: Detect and Measure Impedance Mismatches," *Circuit Cellar* 225, 2009.

SOURCES

Proteus PCB Design software
Labcenter Electronics | www.labcenter.com

Spartan-6 FPGA
Xilinx, Inc. | www.xilinx.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:

—

Circuit Board Plotting
by Curt Carpenter
***Circuit Cellar* 202, 2007**
Circuit board plotting 101. Curt describes how to draw PCB traces onto copper-clad with a Hewlett-Packard 7440A ColorPro pen plotter and a Sharpie Ultra Fine Point permanent marker. Topics: Circuit Board, PCB, Plotter, HPGL, TurboCAD

—

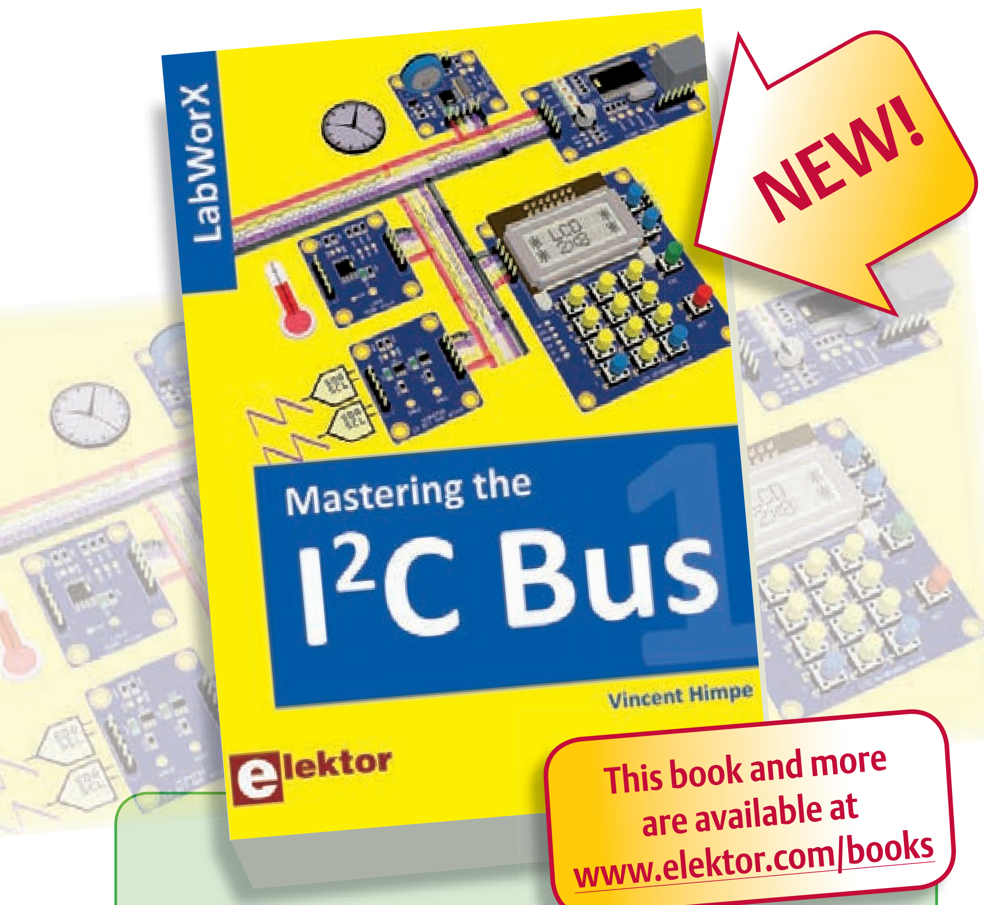
SMT Manufacturing
Take a Board from Prototype to Production
by Zack Gainsforth
***Circuit Cellar* 208, 2007**
Zack walks you through the full development cycle of a USB-to-serial adapter for a programmable knob. The design is intended for car computer systems where it can be programmed to support multiple functions. Now you too can take a similar design from prototype to production. Topics: SMT, Design Capture, Haptic Knob, USB, EDA Software

—

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

Elektor Shop

The world of electronics at your fingertips!



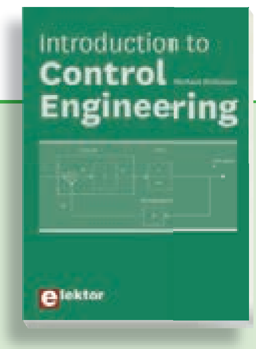
Books

LabWorX: Straight from the Lab to your Brain

Mastering the I²C Bus

LabWorX is a new collection of books, each handling a particular topic in the field of electronics.Each volume condenses all the information, applications and notes collected during hands on work with the covered technology, all into one book. Besides the basics of the technology, in depth examples and applications are given as well as troubleshooting tips on how to proceed if the initial implementation fails. Mastering the I²C Bus is the first book in the LabWorX collection. It takes you on an exploratory journey of the I²C Bus and its applications. Besides the Bus protocol plenty of attention is given to the practical applications and designing a solid system. The most common I²C compatible chip classes are covered in detail. Two experimentation boards are available that allow for rapid prototype development. These are completed by a USB to I²C probe and a software framework to control I²C devices from your computer. All samples programs can be downloaded from the www.elektor.com/labworx support page.

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

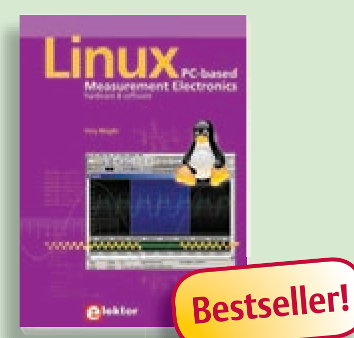


Solutions for control system applications

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



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

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

CD/DVD-ROMs

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

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

Wireless OBD-II

The cheapest way to diagnose faults on a modern car is to connect its OBD-II interface to a (notebook) PC running suitable diagnostics software. However, a wired connection is not always the most suitable, and self-contained OBD testers are a rather expensive and less flexible alternative to using a PC. An interesting option is a wireless OBD interface with a radio interface to a PC: this home-brew solution allows the choice of using either Bluetooth or ZigBee.

OBD-II-Zigbee or Bluetooth interface kit with all parts and enclosure

Art.# 100872-71 • $179.710 (Zigbee)

Art.# 100872-72 • $179.10 (Bluetooth)

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

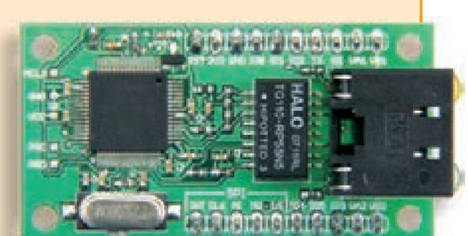
Elektor US
PO Box 180
Vernon, CT 06066
USA
Phone: 860-875-2199
Fax: 860-871-0411
E-mail: order@elektor.com

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

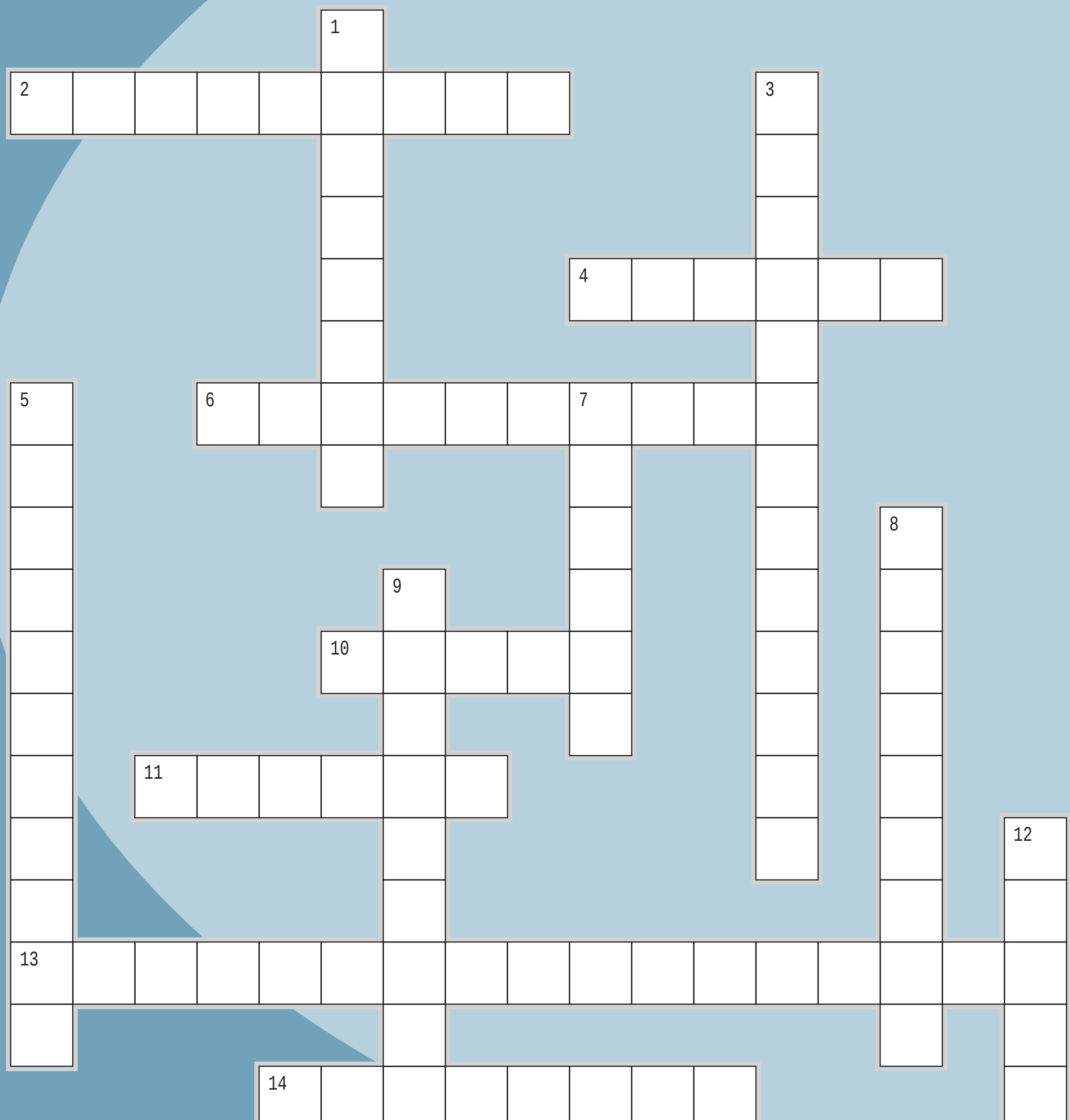
An Internet connection would be a valuable addition to many projects, but often designers are put off by the complexities involved. The 'NetWorker', which consists of a small printed circuit board, a free software library and a ready-to-use microcontroller-based web server, solves these problems and 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

Kits & Modules

CROSSWORD



Across

2. The place to accomplish things
4. Pa
6. Patented by Julius Edgar Lilienfeld in 1925
10. Slang for a massive quantity; think: Carl
11. Not public (e.g., software)
13. 555 Timer [two words]
14. Needed for C, C++, Java, etc.

Down

1. "P" in PIN
3. Denso-Wave code [two words]
5. A derivative of BASIC [two words]
7. Hertz = one cycle per?
8. Ge; semiconductor
9. Authored MINIX
12. A quick fix

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.

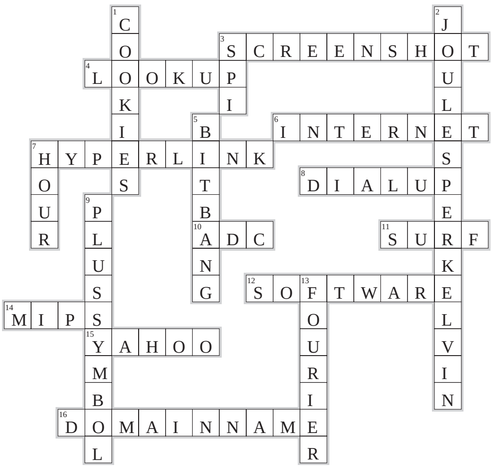
CROSSWORD ANSWERS from Issue 252

Accross

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

Down

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



INDEX OF ADVERTISERS

PREVIEW of September Issue 254

Theme: Data Acquisition

eDASH: A Virtual Dashboard Using OBD-II

Sweep Frequency Generator Design: DDS-Based Test Equipment from Sweat Equity

Li-ion Battery Analyzer

Embedded System Communication: A Control Platform for Ethernet-Enabled Systems

PID Temperature Control: The Ultimate MCU-Based Coffee Cup Warmer

---

EMBEDDED UNVEILED **Linear Positioners**

THE CONSUMMATE ENGINEER **Feedback Data Acquisition**

LESSONS FROM THE TRENCHES **Design Development (Part 3):** Software Design

FROM THE BENCH **Linear Sensor Array Implementation**

PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

Uninterruptable Powered Schizophrenia

Based on my experience, I've come to believe that certain things in our computerized existence are absolute: personal computers seem to have only 40% of their original throughput after three years of use; anything "better" from Microsoft always requires at least two times more computing power to get the same performance as the last version; and, presuming the proper VA selection, uninterruptable power supplies (UPSes) always work—or not. Welcome to my latest frustration.

This is an editorial, not a design article, but let me refresh the subject a bit. First of all, there are three basic UPS types: stand-by offline, stand-by line-interactive, and online. Line-interactive and online UPSes are the high power, expensive commercial stuff you find in server farms and commercial data centers. Besides significantly greater capacity, their higher cost is primarily due to the increased hardware complexity that it takes to make their back-up source and switching virtually transparent.

Unless we design our own back-up systems, the rest of us in the cheap seats have to rely on the less sophisticated and less expensive stand-by offline UPS. This black box (ever seen a UPS that wasn't black?) is simply a DC-AC converter with a fast switch that senses AC power loss and switches the battery-powered AC supply in its place. The good news is that the concept is relatively straightforward. The bad news is that, depending upon the cost and brand, the AC backup can be motor-unfriendly buzzing square waves, step-approximated sine wave, PWM, or true sine wave (very rare), and take 3 to 20 mS to switch. So far, so good.

At last count I have at least a dozen UPSes backing up power to various computers, the home control system, modem and Ethernet switches, DVR, webcams, etc. I've been buying UPSes (and replacing batteries) for a dozen years without really thinking about maximum power issues. For example, I have one 950VA UPS dedicated just to the 12-W HCS power. My backup logic isn't about using one 1000W UPS to keep my entire system powered for five minutes while everything does a coordinated equipment shutdown. It's about powering a dozen 5- to 20-W devices and keeping them on for a couple of hours—long enough for me to be notified and remotely review the situation. As a practical matter, the UPS up times are primarily the quiescent power consumption of the UPS itself (stand-by offline UPSes cannot use external batteries without significant considerations). My "eccentric" back-up strategy has worked fine for years.

In the real world, where most UPS applications are running at 70 to 80% capacity, users have to consider the volt-Ampere (VA) consumption of the load and its power factor (PF). UPS manufacturers love to sell us 500VA UPSes for cheap dollars, but what is that in real Watts? A 120-VAC, 1-amp incandescent bulb (power factor 1) is 120 nVA. An inductive 1-amp load at the same voltage with a power factor of .6 is 200 VA! This confusion between UPS Watts and VA ratings among consumers is one of the reasons that you now see UPS packaging listing back-up time for both terms. But, that still may not help!

My UPS frustration started when I added a UPS to the i7-2600K desktop computer and three LED monitors I told you about a couple editorials ago. Rather than guess about consumption, I plugged the system powerstrip into a Kill-A-Watt power monitor and measured it. The device said the total load was 240 W with a power factor of .9 (all switching power supplies). The VA rating would therefore be 267 VA (i.e., 240/.9). Having both 350 VA (250 W) and 500 VA (350 W) units on the shelf (with new batteries) I plugged the system into the 500VA unit. I pulled the plug and, bang, the backup failed! Say what?

I went down to the Circuit Cellar and pulled out two other brands of 450- and 500-VA UPSes and both failed as well. OK, maybe the power factor is really .6 or .7 and the load is 350 to 400 VA and it's too close. To test my "overhead" hypothesis I borrowed one of the 950-VA UPSes from the HCS and powered up the computer—I pulled the plug and, bang, no power! The 950-VA UPS said it was good for 650 W!

OK, I still didn't understand, but I followed classic Circuit Cellar tradition—if at first you fail, try complete overkill next! I jumped in the car, went down to Office Depot, and bought an APC 1,500-VA UPS rated for 865 W (power factor .6). This time when I pulled the plug, the 240-W load continued running. The UPS LCD confirmed the load was still 240 W and 0.9 power factor.

Without spending a lot of oscilloscope time diagnosing a solved problem (albeit done with zero finesse), I'm at a complete loss understanding why it took a 1,500-VA UPS for a 240-W load. I can only surmise that the 750-W desktop switching power supply or graphics board input currents must be spiking the instant the standby AC power supply switches and it must look like a 700- to 800-W load. Certainly, if you understand more about all this, please tell me.

steve.ciarcia@circuitcellar.com


pico Technology
PicoScope PC Oscilloscope
www.picotech.com