PROJECT: Custom I/O Transmission with USB
LOCATION: United States
PAGE: 28

SECURITY: Electronic Signatures 101
LOCATION: United States
PAGE: 44

INSIGHT: Wireless Data Delivery & Collection
LOCATION: United States
PAGE: 68

CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

JULY 2012
ISSUE 264

INTERNET & CONNECTIVITY

Build an Internet-Enabled Monitoring System

Project Planning & Scheduling Tips

DIY Digital Thermometer

Switch Debouncing

Inside the "EtherCAT Orchestra"

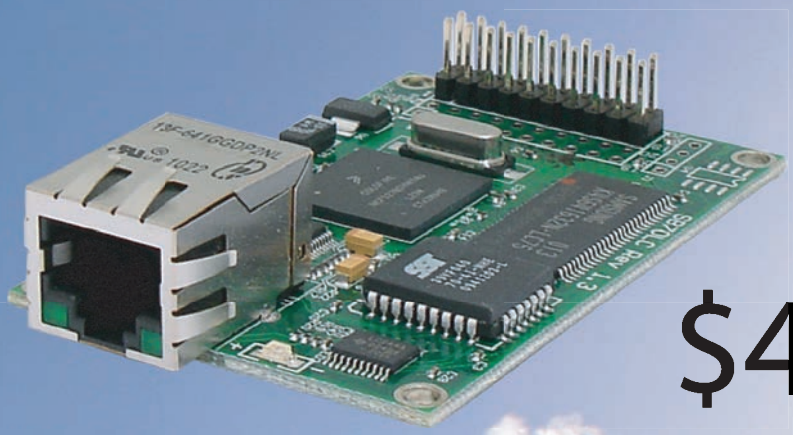
PLUS

Green Computing

Exciting Research that Could Lead to More Energy-Efficient Embedded Systems

07>
7 25274 75349 9
$7.50 U.S. ($8.50 Canada)
www.circuitcellar.com

The NetBurner SB70LC



$49.00
Qty. 100
SB70LC Board

The complete hardware and software solution

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

$99.00
SB70LC Development Kit

The SB70LC Development Kit

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

NetBurner
Networking in One Day!

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

freescale™
Alliance Member

Messe München
International

the world needs innovative electronics. they are on display here.

electronica 2012

inside tomorrow

25th International Trade Fair
for Electronic Components,
Systems and Applications
Messe München
November 13–16, 2012
www.electronica.de

TASK MANAGER

Plan, Construct, and Secure

I am proud to introduce this project-centric issue. We're presenting you with innovative ideas for embedded design projects, providing design tips with real-world examples, and tying everything together with essential information on design planning and security. It's up to you to plan a design, construct it, secure it, and then share your experiences with *Circuit Cellar*.

Interest in building and home control systems (HCSes) never wanes. In fact, articles about such projects have appeared in this magazine since 1988. On page 18, John Breitenbach details how he built an Internet-enabled, cloud-based attic monitoring system. Turn to page 36 for another HCS article. Tommy Tyler explains how to build a handy MCU-based digital thermometer. You can construct a similar system for your home, or you can apply what you learn to a variety of other temperature-sensing applications. Are you currently working on a home automation design or industrial control system? Check out Richard Wotiz's "EtherCAT Orchestra" (p. 52). He describes an innovative industrial control network built around seven embedded controllers.

The rest of the articles in the issue cover essential electrical engineering concepts and design techniques. Engineers of every skill level will find the information immediately applicable to the projects on their workbenches.

Tom Struzik's article on USB is a good introduction to the technology, and it details how to effectively customize an I/O and data transfer solution (p. 28). On page 44, Patrick Schaumont introduces the topic of electronic signatures and then details how to use them to sign firmware updates. George Novacek provides a project development road map for professionals and novices alike (p. 58). Flip to page 62 for George Martin's insight on switch debouncing and interfacing to a simple device. On page 68, Jeff Bachiochi tackles the concepts of wireless data delivery and time stamping.

I encourage you to read the interview with Boston University professor Ayse Kivilcim Coskun on page 26. Her research on 3-D stacked systems has gained notoriety in academia, and it could change the way electrical engineers and chip manufacturers think about energy efficiency for years to come. If you're an engineer fascinated by "green computing," you'll find Coskun's work particularly intriguing.

Special note: The Circuit Cellar *staff dedicates this issue to Richard Alan Wotiz who passed away on May 30, 2012. We appreciate having had the opportunity to publish articles about his inventive projects and innovative engineering ideas and solutions. We extend our condolences to his family and friends.*

cj@circuitcellar.com

CIRCUIT CELLAR®
THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

EDITORIAL CALENDAR

Issue	Theme
258 January	Embedded Applications
259 February	Wireless Communications
260 March	Robotics
261 April	Embedded Programming
262 May	Measurement & Sensors
263 June	Communications
264 July	Internet & Connectivity
265 August	Embedded Development
266 September	Data Acquisition
267 October	Signal Processing
268 November	Analog Techniques
269 December	Programmable Logic

Analog Techniques: Projects and components dealing with analog signal acquisition and generation (e.g., EMI/RF reduction, high-speed signal integrity, signal conditioning, A/D and D/A converters, and analog programmable logic)

Communications: Projects that deal with computer networking, human-to-human interaction, human-to-computer interaction, and electronic information sharing (e.g., speech recognition, data transmission, Ethernet, USB, I^2C, and SPI)

Data Acquisition: Projects, technologies, and algorithms for real-world data gathering and monitoring (e.g., peripheral interfaces, sensors, sensor networks, signal conditioning, A/D and D/A converters, data analysis, and post-processing)

Embedded Applications: Projects that feature embedded controllers and MCU-based system design (e.g., automotive applications, test equipment, simulators, consumer electronics, real-time control, and low-power techniques)

Embedded Development: Tools and techniques used to develop new hardware or software (e.g., prototyping and simulation, emulators, development tools, programming languages, HDL, RTOSes, debugging tools, and useful tips and tricks)

Embedded Programming: The software used in embedded applications (e.g., programming languages, RTOSes, file systems, protocols, embedded Linux, and algorithms)

Internet & Connectivity: Applications that deal with connectivity and Internet-enabled systems (e.g., networking chips, protocol stacks, device servers, and physical layer interfaces)

Measurement & Sensors: Projects and technologies that deal with sensors, interfaces, and actuators (e.g., one-wire sensors, MEMS sensors, and sensor interface techniques)

Programmable Logic: Projects that utilize FPGAs, PLDs, and other programmable logic chips (e.g., dynamic reconfiguration, memory, and HDLs)

Robotics: Projects about robot systems, devices capable of repeating motion sequences, and MCU-based motor control designs (e.g., mobile robots, motor drives, proximity sensing, power control, navigation, and accelerometers)

Signal Processing: Projects and technology related to the real-time processing of signals (e.g., DSP chips, signal conditioning, ADCs/DACs, filters, and comparisons of RISC, DSP, VLIW, etc.)

Wireless Communications: Technology and methods for going wireless (e.g., radio modems, Wi-Fi/IEEE 802.11x, Bluetooth, ZigBee/IEEE 802.15.4, cellular, infrared/IrDA, and MCU-based wireless security applications)

UPCOMING IN CIRCUIT CELLAR

FEATURES

eCompass, by Mark Pedley

Qseven, by Antoine Deschamps

Examining an I/O Port, by Shlomo Engelberg

COLUMNS

Charging with PV Cells, by Jeff Bachiochi

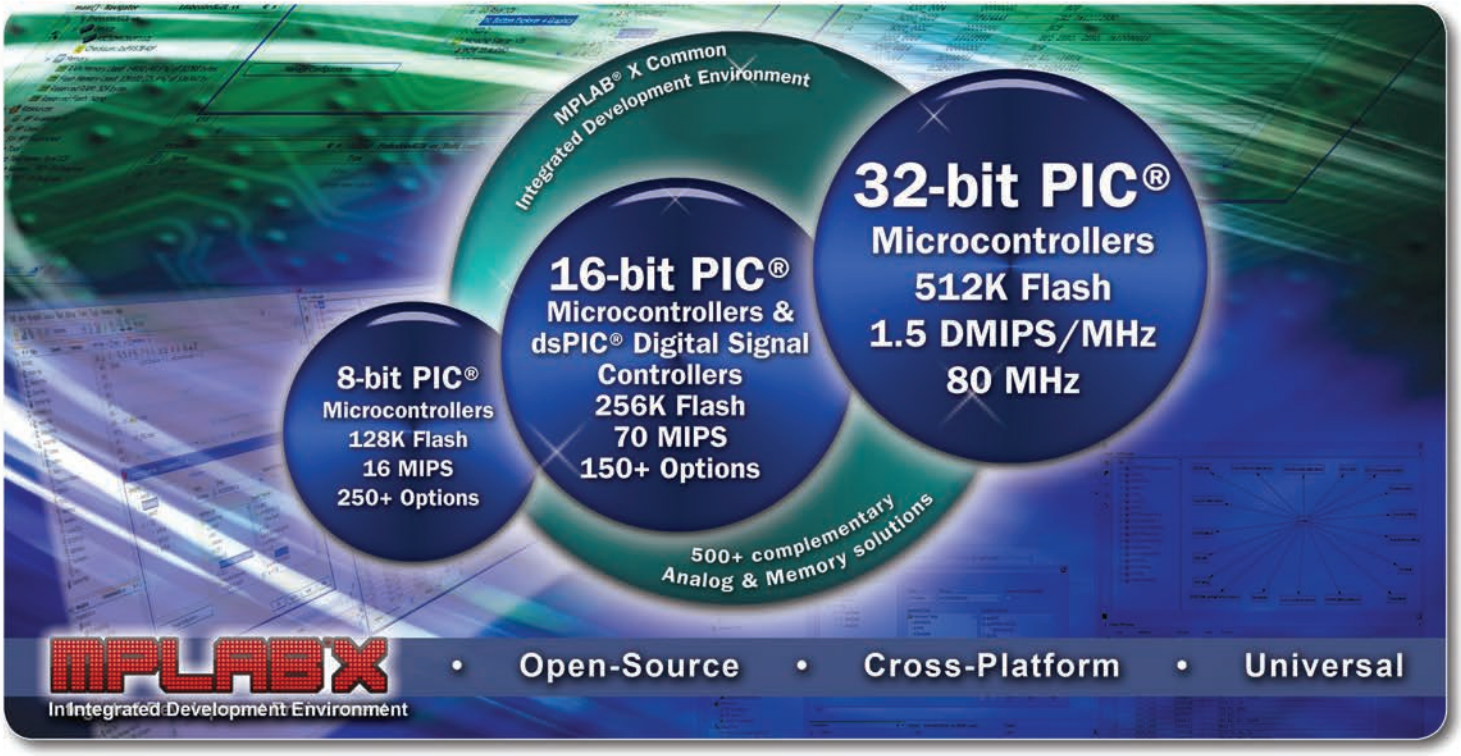
MOSFET Channel Resistance, by Ed Nisley

System-Level RF Design, by Robert Lacoste

Project Development (Part 2), by George Novacek

Concurrency in Embedded Systems (Part 2), by Bob Japenga

One platform for 8-, 16- and 32-bit development - with Microchip's MPLAB® X IDE



MPLAB® X IDE is the free, integrated toolset for all of Microchip's 900+ 8-, 16- and 32-bit PIC® Microcontrollers, dsPIC® Digital Signal Controllers, and memory devices. Based on the open-source NetBeans platform, MPLAB X runs on Windows® OS,MAC® OS and Linux, supports many third-party tools, and is compatible with many NetBeans plug-ins.

MPLAB XC compilers help increase code speed of any PIC® Microcontroller or dsPIC® digital signal controller by 30%, whilst also cutting code size by 35%. These new compilers give designers the choice of Free, Standard or Pro code optimisation levels for 8-bit, 16- or 32-bit development, or a single C compiler suite to support all Microchip Microcontrollers and digital signal controllers.

Microchip's tool chain of compatible compilers and debugger/programmers operate seamlessly within the universal, cross platform and open-source MPLAB® X integrated development environment, reducing both learning curves and tool investments.

START DEVELOPING TODAY

Download a free copy of MPLAB X and choose from a choice of new C compilers:

- MPLAB XC8 for 8-bit MCUs
- MPLAB XC16 for 16-bit MCUs and DSCs
- MPLAB XC32 for 32-bit MCUs
- MPLAB XC Suite for all 900+ PIC MCUs and dsPIC DSCs.

Evaluate MPLAB X today! www.microchip.com/mplabx

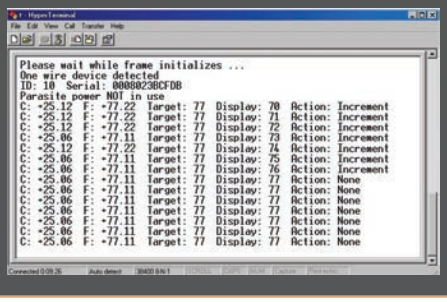
MICROCHIP

Microcontrollers • Digital Signal Controllers • Analog • Memory • Wireless

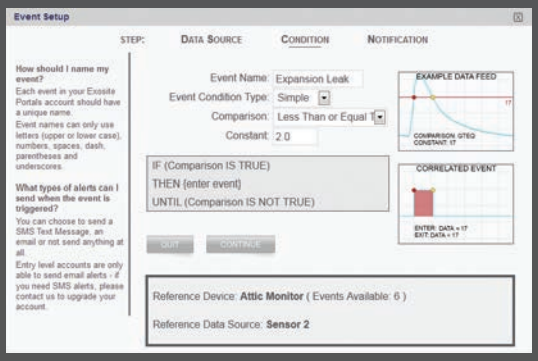
INSIDE ISSUE 264

July 2012 • Internet & Connectivity

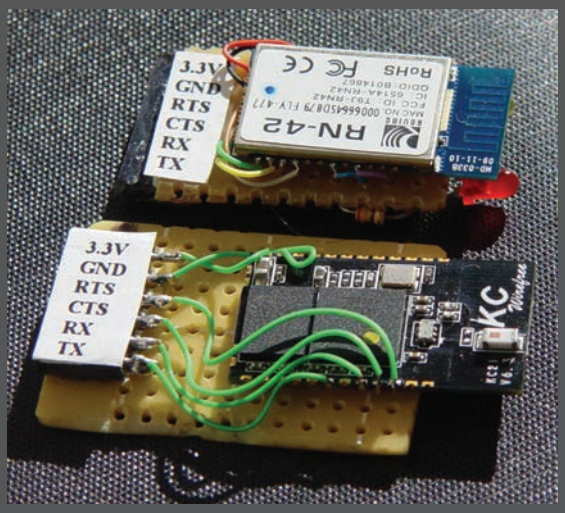
DIY Digital Thermometer p. 36



MCU-Based Remote Sensing p. 18



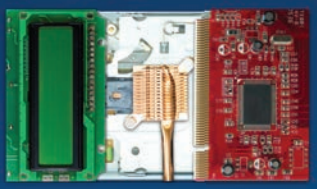
Simplified Data Delivery p. 68

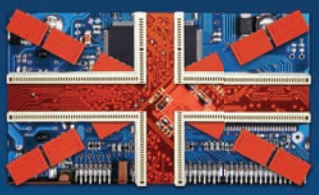


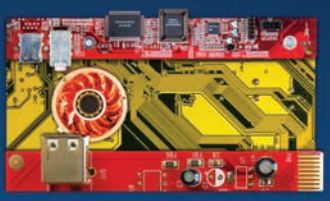
mouser.com
Distributing semiconductors and electronic components for design engineers.

Authorized Distributor

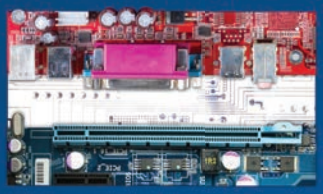

Texas, California, New Jersey **USA**


Jalisco **MEXICO**

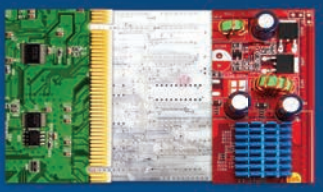

Buckinghamshire **UNITED KINGDOM**

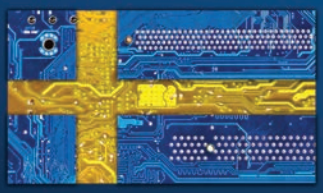

Barcelona **SPAIN**

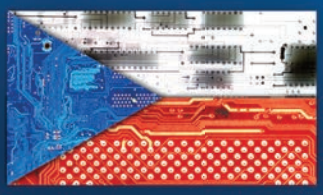

Brive-La-Gaillarde **FRANCE**

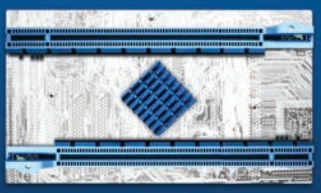

Eindhoven **THE NETHERLANDS**

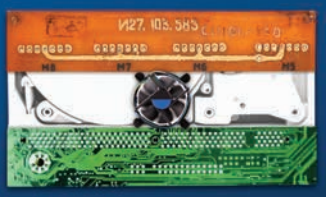

Munich **GERMANY**


Assago-MI **ITALY**


Upplands-Väsby **SWEDEN**

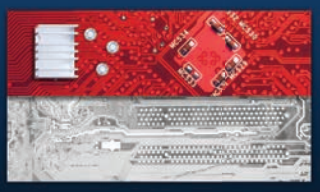

Brno **CZECH REPUBLIC**


Raanana **ISRAEL**


Bengaluru **INDIA**


Hong Kong, Shanghai **CHINA**


Bangkok **THAILAND**


SINGAPORE


Taipei **TAIWAN**

With local support all over the world, we're fluent in technology.

Mouser delivers the components you need, on-time. And with local Technical Support and Customer Service Experts in 19 locations around the world, you'll find the newest components to launch your new design seamlessly.

mouser.com | The Newest Products for Your Newest Designs®

MOUSER ELECTRONICS®
a tti company

The team

The network

audioXPRESS

elektor

CIRCUIT CELLAR
THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

techthefuture
Tech the Future explores the solutions for a sustainable future provided by technology, creativity and science.

VOICE COIL

Our international teams

United Kingdom
Wisse Hettinga
+31 (0)46 4389428
w.hettinga@elektor.com

USA
Hugo Van haecke
+1 860 875 2199
h.vanhaecke@elektor.com

Germany
Ferdinand te Walvaart
+49 (0)241 88 909-0
f.tewalvaart@elektor.de

France
Denis Meyer
+31 (0)46 4389435
d.meyer@elektor.fr

Netherlands
Harry Baggen
+31 (0)46 4389429
h.baggen@elektor.nl

Spain
Eduardo Corral
+34 91 101 93 85
e.corral@elektor.es

Italy
Maurizio del Corso
+39 2.66504755
m.delcorso@inware.it

Sweden
Wisse Hettinga
+31 (0)46 4389428
w.hettinga@elektor.com

Brazil
João Martins
+351214131600
joao.martins@editorialbolina.com

Portugal
João Martins
+351214131600
joao.martins@editorialbolina.com

India
Sunil D. Malekar
+91 9833168815
ts@elektor.in

Russia
Nataliya Melnikova
8 10 7 (965) 395 33 36
nataliya-m-larionova@yandex.ru

Turkey
Zeynep Köksal
+90 532 277 48 26
zkoksal@beti.com.tr

South Africa
Johan Dijk
+27 78 2330 694 / +31 6 109 31 926
J.Dijk @elektor.com

China
Cees Baay
+86 (0)21 6445 2811
CeesBaay@gmail.com

Issue 264 July 2012 ISSN 1528-0608

CIRCUIT CELLAR® (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Vernon, CT 06066. Periodical rates paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate USA and possessions $50, Canada $65, Foreign/ROW $75. All subscription orders payable in U.S. funds only via Visa, MasterCard, international postal money order, or check drawn on U.S. bank.

Cover photography by Chris Rakoczy—www.rakoczyphoto.com

Subscriptions
Circuit Cellar, P.O. Box 462256, Escondido, CA 92046
E-mail: circuitcellar@pcspublink.com
Phone: 800.269.6301, Internet: www.circuitcellar.com
Address Changes/Problems: circuitcellar@pcspublink.com

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

US Advertising
Strategic Media Marketing, Inc.
2 Main Street, Gloucester, MA 01930 USA
Phone: 978.281.7708, Fax: 978.281.7706, E-mail: peter@smmarketing.us
Internet: www.circuitcellar.com
Advertising rates and terms available on request.

New Products: New Products, Circuit Cellar, 4 Park Street, Vernon, CT 06066, E-mail: newproducts@circuitcellar.com

MEMBERSHIP COUNTER

We now have 274283 members in 83 countries.

Not a member yet?

Sign up at www.circuitcellar.com

SUPPORTING COMPANIES

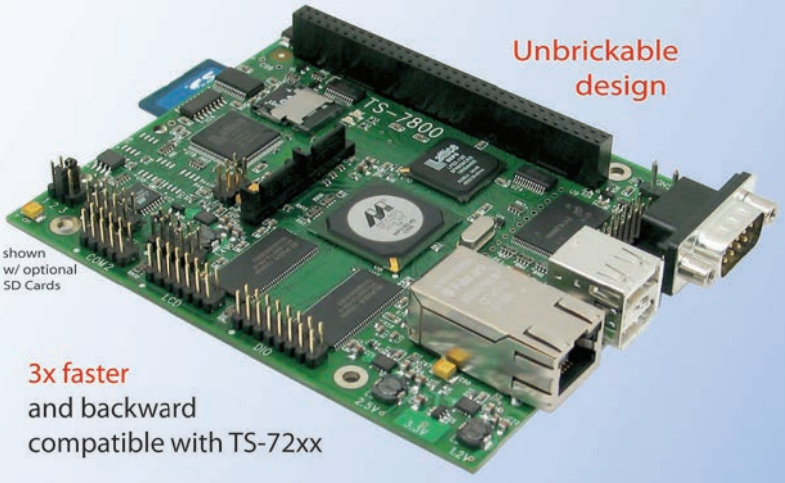
Not a supporting company yet?

Contact Peter Wostrel (peter@smmarketing.us, Phone 978.281.7708, Fax 978.281.7706) to reserve your own space for the next issue of our member's magazine.

Head Office
Circuit Cellar, Inc.
4 Park Street, Vernon, CT 06066, Phone: 860.875.2199

Embedded Systems

High-End Performance with Embedded Ruggedness



Unbrickable design

3x faster and backward compatible with TS-72xx

TS-7800
500MHz 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

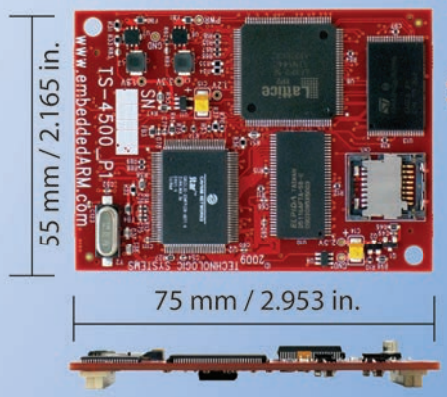
$229 qty 100

$269 qty 1

TS-SOCKET Macrocontrollers
Jump Start Your Embedded System Design

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

- TS-4200: Atmel ARM9 with super low power
- TS-4300: 600MHz ARM9 and 25K LUT FPGA
- TS-4500: Cavium ARM9 at very low cost
- TS-4700: 800MHz Marvell ARM with video
- TS-4800: 800MHz Freescale iMX515 with video
- Several COTS baseboards for evaluation & development



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

series starts at

$92 qty 100

$139 qty 1

- Over 25 years in business
- Open Source Vision
- Never discontinued a product
- Engineers on Tech Support
- Custom configurations and designs w/ excellent pricing and turn-around time
- Most products stocked and available for next day shipping

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

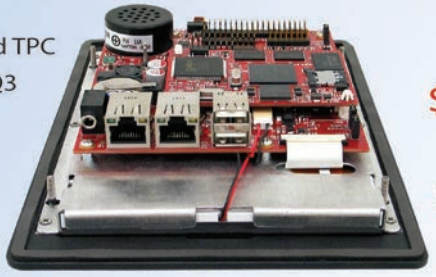
New Products

Touch Panel Computers
800MHz with Video Acceleration

- Resistive touchscreen, LED backlit display
- Gasketed construction
- Tough powder coated finish
- Fanless operation from -20°C to +70°C
- 800MHz ARM CPU
- 256MB RAM, 256MB SLC XNAND Drive
- MicroSD slot
- 5K LUT programmable FPGA
- Dual Ethernet, USB ports
- CAN, RS-232 ports, RS-485
- Mono speaker on PCB, stereo audio jack
- SPI, DIO

Fully enclosed TPC
available Q3

NEW!



series
starts at
$415
qty 100

$479
qty 1

picture of TS-8820-BOX

NEW!

series
starts at
$199
qty 100

$229
qty 1

Technologic Systems now offers three powerful computers targeting industrial process control. Implement an intelligent automation system at low cost with a minimal number of components.

Industrial Controllers
Powerful, Rugged, Affordable

- 250MHz (ARM9) or 800MHz (ARM9 or Cortex-A8) CPU
- Fast startup (under 3 seconds)
- Fanless operation from -20°C to +70°C
- User-programmable opencore FPGA
- Program in Ladder Logic or C
- Debian Linux
- Modbus support
- PoE capable 10/100 Ethernet, USB 2.0 Host Ports
- Industrial screw-down connectors
- Opto-Isolated DIO, Digital Counters, Quadrature
- Up to 46 DIO with PWM
- Opto-Isolation available for RS-232, RS-485 and CAN
- DIN mount option

Technologic Systems

We use our stuff.

Visit our TS-7800 powered website at

www.embeddedARM.com

FAST, STABLE DEBUGGING PROBE

I-jet is a high-performing in-circuit probe designed for fast and stable debugging. The probe is seamlessly integrated into the IAR Embedded Workbench for the ARMC/C++ compiler and debugger tool suite.

I-jet's fast debugging platform features download speeds of up to 1 MBps, JTAG and serial wire debug (SWD) clocking at up to 32 MHz, and serial wire output (SWO) frequencies of up to 60 MHz. The probe can deliver power to the target board and accurately measure target power consumption. It is fully plug-and-play, and offers user-friendly features such as automatic core recognition and direct download into the flash memory of most popular microcontrollers. No power supply is needed since I-jet is entirely powered by USB.

The debugging probe supports microcontrollers based on ARM7, ARM9, ARM11, ARM Cortex-M, ARM Cortex-R4, and ARM Cortex-A5/A8/A9 cores. The serial wire viewer (SWV) is supported using the UART and Manchester encoding modes. Embedded trace buffer (ETB) and JTAG-adaptive clocking are supported, and all JTAG signals can be monitored.

The I-jet costs **$299**.

IAR Systems
www.iar.com

CONTROLLER-ONLY BUCK LED DRIVER

The **SSL2109** is a high-efficiency buck controller for high-power, non-dimmable LED lighting applications using nonisolated topologies. Designed for use with an external power switch, the SSL2109 LED driver IC provides a single design platform for 100-V, 120-V, and 230-V mains input voltages and power ranges up to 25 W.

Based on GreenChip technology, the controller enables manufacturers of LED retrofit lamps and driver modules to optimize external MOSFET selection based on cost, performance, and heat dissipation across different power levels.

The SSL2109 is a controller-only version of the SSL2108*x* family of LED driver ICs with many similar features, including: a high integration level with only 15 components required on the PCB, a small 18 mm × 22 mm PCB area, high efficiency up to 95%, a low electronic bill of materials for LED driver applications, tight LED current regulation, the option for direct PWM dimming, high reliability with IC lifetime matching or surpassing LED lifetime, and a complete set of built-in protections, including LED temperature protection via an NTC temperature sensor.

Contact NXP for pricing.

NXP Semiconductors
www.nxp.com

NEW PRODUCT NEWS

COMPLETE ENGINEERING SOLUTIONS

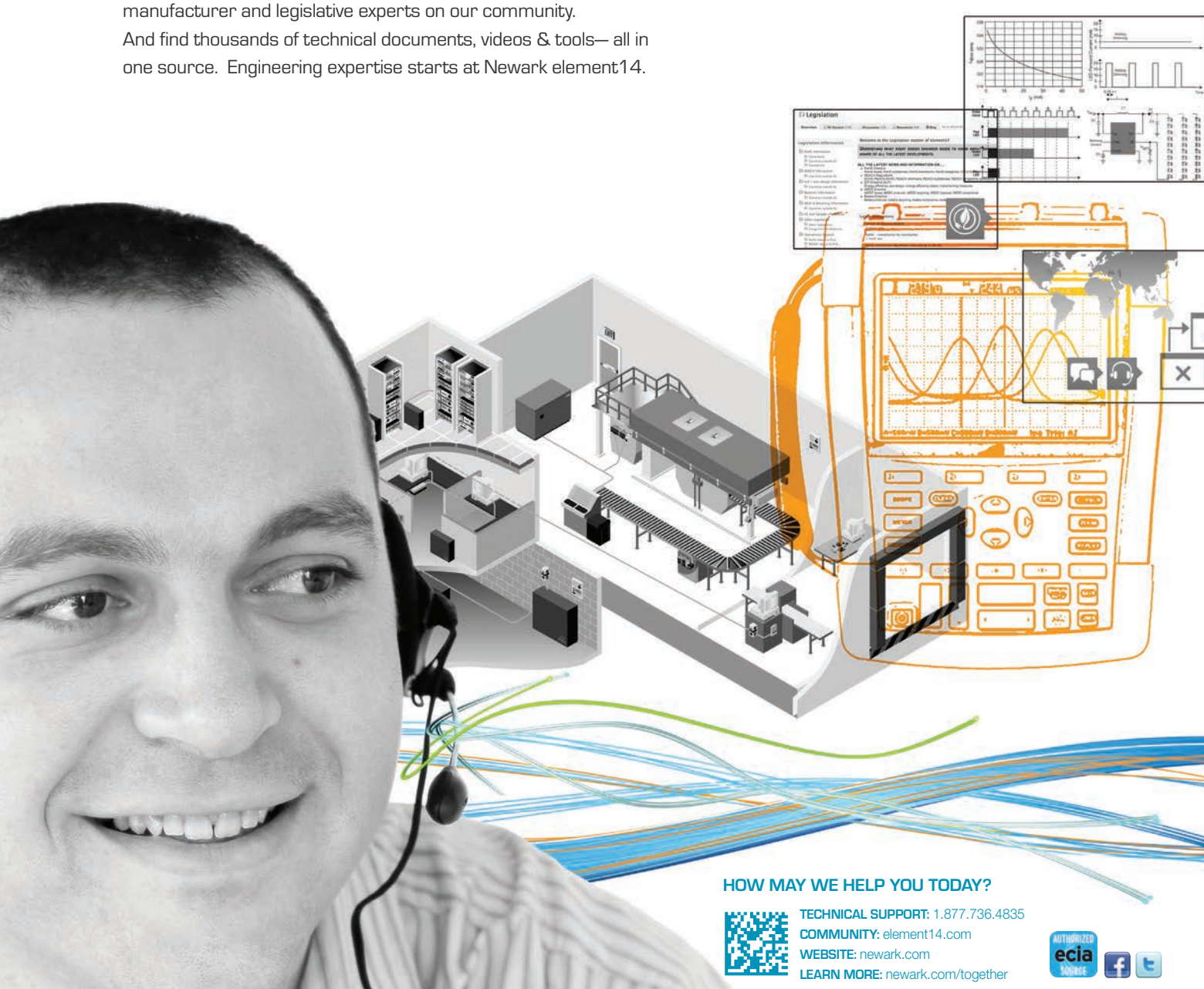
Start here.

Get direct, one-on-one technical support from real engineers with no go-between and no waiting. Access industry, manufacturer and legislative experts on our community. And find thousands of technical documents, videos & tools— all in one source. Engineering expertise starts at Newark element14.

9 out of 10 customers recommend Newark element14 Technical Support

– Customer feedback studies

Newark™ element14

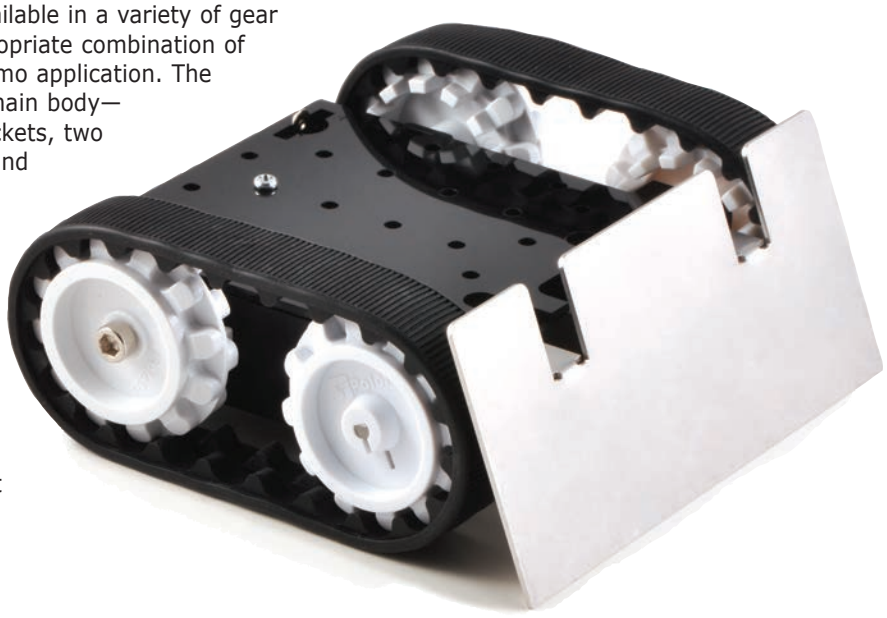


HOW MAY WE HELP YOU TODAY?

TECHNICAL SUPPORT: 1.877.736.4835
COMMUNITY: element14.com
WEBSITE: newark.com
LEARN MORE: newark.com/together

SMALL PLATFORM FOR MINI SUMO COMPETITIONS

The **Zumo** chassis is a small, low-profile, tracked-robot platform with a main body made of ABS plastic. The platform features sockets for two micro metal gearmotors and a compartment for four AA batteries (motors and batteries sold separately). The micro metal gearmotors are available in a variety of gear ratios, making it possible to select the appropriate combination of torque and speed required for a specific Zumo application. The chassis ships as a kit and—along with the main body—includes two silicone tracks, two drive sprockets, two idler sprockets, an acrylic mounting plate, and mounting hardware.



With dimensions of 98 mm × 86 mm, the Zumo qualifies for Mini Sumo competitions. For these applications, Pololu separately offers a basic stainless steel sumo blade that can be mounted to the front of the Zumo chassis to push around other objects (i.e., other Mini Sumo robots). The design file for the sumo blade is publicly available and can be used as a starting point for custom laser-cut blades.

The Zumo chassis costs **$19.95**.

Pololu Corp.
www.pololu.com

STARTER KITS FOR MEMORY-HUNGRY & ENERGY-EFFICIENT APPLICATIONS

The **EFM32-GGSTK3700** and **EFM32LG-STK3600** starter kits are for designers building complex battery-powered products such as portable health and fitness devices and smart accessories. Based on the EFM32 Leopard Gecko and Giant Gecko microcontrollers, the EFM32 starter kits incorporate 1-MB built-in flash memory, onboard debug and current probe functionality, and all of the features required to demonstrate the Gecko microcontrollers' 400-nA sleep modes. The two starter kits support memory-rich devices within the 240-strong ARM Cortex-M EFM32 Gecko family.

The EFM32-GGSTK3700 is based on the EFM32GG990F1024, an energy-friendly microcontroller with 1-MB on-chip flash and 128-KB RAM memory. The EFM32LG-STK3600 is fitted with a 256-KB EFM32LG990F256 device. The memory range options enable designers to implement complex embedded designs while maintaining high-energy efficiency. Both the Leopard Gecko and the Giant Gecko microcontroller can directly control a TFT display. They feature USB drivers that support the host, device, and on-the-go (OTG) protocols.

The EFM32 Leopard Gecko and the Giant Gecko starter kits are equipped with light, metal, and touch sensors designers can use to investigate the Gecko's LESENSE low-energy sensor interface. This enables passive sensing of 16 sensors without host CPU intervention. The hardware feature set is completed by a USB plug, 32-MB onboard NAND flash, an LCD, and a variety of LEDs and push buttons.

All Energy Micro development kits are supported by Simplicity Studio, a free EFM32 development software suite that includes all the latest documentation, examples, software, and drivers for Gecko designers.

The EFM32GG-STK3700 and EFM32LG-STK3600 starter kits cost **$79** each.

Energy Micro
www.energymicro.com

Take your chance to win

EAGLE V6

THE NEXT GENERATION

EAGLE Design Challenge

1st of May – 31st of August 2012

Do you have a great idea for a board?

You have a great idea for a board and want to win a DELL Alienware M17xr3 computer, an EAGLE Pro license or a MICROCHIP - DV164037 & DM163022-1? Participate in the EAGLE design competition, powered by Microchip and hosted by element14!

To get a chance to win include an MCU or DSC in your design made with EAGLE version 6, describe your project on one page, make a screenshot of your layout and post it on **www.element14.com/eagle-competition**

Visit **www.element14.com/eagle-competition** for terms and conditions

In Association with

Newark element14 MICROCHIP

www.newark.com www.element14.com/eagle www.microchip.com www.cadsoftusa.com

SENSOR FEATURES ACCURATE MEASUREMENT & LONG-TERM STABILITY

The **SPD500** is a low-cost positive differential pressure sensor. The fully calibrated, temperature-compensated sensor is designed to measure differential pressure in the 0-to-500-Pa range. It can detect even minute pressure differences (less than 10 Pa). The low-cost sensor features a zero-point accuracy of 0.2 Pa.

Integration of the sensor element and signal-conditioning circuitry on a tiny CMOS silicon chip enables noise-free and precise amplification and digitization of sensor signals. This enables the differential pressure sensor to achieve high measurement accuracy and long-term stability. With an accuracy of 4.5% of the measured value, zero-point stability, and I²C digital output, the SPD500 is well suited for applications in the HVAC industry. Similar to other sensors in the SPD600 series, the differential pressure sensor is available in two different versions. The SDP500 is designed to be screwed directly to a manifold with O-ring sealing. The SDP510 is designed for tube connections.

Contact Sensirion for pricing.

Sensirion AG
www.sensirion.com

ANALOG THREE-AXIS, HIGH-g MEMS ACCELEROMETER

The **ADXL377** is an analog, three-axis, high-g MEMS accelerometer that measures acceleration of high-impact events resulting from shock and vibration within the fullscale range of ±200 g with no signal saturation. This measurement range, combined with an analog output that continuously captures impact data, makes the accelerometer well suited for contact sports where the detection of concussive forces can help indicate traumatic brain injury.

With a 1,600-Hz bandwidth, the ADXL377 is also ideal for use in industrial equipment where shock levels must be closely monitored. The accelerometer features a simple design that eliminates the need for alignment and the placement of orthogonal sensors. The board space requirement is reduced by up to five times that of typical solutions requiring multiple, single-axis accelerometers.

Among other applications, the ADXL377 accelerometer is being designed into the IZOD 2012 IndyCar Series driver impact safety system. The resulting device enabled IndyCar to upgrade the sensors in their communications earpieces.

Additional features of the ADXL377 MEMS accelerometer include: 300 µA (typical) power consumption, single-supply operation (1.8 V to 3.6 V), 10,000-g shock survival, and small, thin packaging.

The ADXL377 costs **$4.79** in 1,000-unit quantities.

Analog Devices, Inc.
www.analog.com

AES-128-BASED TRANSPONDER FOR CAR KEY FOB APPLICATIONS

The **ATA5580** is a secure, ultra-low-power micromodule transponder based on an Atmel AVR microcontroller. The transponder includes the Atmel open immobilizer protocol stack with a built-in, high-performance AES-128 hardware cryptographic unit, a low-frequency (LF) immobilizer interface for power supply and bidirectional communication, and an LF antenna. The ATA5580 standalone immobilizer transponder is designed to be overmolded in simple mechanical keys that usually accompany full-featured remote keyless-entry key fobs.

Based on a land grid array (LGA) package, the ATA5580 transponder doesn't use a lead frame construction, providing high-performance electrostatic discharge (ESD) protection that is important in harsh environments such as overmolding processes. The design, along with the high modulation index, delivers a typical coupling factor in the range of 1% while running a complete AES authentication.

The ATA5580's immobilizer protocol is configurable via EEPROM, which enables designers to easily optimize system parameters such as authentication time against coupling factor/bit security.

Pricing for the ATA5580 starts at **$2** for 10,000-piece quantities.

Atmel Corp.
www.atmel.com

by Rob Dautel (USA)

The RL78 MCU

Shaping the Way the World Experiences Low-Power Design

The RL78 isn't just an MCU, but an extensive ecosystem comprising training, support, and partners. Keep reading to discover all the ways Renesas is ready to help you take your green energy idea to the next level through an extensive network of video tutorials, partner programs, and more.

Remember the 1980s? Big hair, bright, pastel clothing, and electronic music. How about microcontroller (MCU) development? Reading physical databooks and software manuals, working with cumbersome command-line tools, and using as much as 4 K of memory? If you were working on a high-end system, you may have been lucky enough to work with that new technology called DRAM.

Perhaps you're too young to remember MCU work in the 80s. It was pioneering and driven. Every byte counted, you had to know what the bits did, and if you needed a function, you wrote it. Fortunately, the 80s are gone—save for a few bands that have hung on, but that's a topic for another time.

Today's engineers want solutions, reuse, and quick turn around on designs. An MCU company cannot afford to just dump off a few books, wish the design team good luck, and stop back in a few months for production orders. An MCU vendor must provide an ecosystem for the controllers it offers and that ecosystem must be robust and agile enough to hold up to the demands of customers large and small. At Renesas, that's exactly what we've done.

The RL78 has an extensive ecosystem from training to support to partners. In the following few paragraphs, I'll touch on a few key points of the Renesas ecosystem which supports not only the RL78, but all Renesas MCUs.

Training. No one wants to read a thousand-page databook right? Okay, perhaps if you're trying to cure a case of insomnia. Renesas provides a range of options for getting started with the RL78. Our first stop is www.renesasinteractive.com, which is our eLearning site. Operating 24/7, RenesasInteractive holds hundreds of courses on technology, microcontrollers, and tools. This is a great way to get a quick overview on an MCU, brush up on some technology, or dive deeply into how a peripheral works. If you prefer the more hands-on approach, Renesas and our distributors offer local and onsite trainings constantly and these in-person trainings can be tailored to your specific needs. Need even more? Every two years, Renesas hosts its developer's conference, or DevCon. The next one is coming in October 2012, in Orange County, California, U.S. This three-and-a-half day event features more than 120 technical sessions and labs from Renesas and our partners along with expert panels, discussions, and an exhibits floor. You can learn more about DevCon 2012 at www.renesasdevcon.com.

Support. When you need it, we've got it. Renesas provides support from factory FAEs, distribution FAEs trained on Renesas products, an applications center, global support e-mails, and several online services such as our community support forums (www.renesasrulz.com), which host thousands of threads and conversations, and great blogs such as our Dr. Micro, who serves up interesting tech bits on a regular basis. There's also the Renesas Presents YouTube channel (www.youtube.com/renesaspresents), with dozens of videos on a variety of topics.

Partners. The RL78/G13 RDK used in the RL78 Green Energy Challenge is the culmination of many Renesas Alliance partners working tirelessly to support and offer solutions to ease the burden of evaluation and development and aid in time to market. Alliance partners offer products and services ranging from software and stacks to design services, consulting, and full turn-key production. For an overview of the Alliance program and to see all of the partners, check out www.renesas.com/alliance.

Representing just a fraction of the overall Renesas Alliance partners, several have taken an active role in the RL78 Green Energy Challenge. It is these partners who have helped to make the RL78 Green Energy Challenge fun and exciting with additional prizes and weekly giveaways. You can find out more about these partners at www.circuitcellar.com/contests/renesasRL78challenge/sponsors.html.

Although we've touched on a few, there are many other aspects to the Renesas ecosystem supporting the RL78 and other MCUs. I invite you to take a look and make use of these services. Our vast ecosystem is here to serve you and we truly hope you enjoy participating in the RL78 Green Energy Challenge. We're excited to see the great designs that are being dreamt up for the RL78.

In closing, let's have a bit of fun. Since I began with the 80s, I'll end with the 80s. Below is a line from my favorite 80s band. Be one of the first five people to e-mail me the band and title and I'll send you an RL78/G13 RDK.

"How can I explain, when there are few words I can choose?"

Rob Dautel (Rob.Dautel@renesas.com), Senior Manager of Ecosystems at Renesas Electronics America, has more than 24 years of experience in hardware, software, and ASIC design. He is an expert in digital audio, industrial control, and development tools. He "speaks" 22 different programming languages.

For more information about the RL78 Green Energy Challenge, go to: www.circuitcellar.com/renesasrl78challenge

FANLESS CORE i3 & CELERON WITH COM/SERIAL PORT

The **BIS-6763/6761** is built around the Intel HM65 chipset with Intel's Core i3 2367M and Celeron 857 Processor. It consumes less than 25 W and offers an integrated GPU that processes graphics up to 1,000 MHz. Configured with rich I/Os, the BIS-6761/6763 features HDMI + VGA dual-independent display, six USBs, one COM port, a 1-Gb Ethernet port, mic-in and line-out audio on both back and front plates, and a DC 12-V power input for mobile applications. For customizability, the BIS-6761/6763 also offers two mini-PCIe expansion slots onboard for Wi-Fi, TV tuners, and other cellular connection modules. The BIS-6761/6763 provides a powerful, low-maintenance solution with no moving parts for multiple applications including medical, digital signage, industrial automation, green technology, advanced system controller, POS, and kiosks.

The BIS-6761 is designed with standard 100 mm × 100 mm VESA mounting holes on the bottom of the case to mount on walls, cabinets, or monitors. Operating systems supported include Windows XP, Windows 7, Windows XP Embedded, Windows Embedded Standard 7, Linux Ubuntu, Linux Meego, and Linux Yacto.

The BIS-6761/6763's proprietary enclosure enables interchangeable I/O plates to fit a variety of low-profile mini ITX boards as well as different Intel processors (e.g., Intel Atom to Intel Core i5). From industrial automation to digital signage, the BIS-6761/6763 can be configured to route different I/Os from board pin headers (e.g., USBs, COM, etc.) to the front or rear plate to fit specific market needs.

Contact HABEY for pricing.

HABEY USA, Inc.
www.habeyusa.com

NPN

CIRCUIT CELLAR

Test Your EQ

Problem 1a—Is it possible to transmit on/off (DC) signals between two pieces of equipment in both directions simultaneously on the same wire, in much the same way telephones do for audio?

Problem 1b—But doesn't a true hybrid use transformers, or at least some tricky transformer simulation with op-amps to ensure the transmitted signal does not appear on the receive port?

Problem 2a—The conventional way to calculate the magnitude (length) of a vector is to take the square root of the sum of the squares of its components. On small processors, this can be somewhat difficult (especially the square root operation), and various approximations are used instead.

One approximation that works surprisingly well for 2-D vectors and complex numbers is to take the absolute values of the two components, compare them, then add 1/3 of the smaller to the larger.

What is the maximum error using this method?

Problem 2b—Is there a similar formula that gives even better results?

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

by John Breitenbach (USA)

Internet-Enabled Home Control

Build a Cloud-Based Attic Monitor

It's easy to create sophisticated Internet-enabled gadgets with free tools. You can use a microcontroller coupled with free software and a cloud-based data platform to remotely monitor your home with alerts automatically available on your smartphone.

Moving from the Northeast to North Carolina, my wife and I were surprised to find that most homes don't have basements. In the north, the frost line is 36"–48" below the surface. To prevent frost heave, foundations must be dug at least that deep. So, digging down an extra few feet to create a basement makes sense. Because the frost line is only 15" in the Raleigh area, builders rarely excavate the additional 8' to create basements.

The lack of basements means builders must find unique locations for a home's mechanical systems including the furnace, AC unit, and water heater. I was shocked to find that my home's water heater is located in the attic, right above one of the bedrooms (see Photo 1).

During my high school summers I worked for my uncle's plumbing business ("Breitenbach Plumbing—We're the Best, Don't Call the Rest") and saw firsthand the damage water can do to a home. Water heaters can cause some dramatic end-of-life plumbing failures, dumping 40 or more gallons of water at once followed by the steady flow of the supply line.

Having cleaned up the mess of a failed water heater in my own basement up north, I haven't had a good night's sleep since I discovered the water heater in my North Carolina attic. For peace of mind, especially when travelling, I instrumented my attic so I could be notified immediately if water started to leak. My goal was to use a microcontroller so I could receive push notifications via e-mails or text messages. In addition to emergency messages, status messages sent on a regular basis reassure me the system is running. I also wanted to use a web browser to check the current status at any time.

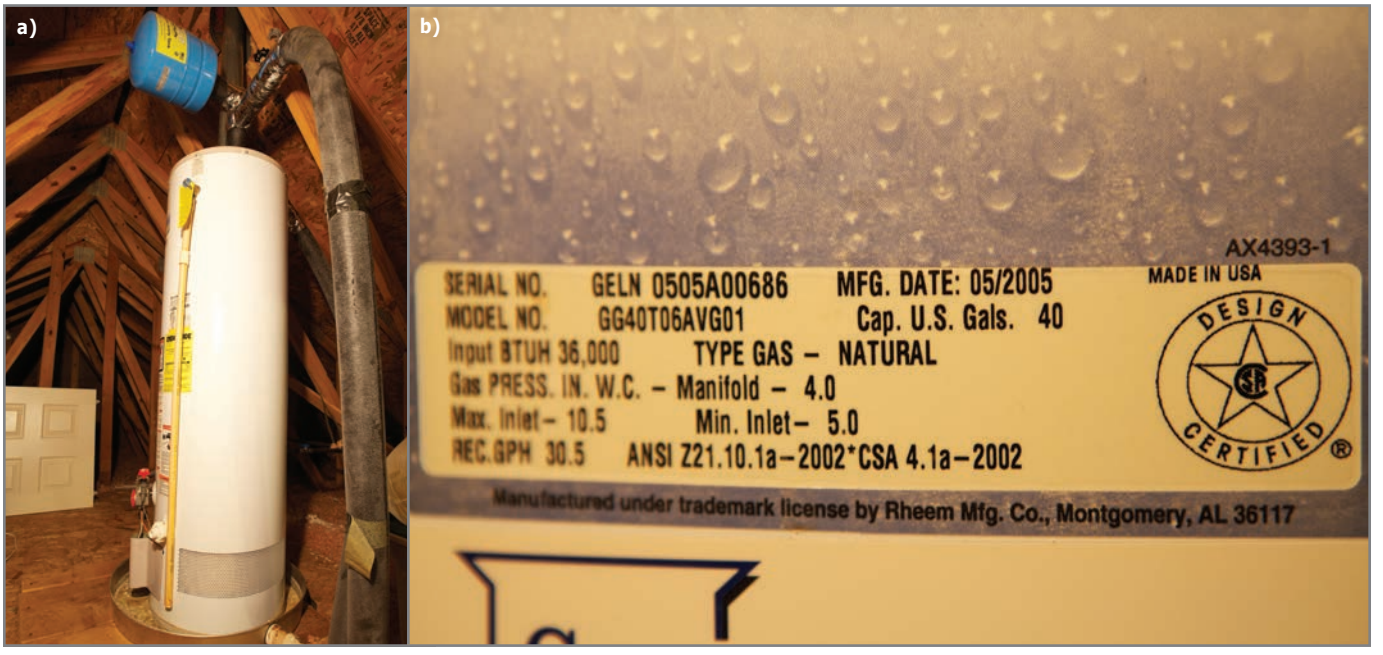


Photo 1a—My home's water heater is located in our attic. **b—**The water heater's label shows it was manufactured in 2005. (Photos courtesy of Michael Thomas)

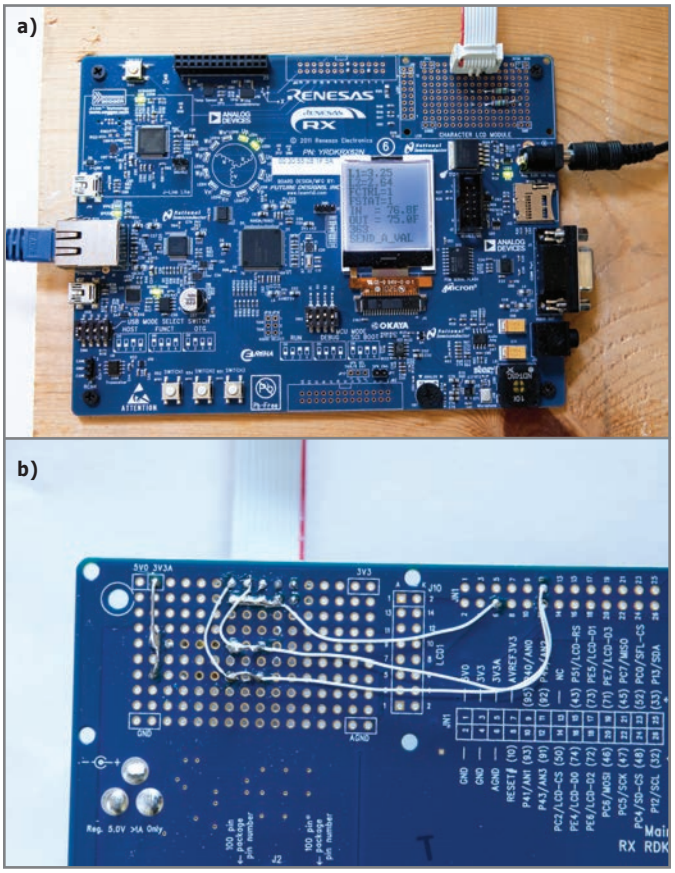


Photo 2a—The completed board, which is based on a Renesas Electronics YRDKRX62N demonstration kit. **b—**The back of the circuit board showing hand assembly (Photos courtesy of Michael Thomas)

Photo 3—A leak sensor (Photo courtesy of Michael Thomas)

MCU & SENSOR

The attic monitor is based on Renesas Electronics's YRDKRX62N demonstration kit, which features the RX62N 32-bit microcontroller (see Photo 2). Renesas has given away thousands of these boards to promote the RX, and the boards are also widely available through distributors. The YRDK board has a rich feature set including a graphics display, push buttons, and an SD-card slot, plus Ethernet, USB, and serial ports. An Analog Devices ADT7420 digital I²C temperature sensor also enables you to keep an eye on the attic temperature. I plan to use this for a future addition to the project that compares this temperature to the outside air temperature to control an attic fan.

SENSING WATER

Commercial water-detection sensors are typically made from two exposed conductive surfaces in close proximity to each other on a nonconductive surface. Think of a single-sided PCB with no solder mask and tinned traces (see Photo 3). These sensors rely on the water conductivity to close the circuit between the two conductors. I chose a sensor based on this type of design for its low cost. But, once I received the sensors, I realized I could have saved myself a few bucks by making my own sensor from a couple of wires or a piece of proto-board.

When standing water on the sensor shorts the two contacts, the resistance across the sensor drops to between 400 kΩ and 600 kΩ. The sensor is used as the bottom resistor in a voltage divider with a 1-MΩ resistor up top. The output of the divider is routed to the 12-bit analog inputs on the RX62N microcontroller. Figure 1 shows the sensor interface circuit. When the voltage read by the analog-to-digital converter (ADC) drops below 2 V, it's time to start bailing. Two sensors are connected: one in the catch pan under the water heater, and a second one just outside the catch pan to detect failures in the small expansion tank.

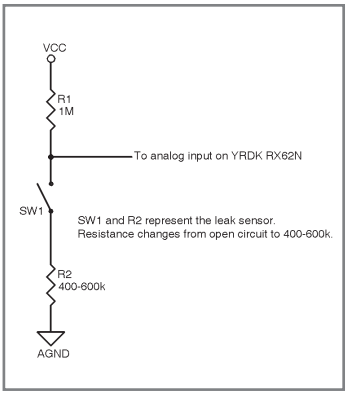


Figure 1—The sensor interface to the YRDK RX62N board

COMMUNICATIONS CHOICES

One of my project goals was to push notifications to my cell phone because Murphy's Law says water heaters are likely to fail while you're away for the weekend. Because I wanted to keep the project costs low, I used my home's broadband connection as the gateway for the attic monitor. The Renesas RX62N microcontroller includes a 100-Mbps Ethernet controller, so I simply plugged in the cable to connect the board to my home network. The open-source µIP stack supplied by Renesas with the YRDK provides the protocol engine needed to talk to the Internet.

There were a couple of complications with

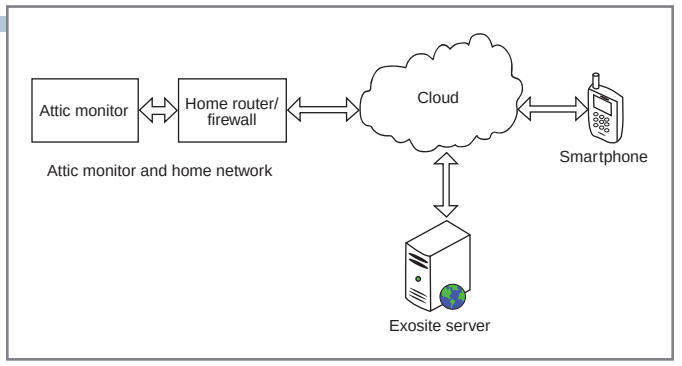


Figure 2—The cloud-based network

using my home network as the attic monitor's gateway to the world. It is behind a firewall built into my router and, for security reasons, I don't want to open up ports to the outside world.

My Internet service provider (ISP) occasionally changes the Internet protocol (IP) address associated with my cable modem. So I would never know what address to point my web browser. I needed a solution that would address both of these problems. Enter Exosite, a company that provides solutions for cloud-based, machine-to-machine (M2M) communications.

TALKING TO THE CLOUD

Exosite provides a number of software components and services that enable M2M communications via the cloud. This is a different philosophy from supervisory control and data acquisition (SCADA) systems I've used in the past. The control systems I've worked on over the years typically involve a local host polling the hundreds or thousands of connected sensors and actuators that make up a commercial SCADA system. These systems are generally designed to be monitored locally at a single location. In the case of the attic monitor, my goal was to access a limited number of data points from anywhere, and have the system notify me rather than having to continuously poll. Ideally, I'd only hear from the device when there was a problem.

Exosite is the perfect solution: the company publishes a set of simple application programming interfaces (APIs) using standard web protocols that enable smart devices to push data to their servers in the cloud in real time. Once the data is in the cloud, events, alerts, and scripts can be created to do different things with the data—in my case, to send me an e-mail and SMS text alert if there is anything wrong with my water heater. Connected devices can share data with each other or pull data from public data sources, such as public weather stations. Exosite has an industrial-strength platform for large-scale commercial applications. It provides free access to it for the open-source community. I can create a free account that enables me to connect one or two devices to the Exosite platform.

Embedded devices using Exosite are responsible for pushing data to the server and pulling data from it. Devices use simple HTTP requests to accomplish this. This works great in my home setup because the attic monitor can work through my firewall, even when my Internet provider occasionally changes the IP address of my cable modem. Figure 2 shows the network diagram.

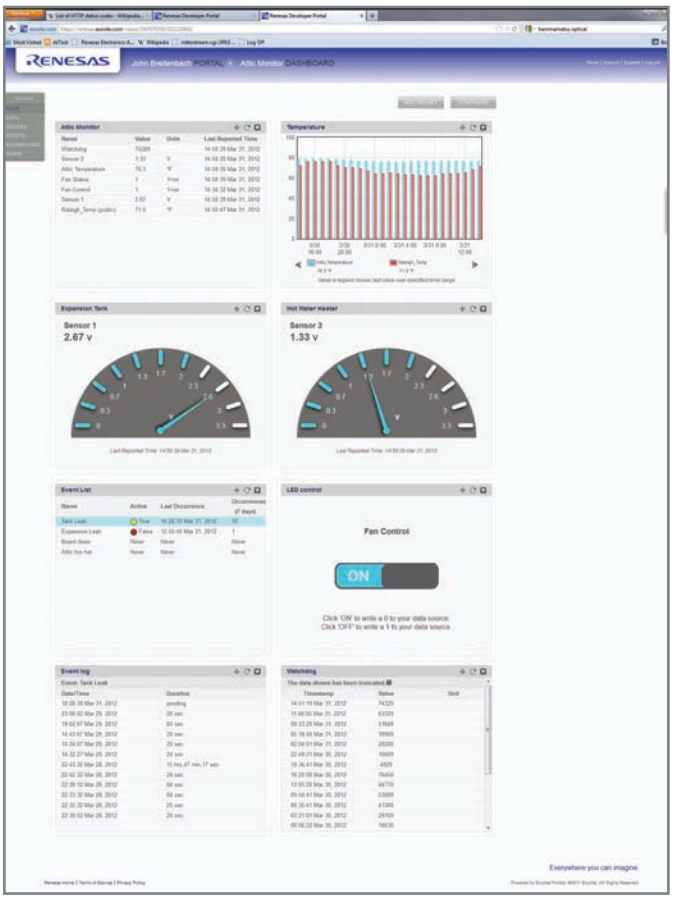


Photo 4—Exosite dashboard for the attic monitor

VIRTUAL USER INTERFACE

Web-based dashboards hosted on Exosite's servers can be built and configured to show real-time and historical data from connected devices. Controls, such as switches, can be added to the dashboards to push data back down to the device, enabling remote control of embedded devices. Because the user interface is "in the cloud," there is no need to store all the user interface (UI) widgets and data in the embedded device, which greatly reduces the storage requirements. Photo 4 shows the dashboard for the attic monitor.

Events and alerts can be added to the dashboard. These are logical evaluations Exosite's server performs on the

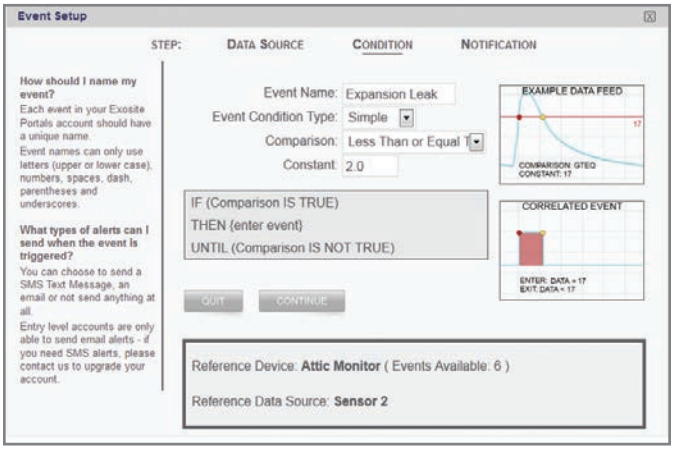
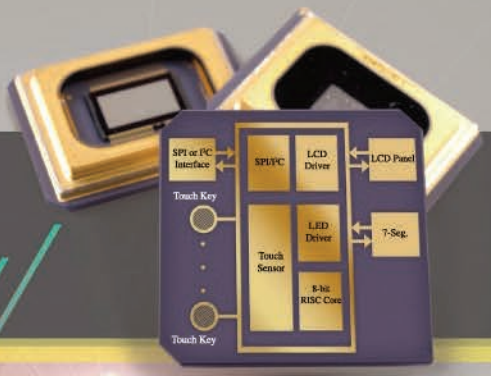


Photo 5—Creating an event in Exosite

HOLTEK
www.holtek.com

Precise Touch Control

Touch Control Solutions with HTouch™



Holtek's new generation BS83, BS84, BS85 and BS87 Flash MCU series, with their fully integrated touch switch circuits, as well as internal communication and display functions ensure a much reduced application external component count. With their high S/N ratio touch switch control structure, automatic compensation for power fluctuations and enhanced noise immunity, these new device ranges provide the perfect platform for extremely stable and effective touch switch solutions. A full feature MCU ensures touch switch product application solutions can be implemented with a minimum of development effort.

- Touch Key Flash MCU
- Touch keys functions require no external components
- High precision internal system oscillator
- Integrated SPI/I^{2}C serial communication interface
- Integrated LED and LCD driver circuits
- Integrated EEPROM

Part No.	MCU	AD	LCD driver	LED driver	Keys
BS83xxx	V				4/8/12/16
BS84xxx	V	V			8/12
BS85xxx	V		V	V	12/20
BS87xxx	V	V	V	V	12/24

Touch Flash MCU	STD Flash MCU	STD 8051 Flash MCU	USB STD Flash MCU	32bit MCU	Enhanced OTP MCU
TinyPower™ MCU	Power Management	UART MCU	Phone MCU	EEPROM	WLED Backlight

HOLTEK SEMICONDUCTOR INC.

Holtek Semiconductor (USA), Inc.
46729 Fremont Blvd., Fremont, CA 94538 Tel: (510)252-9880 Fax: (510)252-9885
E-mail: sales@holtek.com http://www.holtek.com

Listing 1—The POST request

```
POST /api:v1/stack/alias HTTP:1.1
Host: m2.exosite.com
X-Exosite-CIK: ceb78a461dff1e724b479cac6fefc3a5b1b50424
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Content-Length: 54

adc1=1.07&adc2=0.53&ambient=78.0&fido=149&fan_status=1
```

incoming data. Events can be triggered based on simple comparisons (e.g., a data value is too high or too low) or complex combinations of a comparison plus a duration (e.g., a data value remains too high for a period of time). Setting up a leak event for one of the sensors is shown in Photo 5. In this case, the event is triggered when the reported ADC voltage is less than 2 V. An event can also be triggered if Exosite doesn't receive an update from the device for a set period of time. This last feature can be used as a watchdog to ensure the device is still working.

When an event is triggered, an alert can optionally be sent via e-mail. This is the final link that enables an embedded device in my attic to contact me anywhere, anytime, to alert me to a problem. Though I have a smartphone that enables me to access my e-mail account, I can also route the alarm message to my wife's simpler phone through her cellular provider's e-mail-to-text-message gateway. Most cellular providers offer this service, which works by sending an e-mail to a special address containing the cell phone number. On the Verizon network, the e-mail address is <yourcellularnumber>@vtext.com. Other providers have similar gateways.

The attic monitor periodically sends heartbeat messages to Exosite to let me know it's still working. It also sends the status of the water sensors and the current temperature in the attic. I can log in to Exosite at any time to see my attic's real-time status. I have also configured events and alarms that will notify me if a leak is detected or if the temperature gets too hot.

INTERNET ENGINE

One of my goals was to keep the project low cost by using open source and free resources wherever possible. The attic monitor makes use of Adam Dunkel's open-source µIP Ethernet stack. The stack works in Polled mode with a simple timer. A super loop checks to see if any data has been received. If there is new data to be processed, it is passed up the stack to the application layer. Any queued outbound data is sent in the same super loop and stack timers are checked to handle other network tasks such as address resolution protocol (ARP) housekeeping. This stack won't win any performance contests, but you can't beat its simplicity, low cost (free!), and tiny code size. For an application like this, it's perfect.

The µIP super loop also makes a call to `user_app()`, which is the function you fill in with your application code. In the attic monitor, `user_app()` runs a simple state machine that periodically reads the states of the sensors and sends the readings to the cloud.

SEEDING THE CLOUD

The attic monitor establishes a connection with Exosite by opening a TCP/IP socket aimed at the IP address of their server using port 80, which is the standard port for HTTP requests. Once the connection is nailed up, the board sends the current values of all its sensors with a simple HTTP `POST` request. Listing 1 is the `POST` request, most of which is static. The portions in bold are dynamic and must be updated by the application. Each line is terminated by a carriage return/line-feed combination. The third line contains an Exosite client-identification key (CIK), which devices can use to uniquely identify themselves. As a device is provisioned on Exosite, the server provides a CIK. This key must then be programmed into your device and provided with each HTTP request.

The `Content-Length` portion of the header specifies the length of the payload to follow. The payload contains pairs of aliases and matching values, with each pair separated by an "&" character. The alias is a unique string that identifies a piece of data being sent by the device. In the attic monitor, the aliases `adc1` and `adc2` are used to send the voltage read from leak sensors. The number following the "=" is the actual voltage. Since the RX62N includes a floating-point unit, converting the ADC counts to voltage is practically free—it only takes a few microseconds—and I can send all data in engineering units. See Table 1 for a list of the aliases used by the attic monitor.

If the message was correctly formatted and received without errors by Exosite, the server responds with an appropriate HTTP status code. Status code "204" means the message got through. Problems on either the client or server side are indicated by other values.

```
HTTP/1.1 204 No Content
Date: Tue, 27 Mar 2012 21:08:42 GMT
Server: misultin/0.8.1-exosite
Connection: Keep-Alive
Content-Length: 0
```

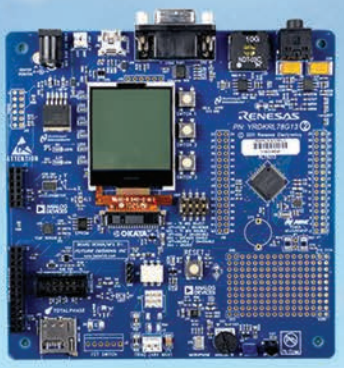
READING THE CLOUD

Just as the attic monitor uses a `POST` request to send data to the cloud, a `GET` request enables the monitor to fetch data

Name	Alias	Type	Source	Description
Sensor 1	adc1	Float	Device	Voltage from leak sensor 1
Sensor 2	adc2	Float	Device	Voltage from leak sensor 2
Attic temperature	ambient	Float	Device	Attic temperature in °F
Fan status	fan_status	Integer	Device	Current fan status (1 = On)
Watchdog	fido	Integer	Device	Watchdog indicator from device
Fan control	fan_ctrl	Integer	Dashboard control	Sends state of On/Off switch from dashboard to device to control the fan.
Raleigh_Temp	outdoor	Float	Public source	Temperature at RDU airport from public Exosite source. Sent to device for fan control.

Table 1—Attic monitor aliases for communications with Exosite

green energy
energy efficient
low power
hydropower
home automation
solar
embedded
Renesas RL78 MCU
metering controllers
monitoring
turbines
reduce
bio-technology
high performance
win big
prizes



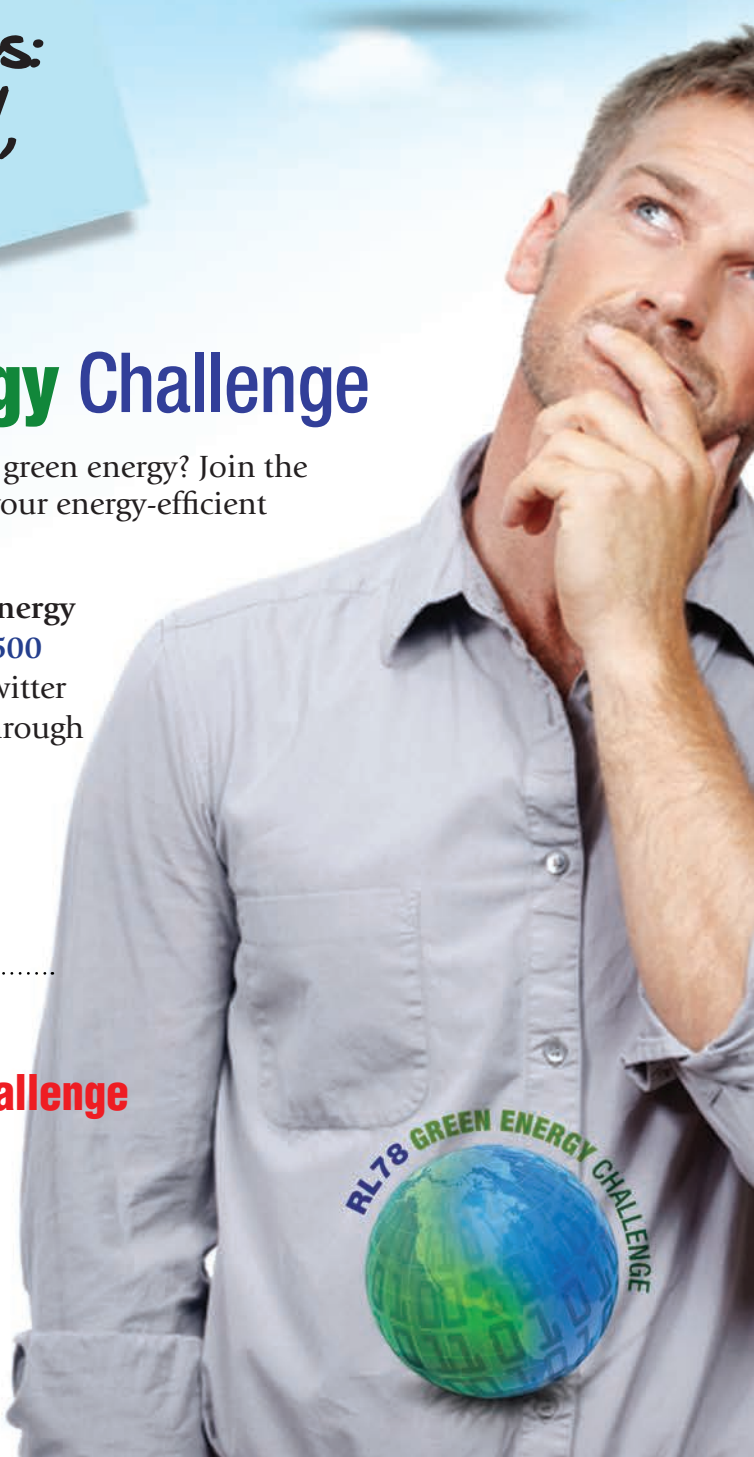
Deadline for entries: August 31, 2012

The RL78 Green Energy Challenge

Do you want to influence how the world experiences green energy? Join the RL78 Green Energy Challenge today and show how your energy-efficient design solution can contribute to a "greener" world.

Use an RL78 MCU to develop a low-power, green-energy design solution and you could win share of a $17,500 cash grand prize. Plus, keep following Renesas on Twitter and Facebook for a chance to win additional prizes through weekly challenges.

RENESAS RL78



For complete details, visit

www.circuitcellar.com/RenesasRL78Challenge

@RenesasAmerica
www.facebook.com/renesasamerica

Participation in Weekly Challenges and receipt of partner prizes is not a factor in selecting winners for the Cash Grand Prize from Renesas. See website for complete rules and details. Void where prohibited by law.

In association with *Elektor* and *Circuit Cellar*

from the cloud. This data can be sourced from other embedded devices connected to Exosite, from public data sources available on Exosite (e.g., local weather), or from widgets on an Exosite dashboard. This is where M2M communication succeeds.

My future plan for the attic monitor is to add intelligent fan control that makes the decision to run the fan based not only on the attic temperature, but also on the outside air temperature. I also want to be able to manually control the fan. The monitor needs to fetch two pieces of information from the cloud: a fan-control flag and the outside air temperature. The fan-control flag is set by the dashboard user interface with a switch widget. The outside air temperature is fetched from an Exosite public data source: the temperature at the Raleigh airport. The HTTP GET request and response are similar in syntax to the POST request. The desired aliases are sent in the GET request, and the Exosite server replies with the corresponding values in the status response. Here's the request:

```
GET /api:v1/stack/alias?fan_ctrl&outside HTTP/1.1
Host: m2.exosite.com
X-Exosite-CIK: ceb78a461dff1e724b479cac6fefc3a5b1b50424
Accept: application/x-www-form-urlencoded; charset=utf-8
```

The list of desired aliases is sent, each separated by an "&". In this request, the device is asking the cloud for current values associated with the aliases fan_ctrl and outside. As with all requests, the device's unique CIK is also sent to verify its identity to the server. The response is a simple message that must be parsed by the device:

```
HTTP/1.1 200 OK
Date: Tue, 27 Mar 2012 21:08:32 GMT
Server: misultin/0.8.1-exosite
Connection: Keep-Alive
Content-Length: 23

outside=61.0&fan_ctrl=1
```

The latest values of the requested aliases are included along with a timestamp. In this example, you can see the current temperature at Raleigh-Durham International Airport, alias outside, is 61°F.

CONSTRUCTION

Construction was simple since most of the electronics were already provided by the YRDKRX62 kit. A small prototyping area on the board provides the needed real estate to add the passives and connectors needed to attach the leak sensors. Up in the attic, one sensor is mounted in the catch pan under the water heater, and another is mounted on the floor just outside the catch pan under the expansion tank.

CHECKING FOR RAIN

After a couple of hours of lashing together some software and hardware, my device monitors a few sensors and periodically transmits the data to the cloud. It also reads data from the cloud that originates in other devices, virtual user interfaces (i.e., dashboards), and public data sources. The final piece of the puzzle is to set up notifications to alert me if it is raining in my attic. This is a simple matter on Exosite. Photo 6 shows the events and corresponding alerts that notify me the instant something goes wrong in the attic. System checkout is a matter of licking your finger and placing it on the leak sensor. Sure enough, within about 30 s I heard a "bing" from my cell phone alerting me that the low voltage had been reported to Exosite, and that an e-mail had been sent indicating a leak. Photo 7 shows the results.

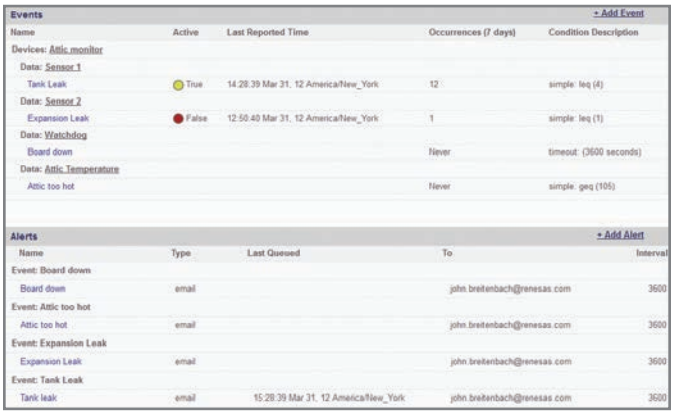


Photo 6—A list of Exosite events and alerts

FUTURE EXPANSION & IMPROVEMENTS

The YRDKRX62N kit has an ADT7420 digital I²C temperature sensor. The ADT7420 has ±0.25°C accuracy and 16-bit resolution, which makes it perfect to read ambient conditions in the attic. The attic monitor also has real-time access to the outside air temperature from the Exosite public source. Without running any additional wires or adding any extra sensors, the attic monitor can use this public data to make intelligent decisions about when to cycle the attic fan. The monitor already includes all the communications hooks needed: it sends the attic temperature to the Exosite portal, fetches the outside air temperature and manual fan override from the cloud, and reports the fan status back to the portal. The fan output is currently an LED on the YRDKRX62N board that lets me know all the communications are working. It's fun to log in to the portal with my cell phone and use the dashboard to control the LED.

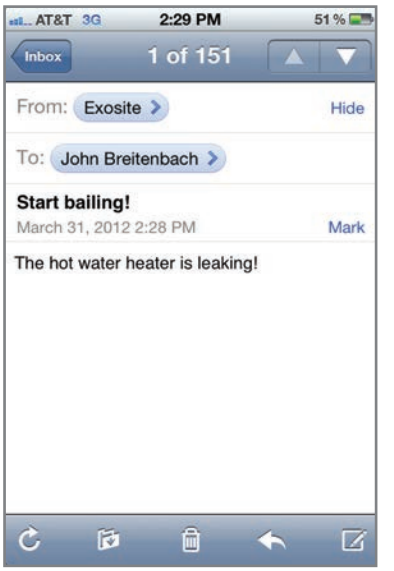


Photo 7—Here's what an alert looks like when it hits my phone.

Have you ever gone in your attic to find the light has been on for the past month? I have,

and one of the planned additions to the attic monitor is a simple ambient light sensor. I can use an Exosite event to send an alert if the light is on for too long, say, a couple of hours. Another planned addition is a local annunciator to send the alarm if water is detected.

FAST, FUN & LESS THAN $50

I'm still amazed that it is now so easy to quickly and inexpensively put together a complex, remote-sensing system. The total cost was less than $50 (my YRDKRX62 was free). Because Exosite uses simple HTTP requests, it only took a few hours to modify the example µIP demo to create a custom device with the data profile to match my application. And Exosite's cloud-based One Platform handles all of the data logging, alarming, and UI heavy lifting.

Five to 10 years ago, before the advent of cloud computing, a project like this would have been a significant undertaking. The code and memory requirements for the target microcontroller would have been much greater, and I would have had to host the UI on the device in an embedded web server, or write a custom protocol to run over a TCP/IP socket and lash it to a custom user interface on a PC. Then I would have had to figure out how to tunnel through my firewall, or it would have been necessary to leave some ports open to the outside world.

I've been programming microcontrollers for more than 20 years—since back when the "Circuit Cellar" column was featured in *Byte* magazine—and it's never been so much fun.

Before joining Renesas as an MCU applications engineer, John Breitenbach (john.breitenbach@renesas.com) spent more than 20 years developing embedded systems for everything from elevators, HVAC controls, and oven monitors to artificial hearts, infrared spectrophotometers, and urodynamic diagnostic systems. He lives with his family in Cary, NC.

PROJECT FILES

To download the project files and application note, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/264.

RESOURCES

Renesas Electronics Corp., "Application Note R01AN 0169EU0101: RX62N Group µIP TCP/IP Protocol Stack Demonstration," 2011.

Renesas Exosite, http://renesas.exosite.com.

SOURCES

ADT7420 Digital I²C temperature sensor
Analog Devices, Inc. | www.analog.com

One Platform cloud-based data platform
Exosite | http://exosite.com

YRDKRX62N Demonstration kit and RX62N microcontroller
Renesas Electronics Corp. | www.renesas.com

Protected Home WS-600 Water sensor
Smarthome, Inc. | www.smarthome.com

A Vision for 3-D Stacked Systems

An Interview with Ayse Kivilcim Coskun

Ayse Kivilcim Coskun is making changes in the fields of energy efficiency, thermal challenges, and 3-D systems. The Boston University engineering professor recently received the A. Richard Newton Graduate Scholarship at the Design Automation Conference (DAC), a National Science Foundation (NSF) CAREER Award and a grant from the Massachusetts Green High-Performance Computing Center (MGHPCC). In May, Ayse and I discussed her engineering background, current courses, and passion for research.—Nan Price, Associate Editor

NAN: When did you first become interested in computer engineering?

AYSE: I've been interested in electronics since high school and in science and physics since I was little. My undergraduate major was microelectronics engineering. I actually did not start studying computer engineering officially until graduate school at University of California, San Diego. However, during my undergraduate education, I started taking programming, operating systems, logic design, and computer architecture classes, which spiked my interest in the area.

NAN: Tell us about your teaching position at the Electrical and Computer Engineering Department at Boston University (BU).

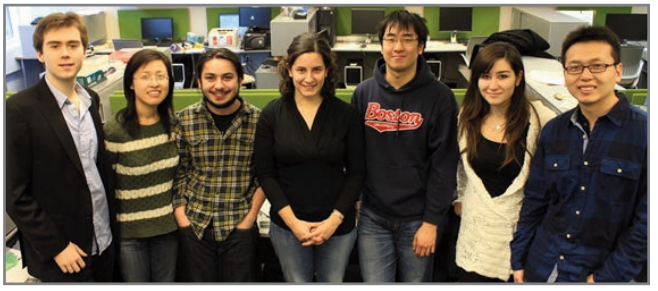
AYSE: I have been an assistant professor at BU for almost three years. I teach Introduction to Software Engineering to undergraduates and Introduction to Embedded systems to graduate students. I enjoy that both courses develop computational thinking as well as hands-on implementation skills. It's great to see the students learning to build systems and have fun while learning.

NAN: As an engineering professor, you have some insight into what excites future engineers. What "hot topics" currently interest your students?

AYSE: Programming and software design in general are certainly attracting a lot of interest. Our introductory software engineering class is attracting a growing number of students across the College of Engineering every year. DSP, image processing, and security are also hot topics among the students. Our engineering students are very keen on seeing a working system at the end of their class projects. Some project examples from my embedded systems class include embedded low-power gaming consoles, autonomous toy vehicles, and embedded systems focusing on healthcare or security applications.

NAN: Tell us about some of the projects you worked on at Sun Microsystems (now Oracle) before you began teaching at BU.

AYSE: I worked with the System Dynamics Characterization and Control (SDCC) team. The general focus of the group is developing measurement, diagnosis, design, or system management methods for improving performance, energy efficiency, and resilience. My research was focused on designing temperature management techniques. Specific projects included temperature-aware job scheduling in multicore servers, developing thermal prediction methods, and designing automated learning methods to understand system dynamics (such as temperature, throughput, etc.) while the servers are running.



Ayse (center) with a group of her students

NAN: How did you come to focus on energy efficiency and thermal challenges?

AYSE: Energy efficiency has been a hot topic for embedded systems for several decades, mainly due to battery-life restrictions. With the growth of computing sources at all levels—from embedded to large-scale computers, and following the move to data centers and the cloud—now energy efficiency is a major bottleneck for any computing system. The focus on energy efficiency and temperature management among the academic community was increasing when I started my PhD. I got especially interested in thermally induced problems as I also had some background on fault tolerance and reliability topics. I thought it would be interesting to leverage job scheduling to improve thermal behavior and my advisor liked the idea too. Temperature-aware job scheduling in multiprocessor systems was the first energy-efficiency related project I worked on.

NAN: In May 2011, you were awarded the A. Richard Newton Graduate Scholarship at the Design Automation Conference (DAC) for a joint project, "3-D Systems for Low-Power High-Performance Computing." Tell us about the project and how you became involved.

AYSE: My vision is that 3-D stacked systems—where multiple dies are stacked together into a single chip—can provide significant benefits in energy efficiency. However, there are design, modeling, and management challenges that need to be addressed in order to simultaneously achieve energy efficiency and reliability. For example, stacking enables putting DRAM and processor cores together on a single 3-D chip. This means we can cut down the memory access latency, which is the main performance bottleneck for a lot of applications today. This gain in performance could be leveraged to run processors at a lower speed or use simpler cores, which would enable low-power, high-performance computing. Or we can use the reduction in memory latency to boost performance of single-chip multicore systems. Higher performance, however, means higher power and temperature. Thermal challenges are already pressing concerns for 3-D design, as cooling these systems is difficult. The project focuses on simultaneously analyzing performance, power, and temperature and using this analysis to design system management methods that maximize performance under power or thermal constraints.

I started researching 3-D systems during a summer internship at the Swiss Federal Institute of Technology (EPFL) in the last year of my PhD. Now, the area is maturing and there are even some 3-D prototype systems being designed. I think it is an exciting time for 3-D research as we'll start seeing a larger pool of commercial 3-D stacked chips in a few years. The A. Richard Newton scholarship enabled us to do the preliminary research and collect results. Following the scholarship, I also received a National Science Foundation (NSF) CAREER award for designing innovative strategies for modeling and management of 3-D stacked systems.

"Our thermal modeling work includes modeling 3-D systems with liquid cooling, where water flows through microchannels in between different layers of the stack. Liquid cooling offers a lot of efficient cooling potential for high-performance 3-D stacks, and we've demonstrated tremendous cooling energy gains in our work."

NAN: In March 2012, you were awarded a Massachusetts Green High-Performance Computing Center (MGHPCC) grant. The grant will help finance your efforts to assess and advance the energy efficiency of large-scale computing. Tell us a little more about the grant and how it will be incorporated into your research.

AYSE: A majority of existing energy management methods are at the hardware level or at system management level (e.g., policies to turn off idle units). Little has been done, so far, to tune software applications for better energy efficiency. In fact, most application optimization has focused on improving performance. Our project's goal is to develop application optimization methods and derive guidelines for better software design to achieve energy-proportional computing at HPC clusters. We are specifically focusing on multithreaded, data-intensive loads that run on HPC centers.

MGHPCC will provide exciting opportunities for research and experimental evaluation. The seed funds ignited a number of multi-institutional collaborations already among BU, Massachusetts Institute of Technology, Harvard, and Northeastern. IEEE Spectrum hailed the center as one of the world's most ambitious university computer centers. Running our experiments on an industrial-scale HPC center will be a really unique opportunity, as such experiments are challenging to set up in university research labs.

NAN: Your website provides information on some of your current research projects, including modeling and management of 3-D stacked architectures and energy and thermal management of manycore systems. Tell us about these projects.

AYSE: Our research on 3-D stacked systems has several fronts. We have developed temperature models as thermal challenges are among major limiting factors for 3-D chips. Our thermal modeling work includes modeling 3-D systems with liquid cooling, where water flows through microchannels in between different layers of the stack. Liquid cooling offers a lot of efficient cooling potential for high-performance 3-D stacks, and we've demonstrated tremendous cooling energy gains in our work. In addition to modeling, we have worked on developing techniques to manage workloads, processor speed, and to tune the architecture for optimizing 3-D stacks for meeting desired performance-temperature-energy tradeoffs.

We are also working on designing energy and reliability management strategies for computing clusters, which contain hundreds of processing nodes and potentially have virtualization layers. Recently, we started researching on the impact on cooling as well, and now we are experimenting in our lab for using workload management and fan control jointly to optimize energy and thermal behavior.

NAN: Is there an electronics engineer or academic who has been the inspiration for the type of work you do today?

AYSE: I believe my professors at Sabanci University, Turkey, where I completed my bachelor's degree in electronics engineering, gave me the first inspiration to pursue an academic career. Later on, I was lucky to work with excellent, passionate researchers at UC San Diego, EPFL, and Sun Microsystems. I love the collaborative nature of engineering research, and now I maintain active collaborations with several companies and universities, while regularly interacting with my colleagues at BU. This interactive environment gives me a continuous research drive.

by Tom Struzik (USA)

USB on a Shoestring

You can replace a parallel port I/O device with a simple, low-budget USB device. A microcontroller and Atmel's LibUSB-Win32 enable you to produce a solution that works on 32-bit Windows XP up to and including 64-bit Windows 7.

Some time back, I had a garage recording studio. People always walked in on recording sessions, so I built a "Recording in Progress" light that was controlled by our recording software. This was simple because our recording software had a decent software development kit (SDK), and using the PC's parallel port to control the light was extremely easy. Even in Windows XP, where direct access to the hardware was becoming limited, parallel port access was still achievable by utilizing freely available kernel-mode drivers.

But then, new PC motherboards stopped having parallel ports. Worse, Microsoft significantly changed the Windows 7 driver architecture so that an old parallel port access driver would not load on Windows 7. Thus began my search for an inexpensive way to implement simple digital I/O on a PC.

THE SEARCH

Based on the features advertised for new motherboards, it was obvious that the motherboard market was moving away from parallel and serial port interfaces. Why fight the market? Could I find some way to easily recreate the parallel port digital I/O functionality with a USB interface? How hard could it be?

Any USB implementation consisted of two cooperating components: the device firmware and the Windows driver. One trick to reduce the development overhead was to use the preinstalled Windows human interface device (HID) drivers for the Windows driver component. The concept was to design a USB device to act like a fake keyboard, mouse, or other input device. This required sufficiently exploring the HID specifications to find a HID driver with enough functionality to be usable but not so much as to be overly complicated.

Understanding the complexities of the USB firmware necessary to present a fully compliant HID USB device was not a simple undertaking. There are products such as Trace Systems's HIDmaker FS source code compiler that help automate the necessary firmware generation, but these are not free tools. The fact that a market exists for such tools confirms that the HID driver approach would be too complicated for the simple 2-bit USB device I envisioned.

Finally, I came across Atmel's Application Note AVR309, which documented a pure software USB implementation. This application note presented a simple bit-bang USB implementation that worked for most AVR processors. Plus, it included the necessary Windows drivers and a matching Windows DLL. In fact, whipping up a prototype took about one evening's work.

I thought I had my answer...until I tried to run it on Windows 7. I found that the AVR309 driver didn't work on Windows 7. Porting an old driver to Windows 7 wasn't the easy path I was looking for either.

However, the firmware in Application Note AVR309 did have possibilities. I took some "liberties" with the USB implementation to make it simple and able to fit into the smallest generic AVR chips. AVR309 basically ignored the normal USB transfer mechanisms. Instead it utilized the USB command message for all its data transfers. Making the AVR309 firmware work with a normal Windows HID driver was fairly difficult, so I needed a different source for a Windows USB driver.

The answer to the Windows USB driver problem was in LibUSB-Win32, a GPL Windows port of a generic Linux USB

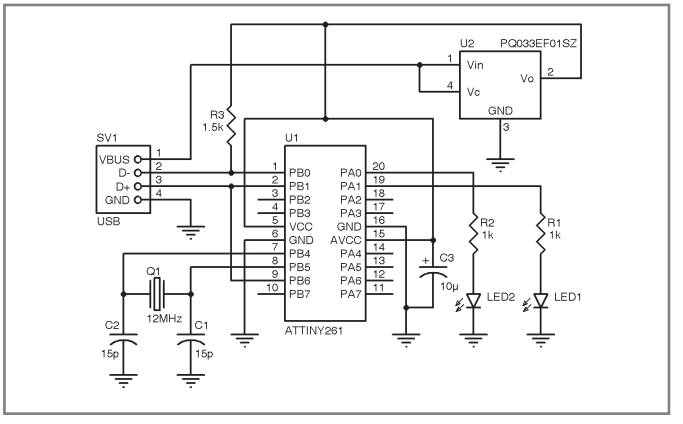


Figure 1—The circuit includes an Atmel ATtiny261, minimum USB, and two indicator LEDs.

driver. The Windows port not only supported Windows XP, but also both the 32-bit and 64-bit versions of Windows 7. LibUSB-Win32 even included a pre-signed driver so the driver would load on the 64-bit version of Windows 7.

LibUSB-Win32 included two methods to access a USB device. The first was a "filter" driver for situations where a normal USB driver already existed and the LibUSB-Win32 driver would be used in addition to the existing USB driver. The second method was to use a generic driver instead of another USB driver. The code also included the necessary DLL to interact with the LibUSB-Win32 driver. This DLL supported all the standard USB transfer methods and requests including vendor-specific control messages. This last part caught my eye because it was exactly what the AVR309 firmware was designed for—passing data via vendor-specific control messages.

The LibUSB-Win32 distribution looked complete. It included several wizards and test applications to assist with implementations. It also included the C header files and necessary libraries for GCC, BCC, and Microsoft Visual C 32-bit, i64, and x64 versions. As a bonus, the distribution included template install scripts and a few coding examples.

My solution for simple digital I/O via USB now included the AVR309 firmware and the LibUSB-Win32 drivers and utilities. This combination was the easy, cost-effective parallel port replacement I wanted.

FIRMWARE SOLUTION

I started on the firmware side of the solution by downloading and unpacking Application Note AVR309. It was useful to review the application note "doc" file that

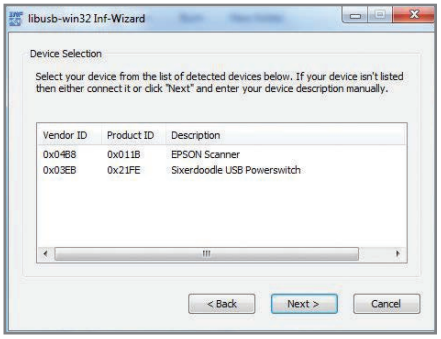


Photo 2—The INF wizard saw both my Atmel AVR309 firmware device and my more "normal" Epson scanner.

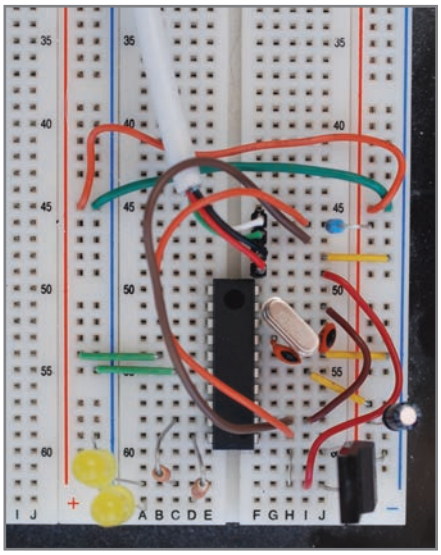


Photo 1—The breadboard: the prototype circuit, which is robust enough for a breadboard environment

outlined the organization of the firmware. The source code included in the application note was coded for an Atmel ATtiny2313 microcontroller, so some minor tweaks were required to port the code to the ATtiny261 microcontroller I planned to use.

The main code change was to specify which I/O pins to use for the USB interface and to properly configure the INT0 interrupt. There were also some optimizations in the hardware/firmware interface that had to be taken into account when laying out the hardware. The USB Data– line had to be tied to Pin 0 and the USB Data+ line had to be tied to INT0. This configuration is well documented in the Atmel application note, so I won't go into detail here. Figure 1 shows my resulting ATtiny261 circuit. The circuit was as bare bones as possible, consisting of only the ATtiny261, a 12-MHz crystal with bypass caps, a 1.5-KΩ pull-up resistor, and a 3.3-V regulator. The regulator was necessary because the processor had to run at 3.3 V so the I/O pin voltages matched the USB specifications for the Data± signal levels. The 1.5-KΩ pull-up on USB Data– was necessary to tell the host this was a low-speed device. Photo 1 shows the resulting breadboard.

At this point, firmware and circuit testing were limited to plugging the device into a USB port on my PC to ensure Windows simply recognized a new device had been plugged in. Windows searched and did not find a driver. Because the LibUSB-Win32 driver hadn't been loaded, I was hoping for this response. If you attempt this project, before plugging in your device, make sure the Vendor and Product ID combination defined in the AVR309 firmware *do not* match any existing USB device on the PC!

WINDOWS DRIVER SOLUTION

I started work on the Windows side of the solution by downloading and unpacking the latest LibUSB-Win32 zip file from the SourceForge website. The first task was getting Windows to load the generic LibUSB-Win32 driver for my AVR309 firmware device. LibUSB-Win32 simplified this step by providing a wizard (inf-wizard.exe) to automatically generate the necessary Windows INF file. The wizard scanned the current USB and displayed all the connected USB devices by Vendor ID, Product ID, and description. I simply selected my device and the wizard automatically generated the necessary INF file (see Photo 2).

Note: there is still no getting around the fact that the AVR309 firmware needs unique Vendor and Product IDs. For evaluation purposes, the default Vendor and Product IDs listed in the application note worked fine, but any duplication of Vendor and Product IDs with other devices existing on the PC *will cause problems*.

I used the new Windows .inf file to attach the generic LibUSB-Win32 driver to my device. Photo 3 shows how my new device showed up in the Windows Device manager. Note that it did not show up under the normal USB Device branch but rather under a branch specific to LibUSB-Win32 devices.

Once Windows officially recognized my device as a LibUSB-Win32 device, it was off

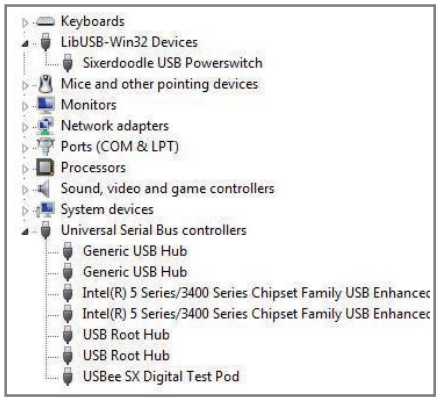


Photo 3—The Windows device manager. Notice that the LibUSB-Win32 devices do not show up under the normal USB controllers branch but rather a separate LibUSB-Win32 branch.

to write software to test access to the device. The steps involved with accessing a device with LibUSB-32 were to initialize the LibUSB-Win32 library and then enumerate the USBs and devices. After initializing LibUSB-Win32, I then had to iterate through the buses and devices to find the Vendor and Product IDs specific to my device. Finally, I had to issue a call to open the selected USB device. These steps are shown in Listing 1.

With the handle to the LibUSB-Win32 device, I then selected the device configuration and claimed the appropriate USB interface. Make sure to claim the interface before doing any operations with the device. Also, make sure to release the interface and close the device at the end of the program. These steps are also shown in Listing 1.

Now, I could finally start doing I/O. It was time to decipher the AVR309 firmware interface itself. The AVR309 firmware used a simple USB vendor control message for all of its communication. The LibUSB-Win32 call for sending a control message is simply `usb_control_msg`.

The arguments to the `usb_control_msg` call took a little deciphering and several arguments ended up being very specific to the AVR309 firmware implementation. The `usb_dev_handle` argument was obvious. It was the result from the `usb_open` call. The `requesttype` argument got a bit more interesting. It had to be a `USB_TYPE_VENDOR` request type and it had to be a `USB_ENDPOINT_IN` subtype.

The `request` argument, then, tells the AVR309 firmware exactly which internal firmware function to execute. Each firmware function within the AVR309 firmware has its own unique request number. The stock AVR309 firmware uses various request numbers, each with its own usage of the `value`, `index`, and `bytes` arguments. Table 3 in Application Note AVR309 details the various function codes and the associated purpose of the other three arguments. For example, the arguments `value` and `index` for Request Code 5, `DoSetOutDataPort`, contain the bits to send to I/O ports. Similarly, the `value` and `index` arguments for Request Code 7, `DoGetInDataPort`, are unused while the first three elements of the bytes array return the data read from I/O pins. The remaining arguments,

Listing 1—Bit toggling C code—the complete C code to toggle the Atmel ATtiny261's Port A0/A1 on and off

```
#include "stdafx.h"
#include "stdio.h"
#include "usb.h"                        // LibUSB-Win32 include file

#define MY_VID 0x03eb                   // VendorID as specified in AVR309
#define MY_PID 0x21fe                   // ProductID as specified in AVR309
#define MY_CONFIG 1                     // Must be 1 for AVR309
#define MY_INTF 0                       // Must be 0 for AVR309

#define SetDataPortDirection 3    // build in
#define GetDataPortDirection 4    // AVR309 USB Control requests
#define SetOutDataPort 5                // to set and get the AVR
#define GetOutDataPort 6                // IO port values
#define DoGetInDataPort 7


int _tmain(int argc, _TCHAR* argv[])
{

     struct usb_bus *bus;          // the USB busses
     struct usb_device *my_USBdev; // my USB device descriptor
     usb_dev_handle *dev = NULL;   // handle to my open USB device

     char Prod[256];               // buffer to start USB product
     //   name string
     int value;  // arguments for the usb
     int index;  //   control msg call
     char OutBuf[255];

     usb_init();  // Initalize libUSB-Win32 library
     usb_find_busses();            // Find all the busses
     usb_find_devices();           // Find all the devices
     //
     // Iterate through the all the devices looking for the VendorID
     // and ProductID that match our AVR309 device firmware
     //
     for (bus = usb_get_busses(); bus; bus = bus->next)
     {
          struct usb_device *USBdev;
          for (USBdev = bus->devices; USBdev; USBdev = USBdev->next)
          {
              if (USBdev->descriptor.idVendor == MY_VID
                       && USBdev->descriptor.idProduct == MY_PID)
              {
          my_USBdev = USBdev;      // remember our device descriptor
                 dev = usb_open(USBdev);

              }
          }
     }

     if (!dev)   // terminate if we failed to find our device
     {
          printf("error: device not found!\n");
          return 0;
     }

     //
     // Show how the simple string call works by getting
     // the product name from the AVR309 device firmware
     //
     usb_get_string_simple(dev, my_USBdev->descriptor.iProduct, Prod,256);
     printf("success: device %04X:%04X opened\n%s\n",MY_VID,MY_PID,Prod);

     if (usb_set_configuration(dev, MY_CONFIG) < 0)
     {
          printf("error: setting config #%d failed\n",MY_CONFIG);
     printf("error: setting config #%d failed\n",MY_CONFIG);
          usb_close(dev);
          return 0;
```

Listing continued on p. 32

HUMANDATA®

FPGA / CPLD Boards from JAPAN

SAVING COST=TIME with readily available FPGA boards

- Basic and simple features, single power supply operation
- Quality and reliability is provided by years of sales
- Same board size and connector layout – ACM/XCM series
- All stocked items are ready to be shipped immediately
- Over 100 varieties of FPGA/CPLD boards are available
- Customizing speed grade and/or any features are possible
- Free download technical documents before purchasing
- High quality and highly reliable FPGA/CPLD boards from Japan
- Almost all products are RoHS compliance

ALTERA FPGA Board

Arria II GX F572 FPGA board

ACM-025 series NEW

Arria II GX | DDR2

EP2AGX45DF25C6N
EP2AGX65DF25C6N
EP2AGX95DF25C6N
EP2AGX125DF25C6N
Credit card size (86 x 54 mm)

RoHS compliant

Cyclone IV E F484 FPGA board

ACM-023 series

Cyclone IV E | MRAM

EP4CE55F23C8N
EP4CE75F23C8N
EP4CE115F23C8N
Credit card size (86 x 54 mm)

RoHS compliant

CycloneIV GX F484 FPGA board

ACM-024 series

Cyclone IV GX | DDR2 | SIF40

EP4CGX50CF23C8N
EP4CGX75CF23C8N
EP4CGX110CF23C8N
EP4CGX150CF23C7N
Credit card size (86 x 54 mm)

RoHS compliant

Cyclone IV E F484 FPGA board

ACM-107 series

Cyclone IV E | MRAM

EP4CE55F23C8N
EP4CE75F23C8N
EP4CE115F23C8N
Compact size (43 x 54 mm)

RoHS compliant

XILINX FPGA Board

Spartan-6 FGG484 FPGA board

XCM-018/018Z series

Spartan-6 | MRAM | DDR2

XC6SLX45-2FGG484C
XC6SLX75-2FGG484C
XC6SLX100-2FGG484C
XC6SLX150-2FGG484C
Credit card size (86 x 54 mm)

RoHS compliant

Spartan-6 FGG484 FPGA board

XCM-110/110Z series

Spartan-6 | MRAM | DDR2

XC6SLX45-2FGG484C
XC6SLX75-2FGG484C
XC6SLX100-2FGG484C
XC6SLX150-2FGG484C
Compact size (43 x 54 mm)

RoHS compliant

Virtex-5 FFG676 FPGA board

XCM-011 series

Virtex-5 | FRAM | SDRAM

XC5VLX30-1FFG676C
XC5VLX50-1FFG676C
XC5VLX85-1FFG676C
XC5VLX110-1FFG676C
Credit card size (86 x 54 mm)

RoHS compliant

Virtex-5 LXT FFG665 FPGA board

XCM-017 series

Virtex-5 | SDRAM | RocketIO | SIF40

XC5VLX30T-1FFG665C
XC5VLX50T-1FFG665C
Credit card size (86 x 54 mm)

RoHS compliant

FPGA/CPLD Stamp Module PLCC68 Series

Easy and Quickly Mountable Module

FPGA Module IC socket mountable

- 50 I/Os (External clock inputs are available)
- 3.3V single power supply operation (Voltage converters for auxiliary power supply are built-in)
- Separated supply-inputs: Core, I/O drivers
- JTAG signal
- All PLCC68 series have common pin assignment
- Very small size (25.3 x 25.3 [mm])
- RoHS compliance
- MADE IN JAPAN

XILINX PLCC68 Series

XP68-03 Spartan-6 PLCC68 FPGA Module

NEW | Spartan-6 | PLCC68

XC6SLX45-2CSG324C
3.3V single power supply operation
On-board oscillator, 50MHz

RoHS compliant

XP68-02 Spartan-3AN PLCC68 FPGA Module

Spartan-3AN | PLCC68

XC3S200AN-4FTG256C
FPGA internal configuration ROM
Two User LEDs

RoHS compliant

ALTERA PLCC68 Series

AP68-04 Cyclone III PLCC68 FPGA Module

NEW | Cyclone III | PLCC68

EP3C25U256C8N
3.3V single power supply operation
On-board oscillator, 50MHz

RoHS compliant

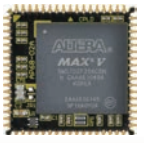
AP68-03 Cyclone III PLCC68 FPGA Module

Cyclone III | PLCC68

EP3C10U256C8N
4Mbit Configuration Device
Two User LEDs
One User Switch(Slide)

RoHS compliant

AP68-02 MAX V PLCC68 CPLD Module



MAX V | PLCC68

5M570ZF256C5N
External Clock inputs
On-board Voltage regulator

RoHS compliant

See all our products, download cables, A/D D/A conversion board, boards with USB chip from FTDI and accessories at :

www.hdl.co.jp/CC/

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

size and timeout, are straightforward. The size argument is the size of the bytes array and timeout is the number of milliseconds to wait for the return from the control message call. A simple Microsoft Visual C++ project to toggle bit A0 and A1 on and off is included on the *Circuit Cellar* FTP site.

At this point, I had a fully functional USB interface to read and write several bits of digital I/O with no-cost software and little hardware. This met my original goal. But, there's even more.

EXPANDING USES

Before I discuss more about what *can* be done with AVR309 firmware and LibUSB-Win32, let's talk about what *can't* be done. For one thing, high-throughput applications don't have a chance. The firmware only implements USB v1.0 low-bandwidth protocol and is theoretically limited to 1.5 Mb/s. However, while shoving data through a USB control message is easy, it is also highly inefficient, so it would likely never achieve the rated 1.5 Mb/s throughput.

The AVR309 firmware, at a minimum, requires a processor running at 12 MHz. At those processor speeds, almost all the processor cycles go to USB overhead leaving little time to do any actual work. So, without throwing more CPU power at the firmware, anything that relies on precise timing or a substantial number of processor cycles will be a lost cause.

But, given those limitations, it is relatively easy to extend the functionality of the AVR309 firmware. Already in the existing firmware, request codes 100 and above are defined as User Request codes. With these codes, it is easy to insert new user functions into the AVR309 firmware. When writing user functions, note that the value and index arguments are word values. Together they make up 4 bytes of data that can be sent to the AVR309 firmware device. Yes, 4 bytes is all that can be sent at once. This is the price to be paid for a simple interface. In the AVR309 firmware the value and index arguments are described in Table 1.

The firmware has somewhat more capacity to send data back to the host. The size of the return buffer (the bytes argument) has a maximum of 255 bytes.

Listing 1 continued from p. 30

```
    }

    if (usb_claim_interface(dev, MY_INTF) < 0)
    {
        printf("error: claiming interface #%d failed\n", MY_INTF);
        usb_close(dev);
        return 0;
    }

    value = 0x0003;              // DDRA 0,1 set to output
    index = 0x0000;
    printf("Set Port Dir returned %d\n",
    usb_control_msg( dev, USB_ENDPOINT_IN | USB_TYPE_VENDOR ,
    SetDataPortDirection, value, index, OutBuf, 1, 250) );

    for(int ii = 0; ii< 10 ; ii++)
    {
    value = 0x0002;      // Set PortA 0b00000010
    index = 0x0000;
    printf("Set Port Data %d returned  %d\n",     value,
        usb_control_msg(dev, USB_ENDPOINT_IN | USB_TYPE_VENDOR ,
        SetOutDataPort, value, index, OutBuf, 1, 250));
    Sleep(500);

    value = 0x0001; // Set PortA 0b00000001
    index = 0x0000;
    printf("Set Port Data %d returned  %d\n",     value,
        usb_control_msg(dev, USB_ENDPOINT_IN | USB_TYPE_VENDOR ,
        SetOutDataPort, value, index, OutBuf, 1, 250));
    Sleep(500);

    }
    value = 0x0000; // Set PortA 0b00000000
    index = 0x0000;
    printf("Set Port Data %d returned  %d\n",     value,
    usb_control_msg(dev, USB_ENDPOINT_IN | USB_TYPE_VENDOR ,
        SetOutDataPort, value, index, OutBuf, 1, 250));
    if(dev != NULL)
    {
        usb_release_interface(dev, MY_INTF);
        usb_close(dev);
    }
    printf("Done.\n");

    return 0;
}
```

Further, the firmware limits the number of bytes sent back to the size argument from the host; otherwise this can potentially overrun the host's data buffer.

The user function code can do whatever it needs to in order to process the 4 input bytes and generate a return buffer. The firmware will automatically return "busy" to the host until it finishes processing and generates a result. Once the input data is processed, it can return a "no data" message to the host by calling OneZero Answer, or it can call ComposeEndXXX Descriptor to return a specific number of bytes. The returned data will be contained in the bytes argument of the usb_control_msg call returned to the host.

The ComposeEndXXXDescriptor routine in the firmware is a bit complex. It can be used to return data from RAM, ROM, or EEPROM. The firmware variables that control the ComposeEndXXXDescriptor function are RAMREAD, which controls the source, and temp0, which controls the number of bytes to return. It took some digging to discern the arguments, but the findings are listed in Table 2.

InputBufferBegin+4 = low byte of value
InputBufferBegin+5 = high byte of value
InputBufferBegin+6 = low byte of index
InputBufferBegin+7 = high byte of index
InputBufferBegin+8 = size

Table 1—The usb_control_msg value and index argument mapping within the Atmel AVR309 firmware.

ROUTE FASTER !

WITH PROTEUS PCB DESIGN

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

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

PROTEUS DESIGN SUITE

Features:

- Hardware Accelerated Performance.
- Unique Thru-View™ Board Transparency.
- Over 35k Schematic & PCB library parts.
- Integrated Shape Based Auto-router.
- Flexible Design Rule Management.
- Polygonal and Split Power Plane Support.
- Board Autoplacement & Gateswap Optimiser.
- Direct CADCAM, ODB++, IDF & PDF Output.
- Integrated 3D Viewer with 3DS and DXF export.
- Mixed Mode SPICE Simulation Engine.
- Co-Simulation of PIC, AVR, 8051 and ARM7.
- Direct Technical Support at no additional cost.

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

labcenter Electronics

www.labcenter.com

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

So, despite the small amount of data transfer and limited speed, there are many other things this software can do. Here are some examples of what can be done with these limited resources. You can add a serial peripheral interface (SPI) to a PC, for example to support a small 2 × 64-character LCD. SPI is a perfect interface for AVR309 firmware because it is self-clocking and isn't impacted by jitter. Better still, character displays are typically very low bandwidth, so the meager 4 bytes per USB control message aren't likely to be an issue. The code I used to insert a bit-bang SPI output function into the AVR309 firmware is also available on *Circuit Cellar*'s FTP site.

You can also add a simple, multichannel analog-to-digital converter (ADC) input to a PC. Most AVR processors have a multichannel ADC built in. As long as fast sample rates aren't needed, it's easy to use one or two command messages to configure the ADC and then read the result with a third command message. It's slow, but it can still add an eight-channel ADC to a PC with relative ease. On my system, with a 12-MHz ATtiny261, I was able to get an ADC sample rate of around 300 Hz. Not bad for a bit-bang USB interface.

MORE METHODS FOR SUCCESS

What if you don't like programming in C? There are several other language wrappers available for LibUSB-Win32, including: C# and .Net, Perl, Python, and Java, as well as others. So don't let lack of C programming experience stop progress. Listing 2 shows a Perl implementation of the same functionality shown in Listing 1.

I came pretty close to my original goal. With the AVR309 firmware and LibUSB-Win32, I produced firmware that will work on almost any AVR processor and I did it with "free" open-source drivers. The solution even supported the dreaded Windows 7/64-bit driver environment. It's not fast, but it's cheap, and it works!

POSTLUDE

After researching and writing this article, I stumbled across Objective Development Software's V-USB, a GNU general public license version of a "Virtual USB

RAMread = Bit mask indicating memory space to return data from
Bit1,0 = 00, return data from ROM
Bit1,0 = 01, return data from RAM
Bit1,0 = 10, return data from EEPROM
Bit2 = 1, insert zero every other byte to build Unicode string
Bit3 = 1, do NOT insert zero on first byte, for Unicode length
Bit7 = 1, clear RAMread after argument after accessing
temp0 = number of transmitted data bytes
ZH,ZL = pointer to buffer of transmitted data
(ROM/RAM/EEPROM as specified in RAMread)

Table 2—Atmel AVR309 firmware routine `ComposeEndXXXDescriptor` arguments

Listing 2—This is the same functionality as the C code in Listing 1, but this time in Perl.

```
#!/usr/bin/perl

use Device::USB;
use strict;
use warnings;

use constant USB_ENDPOINT_IN => 0x80;
use constant USB_TYPE_VENDOR => (0x02 << 5);

use constant MY_VID =>  0x03eb;
use constant MY_PID => 0x21fe;
use constant MY_CONFIG => 1;
use constant MY_INTF => 0;

use constant SetDataPortDirection => 3;
use constant SetOutDataPort => 5;

my $usb = Device::USB->new();
my $dev = $usb->find_device( MY_VID, MY_PID );

printf "Device: %04X:%04X\n", $dev->idVendor(), $dev->idProduct();
$dev->open();
print "Manufactured by ", $dev->manufacturer(), "\n",
          " Product: ", $dev->product(), "\n";
print "set config=",$dev->set_configuration( MY_CONFIG ),"\n";
print "Claim =",$dev->claim_interface(MY_INTF),"\n";

my $bytes = "\x00" x 2;

my $value = 0x0003;
my $index = 0x0000;
print "Set DDRA returned ",
        $dev->control_msg( USB_ENDPOINT_IN | USB_TYPE_VENDOR,
        SetDataPortDirection, $value, $index, $bytes, 1, 250 ),"\n";
my $ii;
for ($ii = 0; $ii <10; $ii++) {
    $value = 0x0001;
    $index = 0x0000;
    print "Set PortA ", $value, " returned ",
        $dev->control_msg( USB_ENDPOINT_IN | USB_TYPE_VENDOR,
        SetOutDataPort, $value, $index, $bytes, 1, 250 ),"\n";
    sleep 1;
    $value = 0x0002;
    $index = 0x0000;
    print "Set PortA ", $value, " returned ",
        $dev->control_msg( USB_ENDPOINT_IN | USB_TYPE_VENDOR,
        SetOutDataPort, $value, $index, $bytes, 1, 250 ),"\n";
    sleep 1;
}
$value = 0x0000;
$index = 0x0000;
print "Set PortA ", $value, " returned ",
        $dev->control_msg( USB_ENDPOINT_IN | USB_TYPE_VENDOR,
        SetOutDataPort, $value, $index, $bytes, 1, 250 ),"\n";
$dev->release_interface(MY_INTF);
```

Port for AVR Microcontrollers." With it's V-USB template, I produced a drop-in replacement for the AVR309 firmware that uses the same LibUSB-Win32 driver and interface DLL and exactly the same circuit.

Objective Development's V-USB implementation has three advantages over the AVR309 Application Note implementation. First, V-USB is still being actively maintained. Second, the majority is written in C rather than Assembly, making modifications much easier. Third, it solves the problem of how to obtain a USB Vendor/Product ID by providing a "generic" ID, which everyone using the tool can share as long as they follow several simple rules. Read the "readme" file that comes with Objective Development's software for more information on those rules. The AVR Studio project for my V-USB firmware and the AVR309 Application Note firmware utilized in this article are both available on the *Circuit Cellar* FTP site.

Tom Struzik (tpstruzik@earthlink.net) has been building and taking things apart from an early age. He built his first Heathkit project at age 12 and sold his first computer program at age 16. Tom has a BSEE from Purdue University and is an IT systems architect in the engineering organization of a Fortune-100 chemical company. Tom continues to write software and build hardware projects at home to "keep his hands dirty." He is currently in a state of wonder over his baby daughter. The bets are on to see if she gets Tom's engineering gene, her mother's music gene, or some combination thereof. Visit Tom at www.JenRathbun.com/Electronics.

PROJECT FILES

To download the project files and application note, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/264.

RESOURCES

Atmel Corp., AVR309 Application Bundle, www.atmel.com/dyn/resources/prod_documents/AVR309.zip.

———, AVR309: "Software Universal Serial Bus (USB)," 2006, http://www.atmel.com/Images/doc2556.pdf.

Google Code, perl-device-usb, http://code.google.com/p/perl-device-usb.

Java libusb, libusb-win32 Wrapper, http://libusbjava.sourceforge.net/wp.

Objective Development Software, "V-USB: Virtual USB Port for AVR Microcontrollers," www.obdev.at/products/vusb/index.html.

SourceForge, libusb-win32, http://sourceforge.net/apps/trac/libusb-win32/wiki.

———, LibUsbDotNet C# USB Library, http://sourceforge.net/projects/libusbdotnet.

SOURCES

ATtiny2313 and ATtiny261 MCUs and AVR309 USB
Atmel Corp. | www.atmel.com

Windows 7 operating system and Visual C++
Microsoft Corp. | www.microsoft.com

HIDmaker FS Compiler
Trace Systems, Inc. | www.tracesystemsinc.com

by Tommy Tyler (USA)

Build an MCU-Based Digital Thermometer

It's easy to convert an unused digital photo frame into a customized outdoor thermometer. This project can be accomplished with a circuit board, a digital temperature sensor, and an SD memory card loaded with temperature images.

Wondering what to do with your unused digital photo frame? With a little effort, a tiny circuit board assembly can be installed in the frame to transform the colorful thin film transistor (TFT) screen into the "ultimate" outdoor thermometer display (see Photo 1). Imagine a thermometer with real numeric digits (not seven-segment stick figures) large enough to be read from 40' to 50' away under any lighting conditions. Combine that with a glare-free, high-contrast screen, wide viewing angles, and an accuracy of ±0.5°F without calibration, and you have a wonderful thermometer that is more a work of art than an instrument, and can be customized and proudly displayed.

Almost any size and brand digital photo frame can be used, although one with 4.5" or 7" (diagonal) screen size is ideal for 2"-high digits. If you don't have a discarded frame to use, some bargains are available for less than $30, if you look for them. Search online for overstocked, refurbished, or open-box units. The modifications are easy. Just drill a few holes and solder a few wires. The postage-stamp size PCB is designed with surface-mount components, so it's small enough to tuck inside the frame. None of the modifications prevent you from using the frame as it was originally intended, to display photographs.

Photo 1—A TFT screen is easily transformed into an outdoor thermometer with the addition of a small circuit board.

PHOTO FRAME DISPLAY

Although digital photo frames vary in details and features, their basic functions are similar. Nearly all of them can store pictures in external memory, usually a small SD card like those used in digital cameras. Most have a half dozen or so push-button switches that control how the frame operates and select what is being displayed. There's usually a Menu button, an Enter or Select button, and several cursor buttons for navigating through on-screen menus.

Photo frames feature a slideshow viewing mode that automatically steps through pictures in sequence. You can set the time each picture is displayed to your preference. You can also turn off the timer and have a manual, single-step slideshow mode where a selected picture is continuously displayed until another is selected with a button press. That's the mode of operation used for the thermometer, and it is key to its accuracy.

The photo frame is loaded with images showing every possible temperature, in precise ascending order. Following power-up, the frame enters Slideshow mode displaying the first image in memory, which provides a known starting point. Based on repeated temperature measurements, the frame keeps incrementing or decrementing the image, 1° at a time, until the display matches the true temperature. After this initial synchronization, the display is simply incremented or decremented whenever the temperature rises or falls by 1° or more.

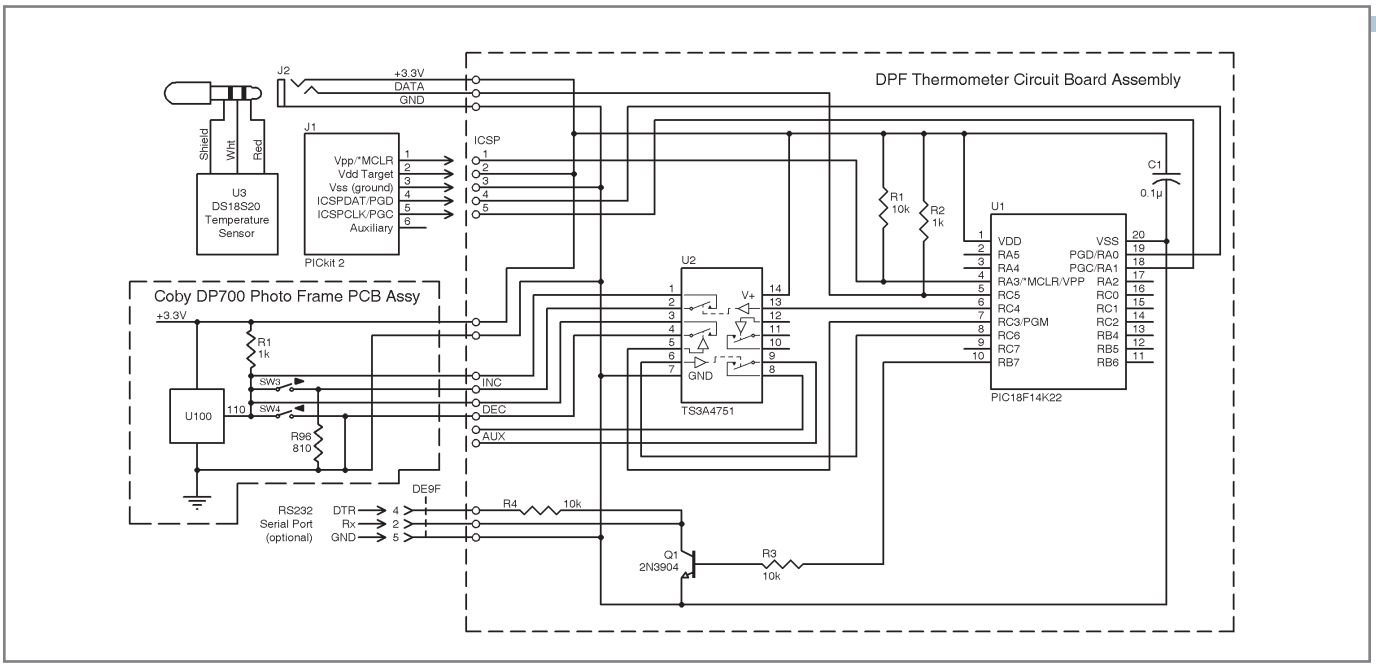


Figure 1—This schematic of the thermometer shows a portion of the Coby DP700 photo frame with a voltage comparator input that responds to different voltage levels from its >and< switches.

The frame responds so reliably, the display never gets out of sync with the true temperature. Following a power interruption, the thermometer automatically resynchronizes itself. In fact, for an interesting and reassuring demonstration at any time, just momentarily turn off power. Synchronization might take a minute or so due to the system's response time, but that's not considered a problem because presumably power interruptions will be infrequent.

CIRCUIT DESCRIPTION

Figure 1 shows a schematic of the thermometer. A Microchip Technology PIC18F14K22 microprocessor U1 periodically polls U3, a factory-calibrated "smart" temperature sensor that transmits the digital value of the current temperature via I/O pin RC5. PIC output pins RC4 and RC3 drive sections of U2, a Texas Instruments TS3A4751 quad SPST analog switch with extremely low on-state resistance. Two of these solid-state switches are wired in parallel with the mechanical switches in the frame that increment and decrement the displayed temperature. RC6 provides an auxiliary output in case you are working with a rare photo frame that requires a third switch be actuated to enter Slideshow mode.

Schematics for digital photo frames are practically nonexistent, but you won't need one. Your main concern is the switches that control how the frame operates. The large-scale processor chip in a photo frame is often starved for I/O pins, and multiple push-button switches sometimes share a common input pin. That's done by using an input configured as a voltage comparator and having different switches input different voltage levels when actuated. Figure 1 includes a portion of the Coby DP700 schematic showing such an arrangement. Switches SW3 (>) and SW4 (<) share input Pin 110 of the frame processor chip (U100). SW3 pulls the voltage down to about 1.5 V to increment the display, and SW4 pulls it all the way down to 0 V to decrement it. If you can gain access to the solder terminals of these switches, you can build this project. Using a solid-state analog switch for U2 enables the PIC control board to work with virtually any model photo frame, without having to worry about voltage, polarity, or switch circuit configuration.

PIC output RB7 continually transmits a running narrative of everything the thermometer is doing. Transistor Q1 provides a standard RS-232 serial output at 38400 bps, no parity, and two Stop bits using the DTR pin for pull-up voltage. This is mainly for testing, troubleshooting, or possibly experimenting with firmware changes. The board also includes a standard in-circuit serial programming (ICSP) interface for programming the PIC with a Microchip PICkit2 development programmer/debugger or similar programming tool.

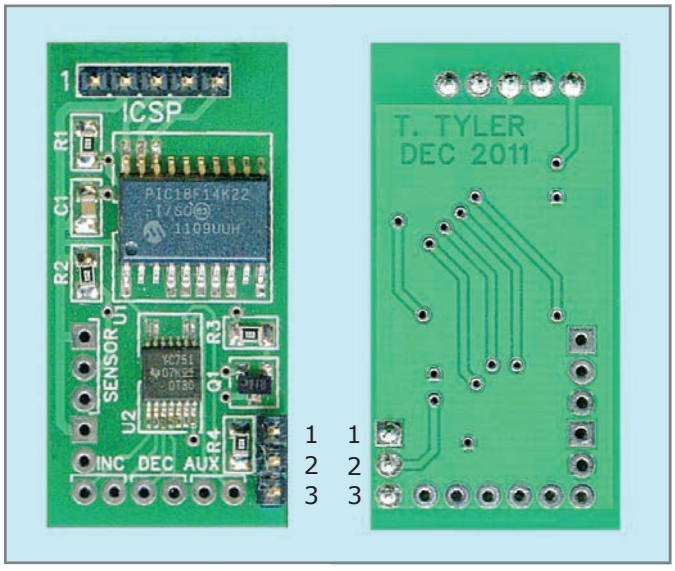


Photo 2—The thermometer circuit board assembly. The five-pin header is a direct plug-in for a Microchip PICkit2 programmer. The three-pin header is the diagnostic serial output.

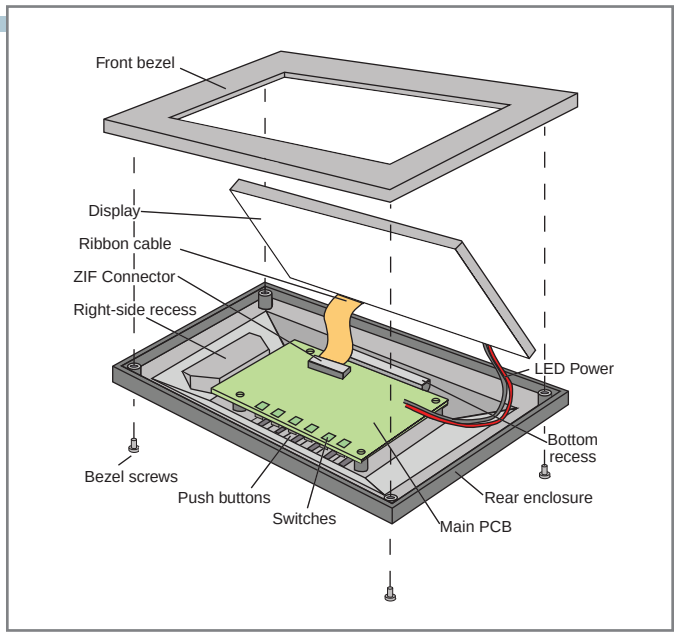


Figure 2—The Colby DP700 photo frame's basic components

Photo 2 shows the thermometer circuit board assembly. The five-pin header is a direct plug-in for a PICkit2 programmer. The three-pin header is the diagnostic serial output. To use that, you'll need a simple three-conductor communication cable terminated with a female DE9 connector. Connect Pin 1 to DE9-4, Pin 2 to DE9-2, and Pin 3 to DE9-5. You can also omit both headers, Q1, R3, and R4 from the assembly if you don't use them. They are not essential to the operation of the thermometer.

WHAT'S UNDER THE HOOD?

I used a Coby DP700 photo frame as an example for the project because it is widely available, easy to modify, and has excellent quality for a low price. Figure 2 shows the basic components of this frame. Many other models in the 5-to-8"-size range have similar design and construction. If you have a different frame and want to use it, you should have little trouble adapting these instructions.

To simplify manufacturing, all the components of a frame are soldered to the main circuit board, including power and USB connectors, the SD card connector, and control switches. The front bezel is secured by four screws at the corners and snap latches at the center of each edge. Remove the screws and pry along the edges to snap the bezel loose. In many designs, the display rests loosely in the rear enclosure, with a little foam tape for cushioning. But in this Coby model, it's held firmly in the bezel with snap latches. Figure 2 shows it separated (mainly for illustration).

With either method of mounting the display, you can lift it from the main board, but you can't remove it completely because it's restrained by wiring. There are two power wires on one side for the LED backlight, and a paper-thin ribbon cable on the other side that carries all the display signals. The ribbon cable can sometimes be unplugged from its zero insertion force (ZIF) connector, but Coby locks the connector-release levers closed with cement. If you damage that ribbon cable or connector trying to unplug it, you can throw away the frame. This article describes how to modify the frame without removing the display.

The ribbon cable is long enough to enable the display to swing open about 90°, but not much more. That makes it awkward to hold it open while making wiring connections, unless you have more hands than I do. One solution is to use a holding fixture made from a scrap of lumber to protect the ribbon cable from stress or damage during modification and testing.

Cut a piece of ordinary 1" × 4" pine board exactly 7.5" long. Chamfer opposite ends of the board at the bottom on one side, and cut a notch in the center of that edge (see Photo 3a). Loosen the bezel and slide it up just far enough so that you can insert the board into the rear enclosure near the bottom, below the lower edge of the bezel (see Photo 3b). The board's chamfered corners should clear the inside radius of the rear enclosure. Temporarily tape the bezel and rear enclosure together while you fasten the board in place with two of the four bezel screws. Leave the board installed until you have completed the entire project, including all testing.

When you need to access the main circuit board to solder wires and install the PIC board, swing the bezel and display perpendicular to the rear enclosure like an open book and secure it firmly to the fixture board with masking tape (see Photo 4a). Later, during set up and testing when you need to see the screen, swing the bezel and display back down and secure them to the rear enclosure with masking tape (see Photo 4b).

MECHANICAL MODIFICATIONS

The only mechanical modification is adding a 3.5-mm stereo jack to connect the remote temperature sensor. You may be able to drill a 0.25" hole in the frame and attach the jack with its knurled ring nut. But sometimes the stereo plug sticks out in a way that spoils the appearance of the frame or interferes with mounting it on a wall. Here's a way to install the jack that keeps it and the sensor cable flat against the

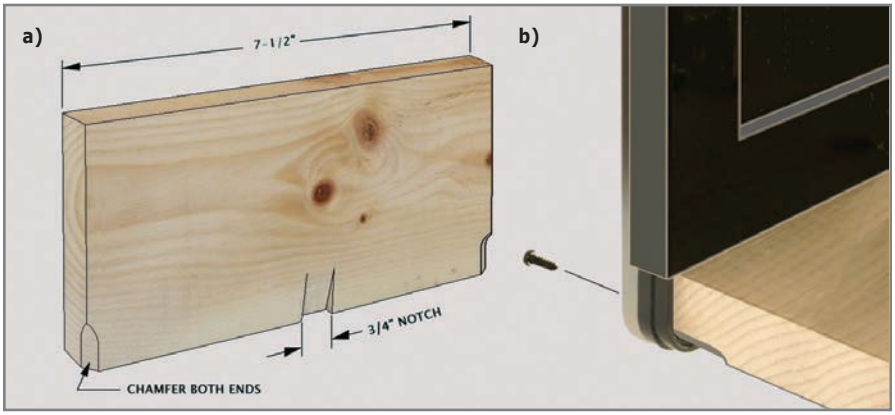


Photo 3a—The lower edge of a pine board is notched and chamfered. **b—**The board is attached to the rear enclosure near the bottom, below the lower edge of the bezel.

rear of the frame and out of sight.

Cut a piece of perforated project board 0.6" × 0.7" and enlarge the three to five holes that line up with the terminals on the side of the jack with a 3/32" drill (see Figure 3). The perforated board acts as a spacer for the stereo plug when cemented to the enclosure.

Before attaching anything to the perforated board, use it as a guide to drill matching terminal holes through the rear enclosure. Select a position low and to the right in the recessed area so it clears the power connector but does not extend below the lower edge of the rear enclosure (see Photo 5).

Discard the knurled ring nut, and solder wires about 10" long to the three main terminals of the jack. Use the smallest-gauge insulated wire you can find, and keep the terminal solder joints a minimum size. Feed the wires through the perfboard and use a glue gun or epoxy to cement the jack to the board. After the cement has set, feed the wires through their corresponding holes in the rear enclosure, past the lower edge of the pine board and into the frame. Use a glue gun or epoxy to cement the perfboard and jack assembly to the outside of the enclosure.

FINAL WIRING

Referring to the wiring diagram in Photo 6, first prep the main PCB by attaching six insulated wires about 8" to 10" long, one wire to 3.3 V, one wire to ground, two wires to SW4, and two wires to SW3. In the Coby frame, all the switches are lined up along the edge of the board as shown in Photo 7. Terminals with an oval drawn around them can be interchangeably used for wiring. Try to keep solder joints within the area of the switch terminal solder pads to minimize the risk of shorts to the ground plane. Mark the wires or use different colors to identify the INC and DEC pairs.

Solder all nine wires to the PIC board—six from the main PCB and three from the stereo jack. Trim the excess wire length so the PIC board will lie easily in the empty space beside the main PCB. Route the wires so they won't get pinched when the bezel and display are replaced. Use masking tape to hold everything in place and keep the PIC board from shorting out.

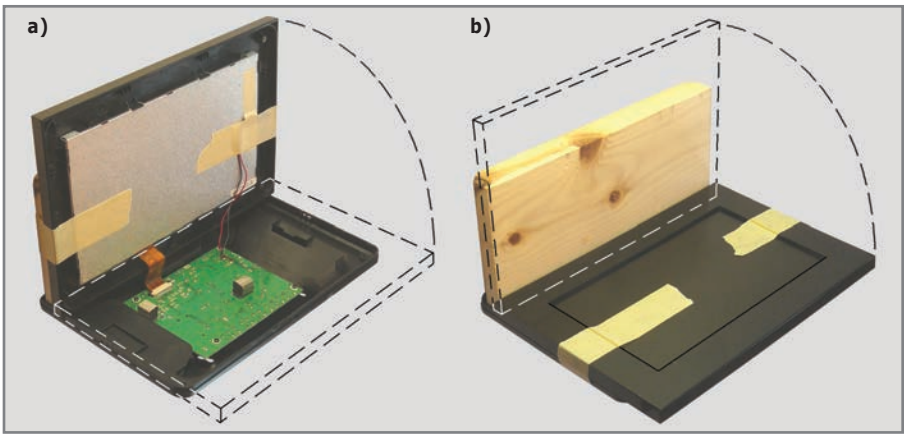


Photo 4a—The bezel and display are firmly secured to the fixture board with masking tape. **b—**During setup and testing the bezel and display can be swung down and secured to the rear enclosure with masking tape.

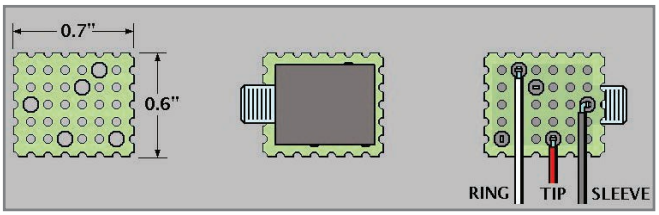


Figure 3—The perforated board spaces the jack away from the rear enclosure to clear the stereo plug.

THE WEATHER-PROOF SENSOR

The Microchip DS18S20 digital temperature sensor is a three-lead package the same size as a TO-92 transistor (see Figure 4). If you keep solder joints to the leads tight and neat, the finished

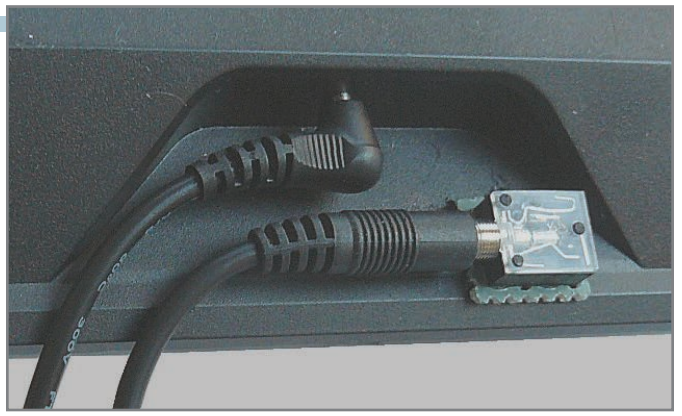


Photo 5—Use the perforated board as a drilling guide

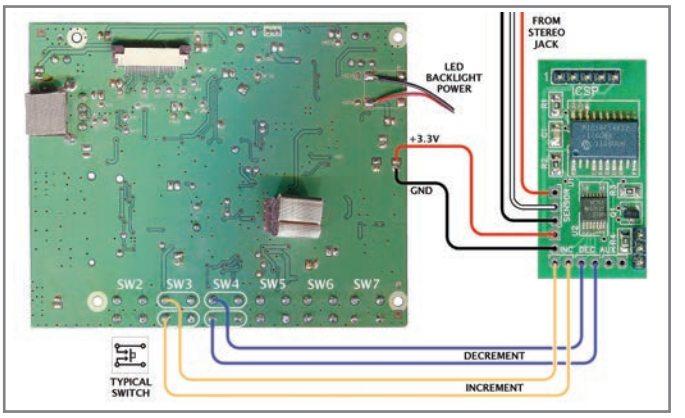


Photo 6—Wiring diagram

probe will be a small cylinder about 0.25" in diameter. It can be fished through a small hole in an external wall or run through a closed window with a compressible weatherstrip gasket. You can use practically any three-conductor cable for the sensor. A 25' stereo cable is small, tough, shielded, and inexpensive.

Insulating short spliced leads with sleeving is always a problem because the sleeving gets in the way of soldering. One way to keep the probe small and strong is to drip a little fast-set epoxy on the soldered leads, after ensuring they aren't touching, and rotate the unit slowly for a couple of minutes until the epoxy stops running and begins to harden. Weatherproof the entire assembly with an inch or so of 0.25" heat-shrink tubing.

LOADING IMAGES INTO MEMORY

Some photo frames don't have internal memory, so I used a plug-in SD memory card for the temperature images. That also makes it easy to change the appearance of the display whenever you want. Any capacity card you can find is more than adequate, since the images average only about 25 KB each and 141 of them is less than 5 MB. A good source for generic 32-MB SD cards is OEMPCWorld. Their SD cards cost less than $4 each, including free shipping via U.S. Postal Service first-class mail. Just search their site for "32-MB SD card."

A download package is available with images in 16 × 9 format showing temperature over the range from −20 to 120°F in numerals about 2" high. The 16 × 9 images will naturally fit the Coby screen and most other brands. There's also a set of 4 × 3 images for frames with that format. Actually, either size will work in any frame. If you use 4 × 3 images in the 16 × 9 Coby with Show Type set up as Fit Screen, there will be bars on the sides. But if it is set up as Full Screen, the images will expand to eliminate the bars, and the numerals will be about 2-5/8" high.

The download filenames have a sequential numeric prefix from 100 to 240, so Windows will list them in order before you copy them to the SD card. Notice that the sequence of images is as follows: 70°, 71°, 72°…119°, 120°, −20°, −19°, −18°…−2°, −1°, 0°, 1°, 2°…67°, 68°, 69°. The first image is *not* the lowest temperature. That's so synchronization can start from 70° instead of all the way from −20°. You can split the temperature range like this as long as there are no extraneous pictures on the SD card, because the frame treats the SD card, in effect, as an endless circular memory, wrapping around from the highest to lowest image when incrementing, and from lowest to highest when decrementing.

If you don't have an SD card reader you'll find plenty of USB models at DealExtreme. Search for "SD card reader." Delivery is slow (usually three weeks) but the price is right, just $2 or $3 with free shipping. When you plug the card and reader into a computer it will appear as a drive (e.g., drive F). Erase any files in the card so it is completely blank. Open the folder containing your image files and make sure they are listed in ascending filename order, from 100_70.jpg through 240_69.jpg. Select (highlight) all 141 files and drag and drop them into the SD card directory, ensuring that you drag them with the file named 100_70.jpg at the top of the group.

Note: if you drag and drop using *any* file other than 100_70.jpg at the top of the list, all the files will be loaded into the SD card, but not in proper sequence. What's worse, you won't be able to tell by looking at the SD card directory. Your first clue that you have made this error will be the initial temperature display on power-up is not 70°F. The thermometer won't display the correct temperature until the SD card is correctly cleared and reloaded.

SETUP & FINAL TEST

It's always best to make sure frame power is disconnected before plugging or unplugging the temperature sensor. Position the frame so that the screen is visible. Plug in the sensor and SD card, then connect power to the frame. After a few seconds, what you see on the screen will depend on how the frame was last used and set up. It may start showing pictures from internal memory, or it may start showing temperature images from the SD card. In either case, the pictures will probably start changing rapidly for a while because the frame thinks it is synchronizing its initial display to the temperature of the sensor. You can't use on-screen menus to check the setup of the frame while it is flipping through all those pictures, so you must wait. After a

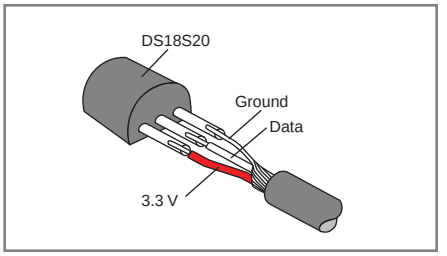


Figure 4—The Microchip DS18S20 digital temperature sensor

couple minutes, when things settle down and the display stops rapidly changing, press Menu to bring up the main menu. Use the left or right arrow buttons to select the Set Up sub menu, then use the Enter, Left, Right, Up, and Down buttons to set up the following parameters: Interval Time = Off, Transition Effect = No Effect, Show Type = Fit Screen, Magic Slideshow = Off.

This terminology is used by the Coby DP700 frame, but other brands use similar terms. The most critical setting is the one that disables the interval timer so that the PIC has complete control over incrementing and decrementing the displayed image.

After completing all the setup adjustments, momentarily disconnect power from the frame and confirm that it properly powers up. The Coby logo should appear for a few seconds, followed by the first image in memory, the starting temperature of 70°F. About 12 s later, the display should start changing in 1° steps until it gets to the current temperature of the sensor. Warm the sensor with your hand to ensure the sensor is responding.

This is a good time to demonstrate an error indicator designed into the thermometer to alert you if the PIC can't communicate with the temperature sensor. Disconnect power and unplug the sensor, then restore power with the sensor disconnected. The display will start at 70°F as before, but this time it will keep incrementing until it reaches 99°F, where it will stop. So if you ever notice the display stuck on 99° when you know it's not that hot outside, check to see if the sensor is unplugged or damaged.

If everything seems to be working properly, you can skip the following section on troubleshooting. Close the frame and start thinking about how and where you will install it.

TROUBLESHOOTING

In case of serious problems, there is a diagnostic feature built into the thermometer control board that enables you to monitor exactly what is happening. The PIC processor transmits a serial message every time it polls the temperature sensor. These messages can be received and displayed through a legacy serial port on your PC or through a USB port if you have a USB-to-serial converter. Almost any communication utility program that will monitor RS-232 serial communications can be used (e.g., ComTest or RealTerm). I'll use Windows' own built-in HyperTerminal for an example, since almost everyone has access to it. You'll find it in the Windows NT folder under Program Files. After connecting your three-wire communication cable, start the program and configure the port for 38,400 bps, 8 data bits, no parity, 1 stop bit, and no flow control. When the frame powers up, you'll see the window begin to fill with data (see Photo 7).

At first, all you'll see is a message telling you the frame is initializing, and below that a number that starts at 12 and counts down to 1. That's a start-up delay to give the photo frame time to initialize and internally load some files. Then the processor will search for a temperature sensor. If it finds one, it prints three lines of information, including its unique serial number, and starts polling approximately once per second.

Each message gives the current temperature reported by the sensor in Centigrade, its Fahrenheit equivalent, and the integer value of Fahrenheit temperature, referred to as Target. Also shown is the currently displayed temperature (referred to as Display). The far right column shows what action will be taken to bring the display 1° closer to the Target, either Increment, Decrement, or None. The program updates the Target whenever it differs from the reported temperature by more than 0.75°F. That adds a small amount of hysteresis to prevent the temperature image from excessively fluctuating.

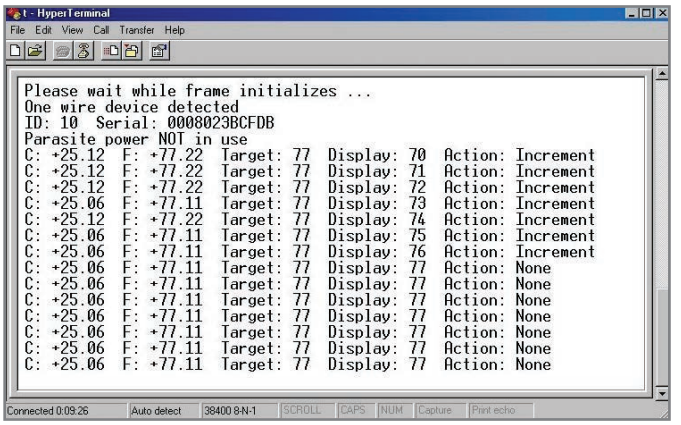


Photo 7—When the frame powers-up, you'll see the window begin to accumulate data

The serial communication display is an excellent tool for checking the accuracy of the thermometer because you can confirm that the displayed temperature agrees with the precise measurement reported by the sensor.

ABOUT THE FIRMWARE

Credit for design of the PIC firmware goes to Kevin R. Timmerman—a talented freelance software design engineer, and owner of the Compendium Arcana website—who collaborated with me on this project. Kevin's backyard in Michigan, as well as mine in Colorado, were the beta-test sites for the design.

A firmware download includes the temperature.hex file needed for programming the PIC, as well as the following source files in case you want to make changes:

```
inverted_main.c
one_wire.c
fuses_14k22.c
one_wire.h

stdint.h
```

The file named one_wire.c deals exclusively with sending and receiving messages to/from the temperature sensor. If you use a photo frame other than the Coby DP700 that has some special requirements, the only file you might need to modify is inverted_main.c. I'll point out some of the key areas.

Line 58 sets the time permitted for the frame to start Slideshow mode after power up. The value of 12 s was originally selected to accommodate a SmartParts photo frame. The Coby is a few seconds faster, but there's no advantage in tweaking this value.

Line 59 sets 250 ms as the duration of contact closure for the solid-state relay switches that increment and decrement the display. If this time is too short, the frame can occasionally skip a step. If it is too long, the frame can occasionally take more than one step at a time. The optimum actuation time may depend partly on the frame's firmware and the debounce routine it uses to discriminate against the mechanical switches' contact bounce. If you have to experiment with this parameter for an unknown frame, keep in mind that some frames initiate action on switch make, others (such as the Coby DP700) on break. A good way to check whether a thermometer steps smoothly through the images is to power up with the sensor unplugged and watch it increment from 70°F to 99°F. If it stops on any value other than 99°F you may be missing or adding steps.

Line 60 tells the PIC that the first image in memory, the start-up display image, is 70°F. The "DEC" routine at line 71 is called to decrement the displayed image, and the "INC" routine at line 79 is called to increment it.

The main loop of the program begins at line 191. First the temperature sensor is polled. A 1,000-ms delay at line 201 gives the sensor time to take the reading and report it. Then the sensor data is converted and processed. If there's no response from the sensor, the reported temperature defaults to 99°F at line 229 to provide the sensor fault indication described earlier. At line 234, the PIC decides if the displayed image needs to be incremented or decremented. After transmitting the serial diagnostic message at line 240, the PIC takes any action necessary to change the display at line 247, completing the loop.

The temperature display can be incremented or decremented by 1° each time around the main loop. Loop time varies from 1 s to 1.25 s, depending on what's going on. Even if you create complex images for your thermometer, the frame will have no trouble keeping up with gradual temperature changes. But if loading the complex pictures causes the frame to struggle during synchronization, just increase the delay at line 201 to slow down loop speed. It's not essential that a thermometer be able to update itself every second.

UNLIMITED OPTIONS

When you finish the project, you will have the satisfaction of knowing you probably have the most accurate thermometer in the neighborhood—providing you take reasonable precautions in locating the sensor. Don't place it in sunlight or near heat sources (i.e., vents or ducts). Even placing it too close to a poorly insulated wall, roof, or window can affect its accuracy. There are articles online about the best places to install outdoor thermometers.

Even after you have completed your modifications to the frame and closed it back up, there are endless ways to customize the project to your taste. And the possibilities go beyond superficial things like painting or decorating the frame. Creating your own set of temperature images is an obvious way to personalize the thermometer and make it unique. Although the work is a little repetitious because of the large number of images required, it can be fun. A suggested size for 16 × 9 images is 960 × 540 pixels. For 4 × 3 images, it is 720 × 540 pixels. With hundreds of font styles to choose from, and endless background colors, the possibilities are unlimited. Imagine a set

of images divided into six or eight groups, each with a different background color, such as "bone-chilling purple" for temperatures below zero, "freezing dark blue" for temperatures below freezing, followed by green, yellow, and increasingly warmer colors all the way to "scorching-hot red" for temperatures above 100°F.

To give you a running start on experimenting with colors, the download includes a set of temperature images in bitmap format. Most frames won't display these if loaded into the SD card as is, but you can modify and save them as JPEG images before using them. And, you don't need an elaborate graphic arts program to do that. You can do a lot with the venerable old Paint program included in most Windows systems. First open one of the bitmap image files with mspaint.exe, and then change the colors of the characters and/or background using the Fill with Color tool. When you are satisfied with the appearance, save it as a JPEG file. Repeat those steps for all the images you want to change. Does this sound like a good way for your kids to earn a little extra spending money? You could make image sets using suitable background colors for each season. Or you could create an image set in a theme motif to celebrate a holiday, birthday, or special event.

To make new or additional characters, experiment with the versatile Text tool included in Paint. And don't be timid about character size when you see a tiny text box with a small font after clicking on a location in the picture. Drop down the font size selection box, ignore its limited offering of sizes 8 through 72, and enter a size of 300. Click once to delete the small text box, then click again to get the 300-size box. Type in a couple of digits and see how enormous they are! Then, be prepared to spend a few hours after you get hooked on sampling and playing with exotic type fonts. Want to make a patriotic thermometer? Start with an image using the font called QuickType II Pi with the bold option selected. Color the digit outlines red, leave the core of the digits white, and color the background blue. Of course the job is easier and the special effects are more abundant if you have a graphics program. All the images in the download packages were made with an old PaintShopPro Version 5 that I found on eBay eight years ago for about $10.

For those living overseas or accustomed to expressing temperature in Centigrade, the download includes an alternate set of images covering the range from −28.9°C to 48.9°C. Images such as 70°F, 71°F, 72°F, and so forth are replaced with their Centigrade equivalents 21.1°C, 21.7°C, 22.2°C, and so forth. The thermometer control can't tell the difference. It goes on incrementing and decrementing images as if it were displaying the temperature in Fahrenheit. By showing temperature in tenths of Centigrade degrees, the thermometer accuracy is unchanged. The temperature sensor is inherently a Centigrade device, and one could modify the PIC firmware to use the reported temperature in degrees C without ever converting it to degrees Fahrenheit. But this method is a lot easier, and enables you to change between Centigrade and Fahrenheit by just swapping the SD card.

If you make changes to accommodate a different photo frame, or if you design temperature images you are proud of and think are particularly unique, you are invited to share them with other readers via an e-mail to the editor.

Tommy Tyler (tomytyler@comcast.net) graduated with honors from Vanderbilt University with a degree in Mechanical Engineering. He retired after a career spanning more than 40 years managing the product design of industrial instrumentation, medical electronics, consumer electronics, and embedded robotic material transport systems. Tommy earned 17 patents from 1960 to 1995. His current hobbies are electronics, technical writing and illustration, and music. Tommy is a contributing expert to the JP1 Forum on infrared remote control technology.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/264.

RESOURCES

Compendium Arcana, www.compendiumarcana.com.

DealExtreme, www.dealextreme.com.

OEMPCWorld, www.oempcworld.com.

SOURCES

DP700 Digital photo frame
Coby Electronics Corp. | www.cobyusa.com

PIC18F14K22 Microprocessor, DS18S20 digital temperature sensor, and PICkit2 development programmer/debugger
Microchip Technology, Inc. | www.microchip.com

TS3A4751 quad SPST Analog switch
Texas Instruments, Inc. | www.ti.com

NEED-TO-KNOW INFO

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

Audio-Enhanced Touch Sensors
by Matt Oppenheim
***Circuit Cellar* 262, 2012**

Using a touch sensor, you can trigger audio tags on electronic devices. Here you learn how to couple capacitive-touch sensors with an Android device. Topics: Capacitive-Touch Sensors, PCB, IOIO Board

Cost-Effective Mobile Data Storage
Interface an SD Memory Card with an MCU
by Mandar Bagul
***Circuit Cellar* 221, 2008**

This article explains how to expand a microcontroller's nonvolatile data storage capacity with an SD card. It's an intelligent way to handle the excessive memory requirements of typical embedded applications. Topics: SD Card, Flash Memory, WinAVR

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

by Patrick Schaumont (USA)

Electronic Signatures for Firmware Updates

Embedded systems benefit from signatures. An electronic signature can verify the origin of a firmware binary, a feature especially useful for firmware that needs to be updated in the field. This article details how electronic signatures are constructed and how you can use them to sign firmware updates.

Suppose you are selling a microcontroller board that needs occasional firmware updates. Can you write the firmware-updating software so it will test whether or not the updates are genuine? You can do it using electronic signatures, which are sequences of bytes that uniquely link a message (the firmware) to an individual (you) or an entity (your company). Electronic signatures are already used in mainstream operating systems. They are used to verify the authenticity of software downloads.

In this article, I demonstrate how that concept can also be applied to embedded systems firmware. The challenge, in this case, is to use a signing system that is secure—yet efficient and lean—in line with the capabilities of a small embedded processor. This article shows how this can be achieved.

Let me begin by clarifying what signatures can and cannot provide. A firmware signature enables you to trace back the firmware binary to a specific origin and, if needed, even to a specific development system. By verifying the signature of a firmware update, your embedded system knows if the firmware update comes from a trusted developer. A signature does not protect your firmware from mistakes. A bug remains a bug, even when it's signed! Also, signing your firmware will not protect its confidentiality. If you are concerned about protecting intellectual property, you need firmware encryption on top of firmware signatures. In this article, I only cover signatures.

Figure 1 illustrates the objective. An embedded development system is used to write, compile, and debug code for an embedded microcontroller using a standard compiler toolchain. When the executable is ready, it is signed by combining the binary image with a private key. As its name implies, the private key is known only to the developer and no one else. The resulting signature

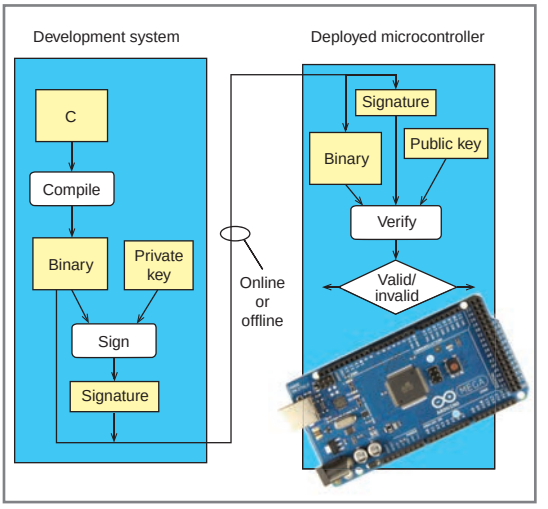


Figure 1—On the development system, electronic signatures are created by signing an executable with a secret key. On the deployed microcontroller, the executable and a public key are used to verify the signature. Upon successful signature verification, the firmware update is accepted.

is a string of bytes that is typically 32 bytes or more. The signature has a particular relationship to the executable. It can only be generated by someone with private key knowledge. Furthermore, a small change to the executable will also yield a different signature. The executable and the signature are then distributed to the deployed microcontroller. At the microcontroller, the executable can be installed and executed. However, before installation, the signature that comes with the firmware update is verified. This requires three elements of information: the executable image, the signature, and a public key. The public key is the counterpart of the private key in the chosen signature scheme. In contrast to the private key, the public key does not have to be kept secret. Anyone with knowledge of the public key can use it to verify the validity of a given signature.

Thus, a signature scheme needs two different keys: a private one to create signatures and a public one to verify them. The scheme cannot work with two identical keys. If the signing key is not private to the signer, the signature can be forged! The cryptographic procedures to create and verify signatures, therefore, rely on asymmetric-key or public-key cryptography, a fascinating subfield of information security.

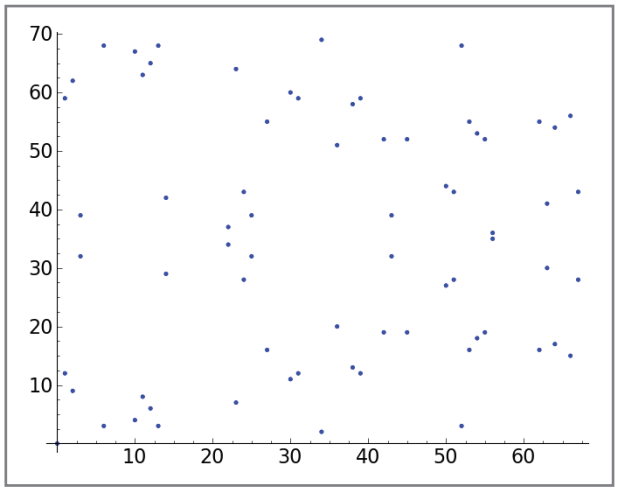


Figure 2—Solutions to the elliptic curve $y^2 + x^3 = x$ over a size 71 finite field

ECC

Public-key cryptography enables you to encrypt and decrypt messages with two different keys. These two related keys are called a key pair. One key is public and the other is private. A key pair belongs to a single individual. The public key is published for everyone to see, while the private key is kept secret. In the past three decades, several public-key cryptographic systems have been described and developed. Perhaps the best known one is RSA, named after its three inventors R. Rivest, A. Shamir, and L. Adleman. All public-key schemes presently in use have a high computational complexity. A rule of thumb is that it takes three orders of magnitude more computations to encrypt a bit using a public-key system compared to the same encryption using a classic symmetric-key system. Therefore, public-key cryptography is used only in cases that cannot be supported using other techniques; and signing is one of them.

The RSA cryptosystem needs large key sizes to be secure—on the order of 1,024 bits per key or higher. Obviously, this can be an issue for constrained, embedded implementation. An alternative public-key system that enables shorter key sizes is elliptic-curve cryptography (ECC). First proposed in the 1980s, ECC has now become established and is supported through several standards (e.g., IEEE 1363-2000, FIPS-186, and ANS X9.62-2005).

The basic principles of ECC are both interesting and remarkable. In this article, I only summarize the fundamental ideas. The Resources section of this article includes a few pointers to in-depth descriptions. The underlying structure used by ECC is an elliptic curve—a function of the form $y^2 = x^3 + ax + b$. The curve is drawn on a finite field, which means any solution (x, y) to $y^2 = x^3 + ax + b$ is made up as elements of a finite field. Figure 2 shows an elliptic curve drawn over a size 71 field. Clearly, a curve in ECC is not a smooth shape, but is a discrete, seemingly random set of points. Cryptographers have found a way to count the points of such a curve as a group. Using an operation called point addition, any two points on the curve can be added to yield a third point, also lying on the curve. In the curve shown in Figure 2, any two points can be added to yield one of the other points shown. The point addition is a well-defined but complex relation and the result of adding two points is hard to predict. Adding a point n times to itself is called the point multiplication: Q = P + P + .. + P = n. P.

In this formula, Q and P are two points of the curve and n is a positive integer. Because of the complex behavior of point addition, the point multiplication behaves like a one-way operation. If you know P and n, it's straightforward to calculate Q. But if you only know P and Q, it's very hard to determine n. This type of one-way operation is precisely what is needed for public-key cryptography. Cryptographers use this property to construct a signature protocol.

The elliptic curves used for public-key cryptography are much larger than the one shown in Figure 2. In this article, for example, I use a standard curve called NIST K-163. This curve is drawn over a 163-bit field, meaning that the x and y coordinates of every point are 163-bit numbers. The point multiplication uses a 163-bit value of n. The calculation of such a point multiplication takes a significant amount of computation. It requires about 250 modular multiplications, each of them multiplying two 163-bit operands. Getting

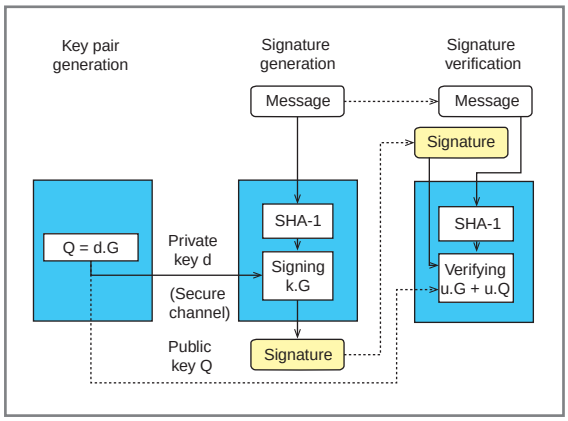


Figure 3—The elliptic curve digital signature algorithm (ECDSA) defines procedures for key pair generation, signature generation, and signature verification. Generating a key pair and generating a signature each require a single-point multiplication, while verifying a signature requires two.

this done efficiently on a microcontroller is challenging, and a good deal of the effort in cryptographic engineering is optimizing such complex operations.

The point multiplication in ECC is the basis of cryptographic operations, and signing and verifying signatures is one of them. The standard I'll be using is the elliptic curve digital signature algorithm (ECDSA).

ECDSA

A digital signature standard needs at least three procedures. It needs to specify how to generate a key pair, how to sign a message, and how the signature is verified. Figure 3 shows an overview of the steps involved in ECDSA. The figure leaves out the details of the math, but illustrates the most important operations in ECDSA. Generating a key pair requires a single-point multiplication. For a given, agreed-upon base point G, the private key is a secret scalar d, and the public key is the result of multiplying G and d. In the NIST K-163 curve used in this article, d is a 163-bit number, while Q consists of 163-bit coordinates x and y.

The signature-generation procedure uses the private key d and an input message to create a signature. Signature generation costs a single-point multiplication. Rather than directly signing the message, ECDSA produces a signature starting from a SHA-1 digest of the message. You may recall SHA-1 from my May 2012 article, "One-Time Passwords from Your Watch," (*Circuit Cellar* 262). It's a hash algorithm that converts an arbitrary-length message to a fixed-length, 160-bit digest. The signature-verification procedure combines the public key, the signature, and the message into a test that demonstrates whether or not a signature is valid. Signature verification costs two point multiplications. Just as with signature generation, the verification is done on the digest of the message, rather than on the message itself. I'll now use ECDSA to build a firmware-update verification system for an Atmel ATmega2560 microcontroller.

ATmega2560 & RELIC

The ATmega2560 microcontroller is based on the popular AVR series of processors. The 16-bit processor comes with 256-KB on-chip flash memory, 8 KB of RAM, and 4 KB of EEPROM. There are several factors that make this microcontroller an interesting case to demonstrate digital firmware signatures.

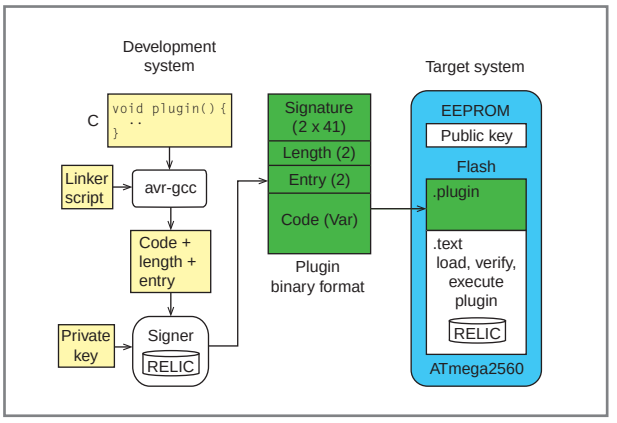


Figure 4—Design flow for firmware plugins. On the development system, a C program is compiled, statically linked, and signed. The binary format for the plugin starts with the signature (82 bytes in length), followed by a length field (2 bytes), an entry point (2 bytes), and a variable-length section of code. On the target system, the plugin is loaded at address 0x1000. The public key, stored in EEPROM, is used to verify the signature and execute the plugin code.

The first factor is that this microprocessor either runs a lightweight operating system or doesn't run an operating system at all. I will be developing the prototype in this article as a bare-metal program, so that it does not require any operating system support.

The second factor is that the ATmega2560 is a lightweight processor with limited computational power. I am using a 16-MHz implementation on an Arduino Mega 2560 board. Limited computational power is an important issue because of the complexity of a signature verification algorithm such as ECDSA. Although signature-verification needs to be done only upon a firmware update, it may still add a considerable start-up delay.

The third factor is that the ATmega2560 has a Harvard architecture. It uses a separate memory space for program and data. However, when calculating a signature for firmware, the ATmega2560 needs to access a program as data. Specialized library support is required to read opcodes from program space and perform computations on them.

Rather than writing an implementation for ECDSA from scratch, I use an open-source public-key cryptographic library. The RELIC library, developed by D. Aranha and C. Gouvea at the University of Brazil, is a highly optimized and flexible library for cryptography. Of particular interest for this project is that the library has support for AVR. Furthermore, it can be easily migrated between AVR, X86, ARM, and MSP430 platforms. This is an important advantage. Point multiplications use long word-length operands and modular arithmetic, neither of which are natively supported by a microprocessor's instruction set. With the portability of RELIC, I can develop the initial prototype of the signing system on a PC and then cross-compile it for the AVR with almost no source code modifications. Furthermore, I can also generate firmware signatures on an X86 system and verify them on an AVR target.

FIRMWARE PLUGIN

For the chosen target platform, I need to build a design flow to create, sign, and deploy firmware updates for the ATmega2560. Because there is no operating system, the runtime loading and linking of a firmware update on the target processor must be simple. I took a minimalistic approach. I made the assumption that firmware updates are sections of statically linked code in a fixed portion of the flash memory on the ATmega2560 microcontroller. I also assumed that

Listing 1—This example illustrates the firmware signing concept.

```
#include "relic.h"
void genkey() {
  ec_t q;                 // public key
  bn_t d;                 // private key
  cp_ecdsa_gen(d, q);     // generate key pair

  printf("Private Key\n");
  bn_print(d);
  printf("\n");

  printf("Public Key\n");
  ec_print(q);
  printf("\n");
}
```

Address	Function
Private (n)	291FAEC3D1F26F058D00AC6565762C4352DD1FAC
Public (Qx)	036B2DA51E923541B1CE967A16F074248536716778
(Qy)	02E1827FA49665A4ED24FB0A03C9DBFFC4FF1F663B

Table 1—Key pair used in this example

firmware updates are loaded into a contiguous memory segment starting at address 0x1000 and that they will be less than 64 KB long. Each firmware update can have a different length and a different entry point, but they should be fully self-contained. There is no runtime relocation or linking done on the target system. With these assumptions, the name "firmware plugin" may be more appropriate than firmware update, so I will be using this term.

Figure 4 demonstrates the design flow for the plugin. A plugin starts as a C program that is compiled using the AVR cross compiler. The plugin has no `main()` function, but it has a function `plugin()`that serves an equivalent purpose. The plugin has a length and an entry point. The length is defined by the amount of C code in the plugin, while the entry point is defined by the starting address of the `plugin()` function. The plugin is compiled and linked to start at a fixed address 0x1000. This can be achieved with a linker script for the cross compiler. My linker script locates the start of the code segment `(.text)` at address 0x1056. The offset of 0x56 bytes (84 decimal) keeps room for the signature in the plugin image. Once the plugin is compiled and linked, I run a signing program that creates the ECDSA signature and produces the resulting plugin binary.

In the target system, the public key needed to verify the signature is loaded in EEPROM. The choice to install the public key in EEPROM is a matter of convenience. It enables preinstallation of a public key into the target system before the actual firmware is loaded. Conceptually, different firmware developers or companies could use different public keys. Once a public key from a given firmware developer is available on the system, firmware updates from that developer could be accepted. My design only uses a single public key. To install the plugin binary on the target system, the plugin binary has to be loaded at address 0x1000 in the target system. The specific loading technique depends on the memory configuration and the features of the

Listing 2—A tiny program serving as the test plugin

```
1. #include <avr/io.h>
2. #include <util/delay.h>
3.
4. void slow(unsigned a) {
5.    _delay_ms(a);
6. }
7.
8. void plugin() {
9.    DDRB |= _BV(DDB7);
10.   while (1) {
11.     PORTB |= _BV(PORTB7);
12.     slow(25);
13.     PORTB &= ~_BV(PORTB7);
14.     slow(300);
15.   }
16.}
```

target platform. In my concept demonstrator, I represent a plugin binary as an array of bytes in C, which can be compiled together with the plugin installer. When the plugin installer executes, it will read the public key that verifies the plugin from EEPROM. If the signature checks out correctly, the target system can call `plugin()`.

FIRMWARE SIGNING

Listing 1 is a small example that illustrates the firmware signing concept. The complete set of source code files is available on *Circuit Cellar*'s FTP site. I start by configuring RELIC for the NIST K-163 curve, and compiling it for X86 as well as AVR. I also generate a key pair with a private and a public key. RELIC provides a single function that can generate such a key pair, starting from an internally generated random number.

The resulting key pair is shown as hex strings in Table 1. These will be used to sign and verify the firmware plugin. Listing 2 shows a tiny program that will serve as test plugin. It blinks the light on an Arduino Mega 2560 board with a 100-ms period. To create a signature, first compile and link the program as shown in Figure 5. This a standard compilation sequence for ATmega2560 code, with the exception of the linker command. The `–nostartfiles` flag prevents inclusion of standard startup files. It enables you to create a linked binary from C with no main function. The `–Wl,–T avr6.custom` flag selects a custom linker script, which allocates the start of the code segment (`.text`) at address 0x1056. The `avr-objcopy` program converts the ELF format of the executable to a memory-image binary format.

```
avr-gcc -g -v -Os -DF_CPU=16000000UL \
    -mmcu=atmega2560 -c -o plugin.o plugin.c
avr-gcc -g -v -nostartfiles -mmcu=atmega2560 plugin.o \
    -Wl,-Map=plugin.map -Wl,-T avr6.custom -o plugin
avr-objcopy -O binary -R .eeprom plugin plugin.bin
```

Figure 5—Compile and link this program to create a signature.

Inspection of the linker map file (`plugin.map`) shows the location of functions in the executable image, and Table 2 shows a partial list of functions. Besides the `slow()` and `plugin()` functions from Listing 1, the compiler also inserted additional library support functions. The map file also indicates the entry point for the plugin: 0x10fc, the start address of `plugin()`. Finally, my custom linker script emits a symbol `_eplugin` at the end of the code segment. This way, I can determine the size of the plugin code segment from the map file. The size is 0x1dc6 — 0x1056, or 3440 bytes.

Next, the plugin needs to be signed. I used the RELIC library to write a signing

Address	Function
0x1056	slow
0x10fc	plugin
0x1114	__fixunssfsi
0x1404	__subsf3
0x1466	__addsf3
0x14c0	__mulsf3
…	…
0x1d58	__prologue_saves__
0x1d90	__epilogue_restores__
0x1dc6	_eplugin

Table 2—Symbols in the example plugin binary

Listing 3—A sample signature-verification program

```
1. extern __attribute__((__section__(".plugin")))
2.        unsigned char plugin[];
3.
4. unsigned verifysignature() {
5.   ec_t q;      // public key
6.   bn_t r, s;  // signature
7.   char c[64];
8.   unsigned chk, len, ofs, i;
9.
10. core_init();
11. ec_param_set_any();
12. ec_new(q);
13. bn_new(r);
14. bn_new(s);
15.
16. // read public key from EEPROM
17. for (chk=0; chk<42; chk++)
18.     c[chk] = eeprom_read_byte(chk);
19. fb_read(q->x, c, 42, 16);
20. for (chk=0; chk<42; chk++)
21.     c[chk] = eeprom_read_byte(chk+42);
22.  fb_read(q->y, c, 42, 16);
23.  fb_read(q->z, "1", 1, 16);
24.
25. // read signature from plugin
26. for (chk=0; chk<41; chk++)
27.     c[chk] = pgm_read_byte(&(plugin[chk]));
28. bn_read_str(r, c,  41, 16);
29. for (chk=41; chk<82; chk++)
30.     c[chk-41] = pgm_read_byte(&(plugin[chk]));
31. bn_read_str(s, c, 41, 16);
32.
33. // read length and entry point from plugin
34. len =   pgm_read_byte(&plugin[82])*256 +
35.          pgm_read_byte(&plugin[83]);
36. ofs =   pgm_read_byte(&plugin[84])*256 +
37.          pgm_read_byte(&plugin[85]);
38.
39. // verify signature
40. chk = code_cp_ecdsa_ver(r, s,    // signature
41.                          (PGM_P) &(plugin[86]), // 'message'
42.                          (unsigned) len,        // message length
43.                          q);                    // public key
44. ec_null(q);
45. bn_null(r);
46. bn_null(s);
47.
48. return chk ? (unsigned) (&(plugin[86 + ofs])) : 0;
49.}
50.
51.typedef void (*pluginptr_t)();
52.
53.int main(void) {
54.  unsigned plugin;
55.  PGM_VOID_P call_p;
56.  // verify signature
57.  plugin = verifysignature();
58.  if (plugin) {
59.     // signature correct, jump to plugin entry point
60.     plugin = plugin / 2; // to word address
61.     call_p = (PGM_VOID_P) plugin;
62.     ((pluginptr_t) call_p)();
63.  } else {
64.     // fail - show slow blinking led
65.     DDRB |= _BV(DDB7);
66.     while (1) {
67.       PORTB |= _BV(PORTB7);
68.       _delay_ms(500);
69.       PORTB &= ~_BV(PORTB7);
70.       _delay_ms(500);
71.     }
72.  }
73. }
```

program that reads the compiled plugin and signs it. The signing program specifies the private key, the start of the plugin (0x1056), the end (0x1dc6), and the entry point (0x10fc):

```
signer 291FAEC3D1F26F058D00AC6565762C4352DD1FAC \
  1056 1dc6 10fc plugin.bin
```

The signing program creates an array of bytes for the signed plugin binary. The plugin can now be loaded onto the target system. Listing 3 shows a sample signature-verification program. The plugin[] contains the signed plugin binary. Using a C __attribute__, this array is allocated in a section .plugin, which is placed at address location 0x1000 in program memory with a linker script. The signature verification function, verify signature(), reads the public key from EEPROM, and the signature from plugin[] and uses the RELIC function code_cp_ecdsa_ver to verify the signature. The function was slightly modified to enable reading from program space rather than data space. If the signature is correct, the function will return the starting address of the plugin, otherwise it returns 0. The main function simply calls verifysignature, and may call the plugin if the signature is correct. The entire signature verification process on a 16-MHz ATmega2560, takes around 2 s. I did not enable the advanced optimizations of RELIC. Of course, Listing 3 is only a proof of concept; depending on your particular application for signatures, you may need to revise this integration.

PENDING RISKS

What are the remaining risks to this implementation? As far as signatures are concerned, the most obvious issue may be someone trying to change the public key to enable verification of false plugins. Cryptographers call this a man-in-the-middle attack: an attacker makes you believe a public key is genuine, while in reality the key is a fake. Changing the public key would require rewriting the EEPROM. The ATmega2560, although not specifically designed for security applications, has lock bits that can be programmed to prevent overwriting the EEPROM.

Another strategy may involve creating a signature—called a certificate—for the public key itself. Creating a certificate requires that a trusted third party signs the public key. The public key of the trusted third party itself must be easy to verify and universally available. Using certificates makes the implementation safer, but it increases the complexity of signature verification and may require online communication to retrieve the trusted third party's public key.

PRAISE FOR PUBLIC-KEY CRYPTOGRAPHY

I demonstrated the potential of digital signatures and their use in the context of verifying the origin of firmware in an embedded system. Public-key cryptography is an important tool in the support of embedded security. It is crucial to implement digital signatures, but also to other applications in authentication and privacy. Public-key cryptography can address problems that cannot be handled with classic, symmetric-key cryptography.

Patrick Schaumont is an associate professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. He works with his students on research projects in embedded security, covering hardware, firmware, and software. You may reach him at schaum@vt.edu.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/264.

RESOURCES

AVR Libc, www.nongnu.org/avr-libc/user-manual/index.html.

D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer Professional Computing Series, 2004.

National Institute for Standards and Technology, "Federal Information Processing Standards Publication 186-3: Digital Signature Standard (DSS)," 2009, http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf.

L. B. Oliveira, et al, "TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks," Computer Communications, 2010, http://sites.google.com/site/barbosaleonardo/Home/03.pdf.

RELIC Toolkit, http://code.google.com/p/relic-toolkit.

P. Schaumont, "One-Time Passwords from Your Watch," *Circuit Cellar* 262, 2012.

SOURCES

Arduino Mega 2560 board
Arduino | http://arduino.cc/en

ATmega2560 Microcontroller
Atmel Corp. | www.atmel.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:

—

One-Time Passwords from Your Watch
by Patrick Schaumont
***Circuit Cellar* 262, 2012**

The benefit of a one-time password is that you don't have to remember it. You only need to use it once. This article describes how to implement one-time passwords with a Texas Instruments Chronos Watch and use them to log into existing web services. Topics: Passwords, Algorithms, Google Authenticator, CCStudio

—

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

▶ CUWIN6300

▶ CUWIN5500

▶ CUWIN5200

Windows CE 6.0 Touch Controller

CUWIN

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

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

C-Programmable Modular Industrial Controller

MOACON

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

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

BASIC with LADDER LOGIC CONTROLLER

New

Integrated CUBLOC Controller and I/O Board

CB210

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

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

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

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

CUPC

Industrial Touch Panel PC with ATOM Processor

▶ CUPC-P200

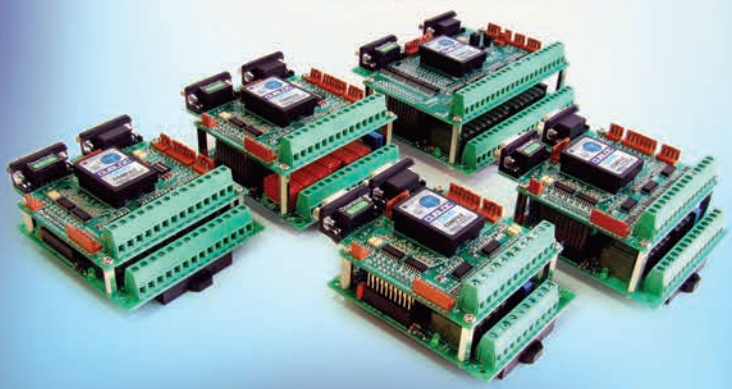
▶ CUPC-P220

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

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

CUSB

Integrated Industrial Controllers



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

by Richard Wotiz (USA)

EtherCAT Orchestra

Sometimes you can find industrial control networks in the most unexpected places. This article describes an unusual musical machine that's built around seven embedded controllers and an EtherCAT network.

I have long been interested in automatically controlled musical instruments. When I was little, I remember being fascinated whenever I ran across a coin-operated electromechanical calliope or a carnival hurdy-gurdy. I could spend all day watching the many levers, wheels, shafts, and other moving parts as it played its tunes over and over. Unfortunately, the mechanical complexity and expertise needed to maintain these machines makes them increasingly rare. But, in our modern world of pocket-sized MP3 players, there's still nothing like seeing music created in front of you.

I recently attended the Design West conference (formerly the Embedded Systems Conference) in San Jose, CA, and ran across an amazing contraption that reminded me of old carnival music machines. The system was created for Intel as a demonstration of its Atom processor family, and was quite successful at capturing the attention of anyone walking by Intel's booth (see Photo 1).

Photo 1—This is Intel's computer-controlled orchestra. It may not look like any musical instrument you've ever seen, but it's quite a thing to watch. The inspiration came from Animusic's "Pipe Dream," which appears on the video screen at the top.

The concept is based on Animusic's music video "Pipe Dream," which is a captivating computer graphics representation of a futuristic orchestra. The instruments in the video play when virtual balls strike against them. Each ball is launched at a precise time so it will land on an instrument the moment each note is played.

The demonstration, officially known as Intel's Industrial Control in Concert, uses high-speed pneumatic valves to fire practice paintballs at plastic targets

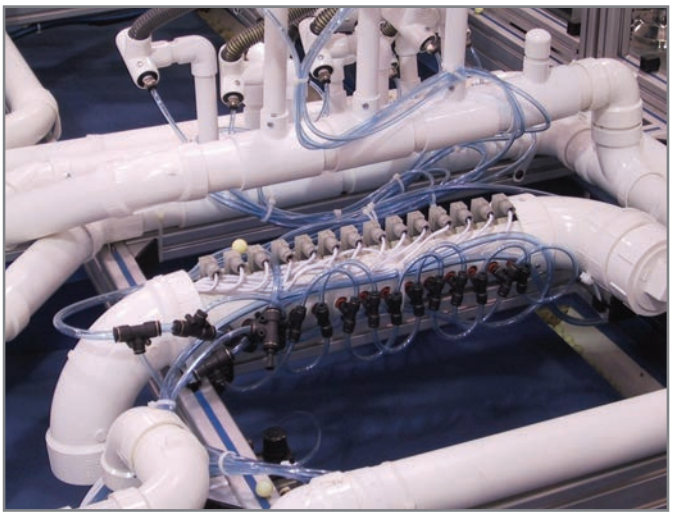


Photo 2—This is one of several sets of pneumatic valves. Air is supplied by the many tees below the valves and is sent to the ball-firing nozzles near the top of the photo. The corrugated hoses at the top supply balls to the nozzles.

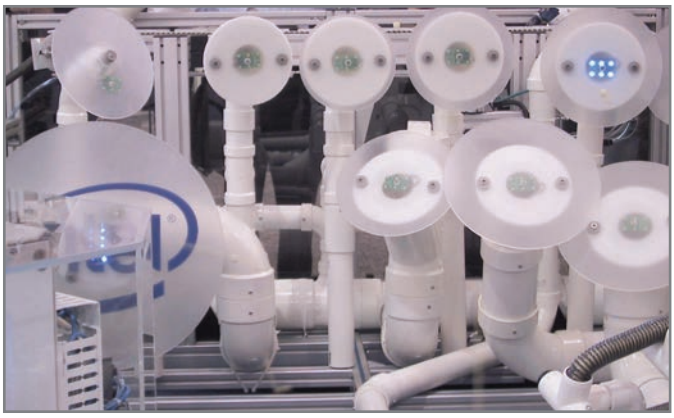


Photo 3—These are the targets at which the nozzles from Photo 2 are aimed. If you look closely, you can see a ball just after it bounced off the illuminated target at the top right.

of various shapes and sizes. The balls are made of 0.68"-diameter soft rubber. They put on quite a show bouncing around while a song played. Photo 2 shows one of the pneumatic firing arrays. The valves are the gray boxes lined up along the center. When each one opens, a burst of air is sent up one of the clear hoses to a nozzle to fire a ball. The corrugated black hoses at the top supply the balls to the nozzles. They're fed by paintball hoppers that are refilled after each performance. Each nozzle fires at a particular target (see Photo 3). Each target has an array of LEDs that shows when it's activated and a piezoelectric sensor that detects a ball's impact. Unfortunately, slight variations in the pneumatics and the balls themselves mean that not every ball makes it to its intended target. To avoid sounding choppy and incomplete, the musical notes are triggered by a fixed timing sequence rather than the ball impact sensors. Think of it as a form of mechanical lip syncing. There's a noticeable pop when a ball is fired, so the system sounds something like a cross between a pinball machine and a popcorn popper. You may expect that to detract from the music, but I felt it added to the novelty of the experience.

The control system consists of seven separate embedded systems, all based on Intel's Atom D525 dual-core microprocessor, on an Ethernet network (see Figure 1). One of the systems is responsible for the real-time control of the mechanism. It communicates over an Ethernet control automation technology (EtherCAT) bus to several slave units, which provide the I/O interface to the sensors and actuators.

EtherCAT

EtherCAT is a fieldbus providing high-speed, real-time control over a conventional 100 Mb/s Ethernet hardware infrastructure. It's a relatively recent technology, originally developed by Beckhoff Automation GmbH, and currently managed by the EtherCAT Technology Group (ETG), which was formed in 2003. You need to be an ETG member to access most of their specification documents, but information is publicly available. According to information on the ETG website, membership is currently free to qualified companies. EtherCAT was also made a part of international standard IEC 61158 "Industrial Communication Networks—Fieldbus Specifications" in 2007.

EtherCAT uses standard Ethernet data frames, but instead of each device decoding and processing an individual frame, the devices are arranged in a daisy chain, where a single frame is circulated through all devices in sequence. Any device with an Ethernet port can function as the master, which initiates the frame transmission. The slaves need specialized EtherCAT ports. A two-port slave device receives and starts processing a frame while simultaneously sending it out to the next device (see Figure 2). The last slave in the chain detects that there isn't a downstream device and sends its frame back to the previous device, where it eventually returns to the originating master. This forms a logical ring by taking advantage of both the outgoing and return paths in the full-duplex network. The last slave can also be directly connected to a second Ethernet port on the master, if one is available, creating a physical ring. This

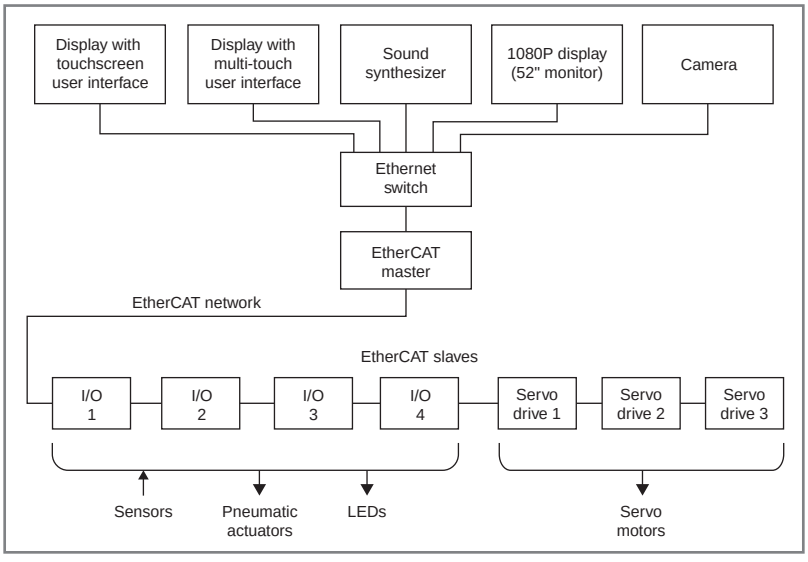


Figure 1—A block diagram of the system. Each block across the top is an embedded system providing some aspect of the user interface. The real-time interface is handled by the modules at the bottom. They're controlled by the EtherCAT master at the center.

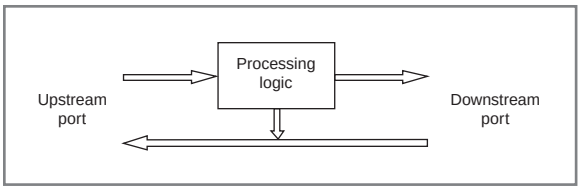


Figure 2—Each EtherCAT slave processes incoming data as it sends it out the downstream port. If it's at the end of the line and there is no downstream device connected, it sends the data back out the upstream port instead. Data coming in on the downstream port is always forwarded to the upstream port without any processing.

creates redundancy in case there is a break in the network. A slave with three or more ports can be used to form more complex topologies than a simple daisy chain. However, this wouldn't speed up network operation, since a frame still has to travel through each slave, one at a time, in both directions.

The EtherCAT frame, known as a telegram, can be transmitted in one of two different ways depending on the network configuration. When all devices are on the same subnet, the data is sent as the entire payload of an Ethernet frame, using an EtherType value of 0x88A4 (see Figure 3a). If the telegrams must pass through a router or switch onto a different physical network, they may be encapsulated within a UDP datagram using a destination port number of 0x88A4 (see Figure 3b), though this will affect network performance. Slaves do not have their own Ethernet or IP addresses, so all telegrams will be processed by all slaves on a subnet regardless of which transmission method was used. Each telegram contains one or more EtherCAT datagrams (see Figure 4). Each datagram includes a block of data and a command indicating what to do with the data. The commands fall into three categories. Write commands copy the data into a slave's memory, while read commands copy slave data into the datagram as it passes through. Read/write commands do both operations in sequence, first copying data from memory into the outgoing datagram, then moving data that was originally in the datagram into memory. Depending on the addressing mode, the read and write operations of a read/write command can both access the same or different devices. This enables fast propagation of data between slaves.

Each datagram contains addressing information that specifies which slave device should be accessed and the memory address offset within the slave to be read or written. A 16-bit value for each enables up to 65,535 slaves to be addressed, with a 65,536-byte address space for each one. The command code specifies which of four different addressing modes to use. Position addressing specifies a slave by its physical location on the network. A slave is selected only if the address value is zero. It increments the address as it passes the datagram on to the next device. This enables the master to select a device by setting the address value to the negative of the number of devices in the network preceding the desired device. This addressing mode is useful during system startup before the slaves are configured with unique addresses. Node addressing specifies a slave by its configured address, which the master will set during the startup process. This mode enables direct access to a particular device's memory or control registers. Logical addressing takes advantage of one or more fieldbus memory management units (FMMUs) on a slave device. Once configured, a FMMU will translate a logical address to any desired physical memory address. This may include the ability to specify individual bits in a data byte, which provides an efficient way to control specific I/O ports or register bits without having to send any more data than needed. Finally, broadcast addressing selects all slaves on the network. For broadcast reads, slaves send out the logical OR of their data with the data from the incoming datagram.

Each time a slave successfully reads or writes data contained in a datagram, it increments the working counter value (see Figure 4). This enables the master to confirm that all the slaves it was expecting to communicate with actually handled the data sent to them. If a slave is disconnected, or its configuration changes so it is no longer being addressed as expected, then it will no longer increment the counter. This alerts the master to rescan the network to confirm the presence of all devices and reconfigure them, if necessary. If a slave wants to alert the master of a high-priority event, it can set one or more bits in the IRQ field to request the master to take some predetermined action.

TIMING

Frames are processed in each slave by a specialized EtherCAT slave controller (ESC), which extracts incoming data and inserts outgoing data into the frame as it passes through. The ESC operates at a high speed, resulting in a typical data delay from the incoming to the outgoing network port of less than 1 µs. The operating speed is often dominated by how fast the master can process the data, rather than the speed of the network itself. For a system that runs a process feedback loop, the master has to receive data from the previous cycle and process it before sending out data for the next cycle. The minimum cycle time T_{CYC} is given by: $T_{CYC} = T_{MP} + T_{FR} + N \times T_{DLY} + 2 \times T_{CBL} + T_J$. T_{MP} = master's processing time, T_{FR} = frame transmission time on the network (80 ns per data byte + 5 µs frame overhead), N = total number of slaves, T_{DLY} = sum of the forward and

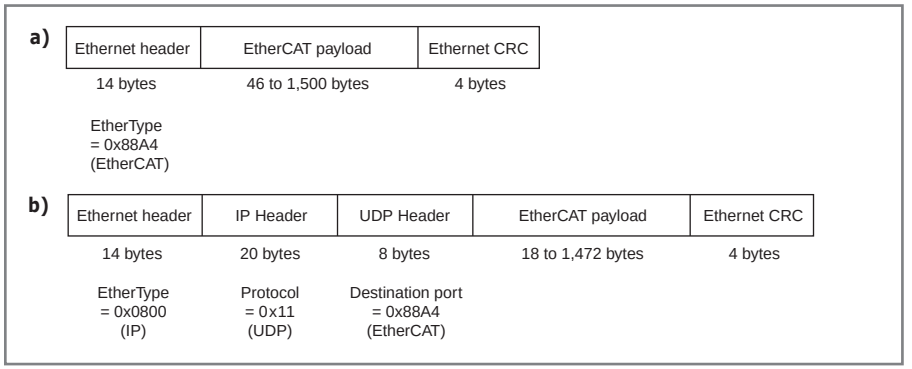


Figure 3a—An EtherCAT frame uses the standard Ethernet framing format with very little overhead. The payload size shown includes both the EtherCAT telegram and any padding bytes needed to bring the total frame size up to 64 bytes, the minimum size for an Ethernet frame. **b—**The payload can be encapsulated inside a UDP frame if it needs to pass through a router or switch.

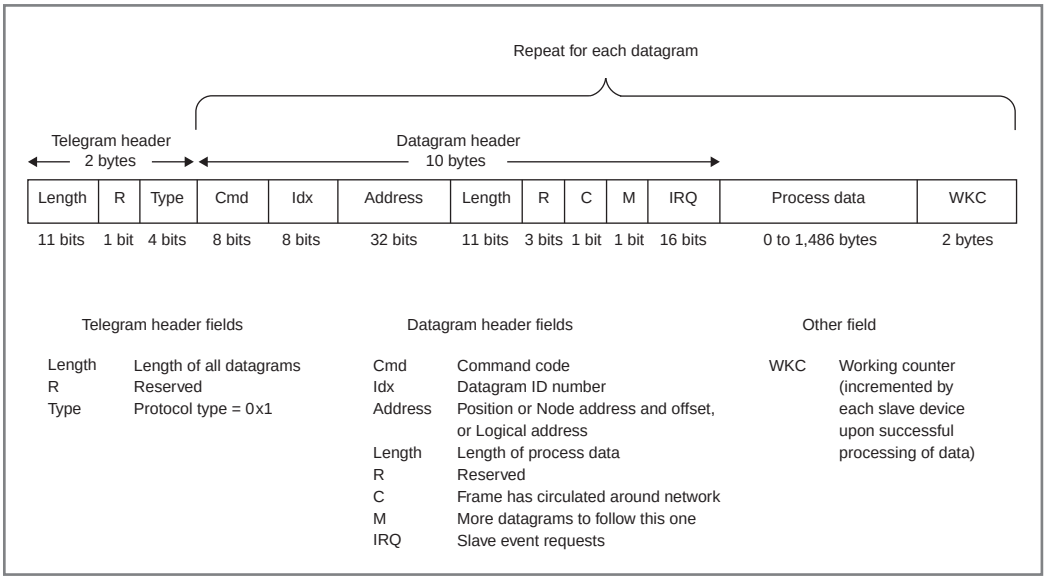


Figure 4—An EtherCAT telegram consists of a header and one or more datagrams. Each datagram can be addressed to one slave, a particular block of data within a slave, or multiple slaves. A slave can modify the datagram's Address, C, IRQ, Process data, and WKC fields as it passes the data on to the next device.

return delay times through each slave (typically 600 ns), T_{CBL} = cable propagation delay (5 ns per meter for Category 5 Ethernet cable), and T_J = network jitter (determined by master).[1]

A slave's internal processing time may overlap some or all of these time windows, depending on how its I/O is synchronized. The network may be slowed if the slave needs more time than the total cycle time computed above. A maximum-length telegram containing 1,486 bytes of process data can be communicated to a network of 1,000 slaves in less than 1 ms, not including processing time.

Synchronization is an important aspect of any fieldbus. EtherCAT uses a distributed clock (DC) with a resolution of 1 ns located in the ESC on each slave. The master can configure the slaves to take a snapshot of their individual DC values when a particular frame is sent. Each slave captures the value when the frame is received by the ESC in both the outbound and returning directions. The master then reads these values and computes the propagation delays between each device. It also computes the clock offsets between the slaves and its reference clock, then uses these values to update each slave's DC to match the reference. The process can be repeated at regular intervals to compensate for clock drift. This results in an absolute clock error of less than 1 µs between devices.

MUSICAL NETWORKS

The orchestra's EtherCAT network is built around a set of modules from National Instruments. The virtual conductor is an application running under LabVIEW Real-Time on a CompactRIO controller, which functions as the master device. It communicates with four slaves containing a mix of digital and analog I/O and three slaves consisting of servo motor drives. Both the master and the I/O slaves contain a FPGA to implement any custom local processing that's necessary to keep the data flowing. The system runs at a cycle time of 1 ms, which provides enough timing resolution to keep the balls properly flying.

I hope you've enjoyed learning about EtherCAT—as well as the fascinating musical device it's used in—as much as I have.

Author's note: I would like to thank Marc Christenson of SISU Devices, creator of this amazing device, for his help in providing information on the design.

Editor's note: Richard Alan Wotiz started taking products apart when he was old enough to pick up a soldering iron. In 1991, he started his design consulting business, and his specializations were hardware and software for consumer products and children's toys. Richard passed away on May 30, 2012. The entire Circuit Cellar *community benefitted from his work, and his many projects and articles will continue to inspire engineers for years to come.*

REFERENCE

[1] National Instruments Corp., "Benchmarks for the NI 9144 EtherCAT Slave Chassis," http://zone.ni.com/devzone/cda/tut/p/id/10596.

RESOURCES

Animusic, LLC, www.animusic.com.

Beckhoff Automation GmbH, "ET1100 EtherCAT Slave Controller Hardware Data Sheet, Version 1.8", 2010, www.beckhoff.com/english/download/ethercat_development_products.htm.

EtherCAT Technology Group, "The Ethernet Fieldbus", 2009, www.ethercat.org/pdf/english/ETG_Brochure_EN.pdf.

Intel, Atom microprocessor, www.intel.com/content/www/us/en/processors/atom/atom-processor.html.

SOURCES

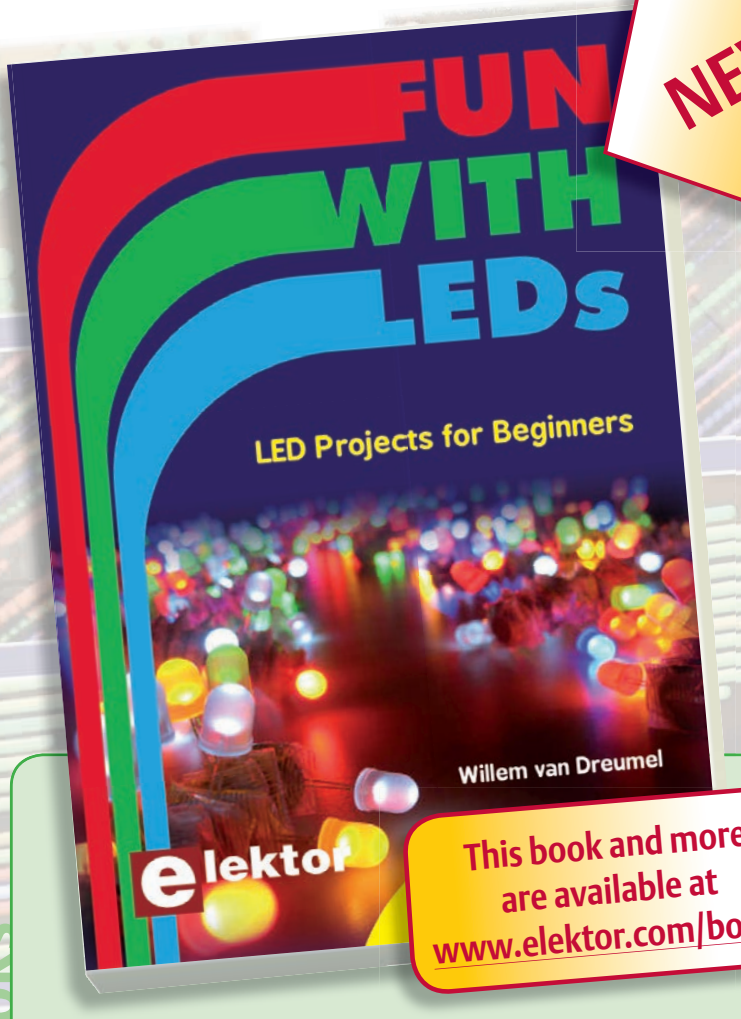
Atom D525 dual-core microprocessor
Intel Corp. | www.intel.com

LabVIEW Real-Time modules, CompactRIO controller, and EtherCAT devices
National Instruments Corp. | www.ni.com

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

Elektor Shop

The world of electronics at your fingertips!



NEW!

This book and more are available at www.elektor.com/books

Books

Associated 60-piece Starter Kit available

Fun with LEDs

This booklet presents more than twenty exciting projects covering LEDs, aimed at young & old. From an Air Writer, a Party Light, Running Lights, a LED Fader right up to a Christmas Tree. Use this book to replicate various projects and then put them into practice. To give you a head start each project is supported by a brief explanation, schematics and photos. In addition, the free support page on the Elektor website has a few inspiring video links available that elaborate on the projects. A couple of projects employ the popular Arduino microcontroller board that's graced by a galaxy of open source applications. The optional 60-piece Starter Kit available with this book is a great way to get circuits built up and tested on a breadboard, i.e. without soldering.

96 pages • ISBN 978-1-907920-05-9 • $38.00

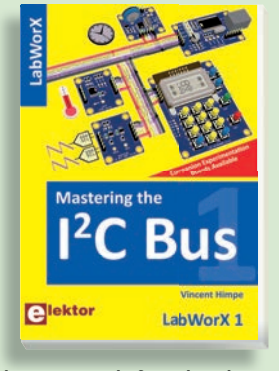
Bestseller!

A comprehensive and practical how-to guide

Design your own PC Visual Processing and Recognition System in C#

This book is aimed at Engineers, Scientists and enthusiasts with developed programming skills or with a strong interest in image processing technology on a PC. Written using Microsoft C# and utilizing object-oriented practices, this book is a comprehensive and practical how-to guide. The key focus is on modern image processing techniques with useful and practical application examples to produce high-quality image processing software. All code examples used are available – free of charge – from the Elektor website.

307 pages • ISBN 978-1-907920-09-7 • $57.30



LabWorX: Straight from the Lab to your Brain

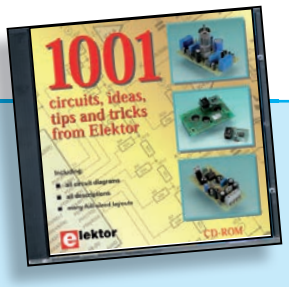
Mastering the I²C Bus

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

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

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

CD/DVD-ROMs



Circuits, ideas, tips and tricks from Elektor

CD 1001 Circuits

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

ISBN 978-1-907920-06-6 • $55.70

A whole year of Elektor magazine onto a single disk

DVD Elektor 2011

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

ISBN 978-90-5381-276-1 • $37.90

Embedded Linux Made Easy

Today Linux can be found running on all sorts of devices, even coffee machines. Many electronics enthusiasts will be keen to use Linux as the basis of a new microcontroller project, but the apparent complexity of the operating system and the high price of development boards has been a hurdle. Here Elektor solves both these problems, with a beginners' course accompanied by a compact and inexpensive populated and tested circuit board. This board includes everything necessary for a modern embedded project: a USB interface, an SD card connection and various other expansion options. It is also easy to hook the board up to an Ethernet network.

Populated and tested Elektor Linux Board

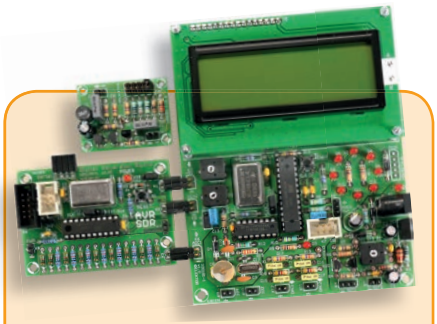
Art.# 120026-91 • $93.30

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

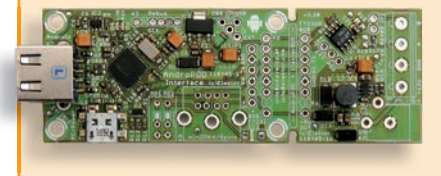


AVR Software Defined Radio

This package consists of the three boards associated with the AVR Software Defined Radio articles series in Elektor, which is built around practical experiments. The first board, which includes an ATtiny2313, a 20 MHz oscillator and an R-2R DAC, will be used to make a signal generator. The second board will fish signals out of the ether. It contains all the hardware needed to make a digital software-defined radio (SDR), with an RS-232 interface, an LCD panel, and a 20 MHz VCXO (voltage-controlled crystal oscillator), which can be locked to a reference signal. The third board provides an active ferrite antenna. This bundle also includes the assembled and tested FT232R USB/Serial Bridge/BOB PCB!

Signal Generator + Universal Receiver + Active Antenna: PCBs and all components + USB-FT232R breakout-board

Art.# 100182-72 • $133.00



AndroPod

With their high-resolution touchscreens, ample computing power, WLAN support and telephone functions, Android smartphones and tablets are ideal for use as control centres in your own projects. However, up to now it has been rather difficult to connect them to external circuitry. Our AndroPod interface board, which adds a serial TTL port and an RS485 port to the picture, changes this situation.

Andropod module with RS485 Extension

Art.# 110405-91 • $74.00

Kits & Modules

by George Novacek (Canada)

Project Development (Part 1)

Plans, Schedules, and Task Management

There's an old saying, "If you don't know where you want to go, you'll never get there." Engineering projects are the same way. To be successful, proper planning is essential. This article series begins with tips for project planning, scheduling, and management.

It's been said that paperwork is the bane of engineering. It's hard to argue against it. Paradoxically, thanks to computers, there's no improvement in sight. Contrary to the promise of the "paperless office"—where once a few typewritten pages with hand-drawn illustrations and corrections by the whitener sufficed—a bound book with full-color illustrations is expected today. Since the introduction of office computers, I have observed a steady growth in paper consumption. That said, I hasten to add that paperwork is something engineering cannot do without. It is only the form and the extent of it that, in my view, has gone over the top and represents a significant, unnecessary drain on resources. Let's consider some of the documentation we can't do without.

DESIGN PLANNING

Figure 1 depicts the fundamental stages in product design. It makes no difference whether you work for a big company or are a "one-man band." The steps are the same, but their extent will vary according to the type of product, the customer requirements, and the size of your team. Especially important are the "milestones" shown in green in Figure 1. They enable quick assessment of the project status, can serve to schedule progressive payments, and provide tracking data for progress reports. They also serve for planning formal reviews, which I'll address in detail in a future article.

Many engineers dislike reviews and consider them a nuisance and nothing more than "dog and pony shows" for the customer. Quite the opposite is true. Reviews are to the engineer's advantage. When attended and signed off by the customer, they confirm the project's compliance with the customer's requirements and thus prevent the customer from demanding unpaid changes or a redesign at a later time.

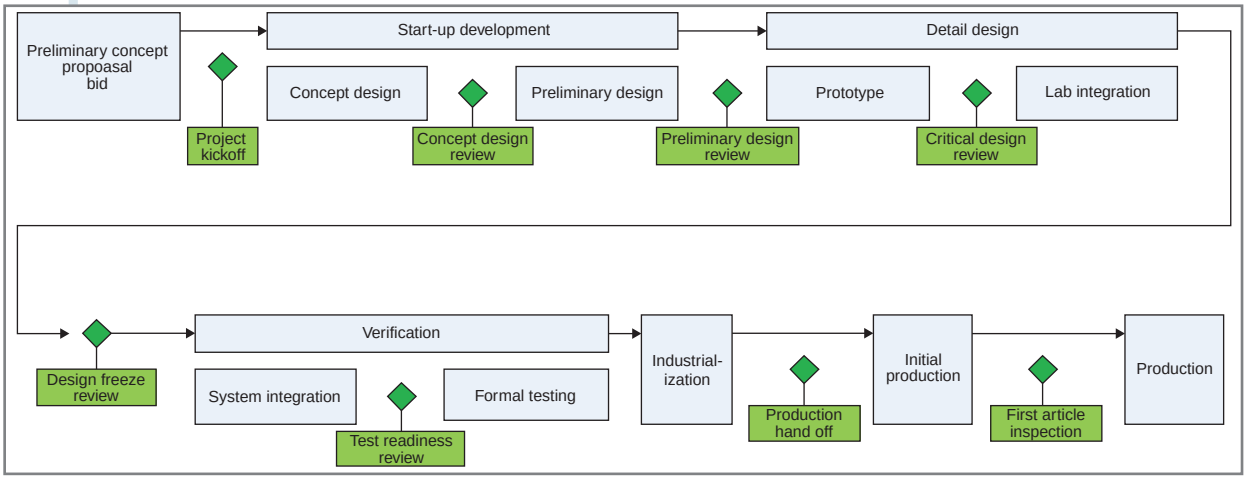


Figure 1—Product development steps

The first important documents needed to successfully execute a design project are plans. In the next several articles, I'll focus on what specific engineering plans should address and how they affect the development process. Engineers tasked with preparation of the plans must have a good understanding of their purpose and subjects. In addition to charting the way, the data prepared for these documents are crucial for bidding—whether the customer is another company or an internal sales department. Budgeting, scheduling, resource planning, and procuring materials and services are necessary. The data may vary from a few Excel worksheets to a full-color publication some companies may demand.

A schedule with identified resource needs is the key to preparing the budget and many plans. Assuming the proposal is technically sound, the schedule and price must also be realistic. Price alone cannot win a bid if the buyer isn't confident the product will be developed on time and the company will not get into financial trouble due to a low-balled price.

Most large companies, especially those supplying military or safety-critical products, have their own data item descriptions (DIDs) guiding the document's content. ISO9001 certifications also require formal plans for many activities. But, even if you work for yourself or in an industry where formal plans are not called for, you should spend time planning and documenting your intended activities.

SCHEDULING & TASK MANAGEMENT

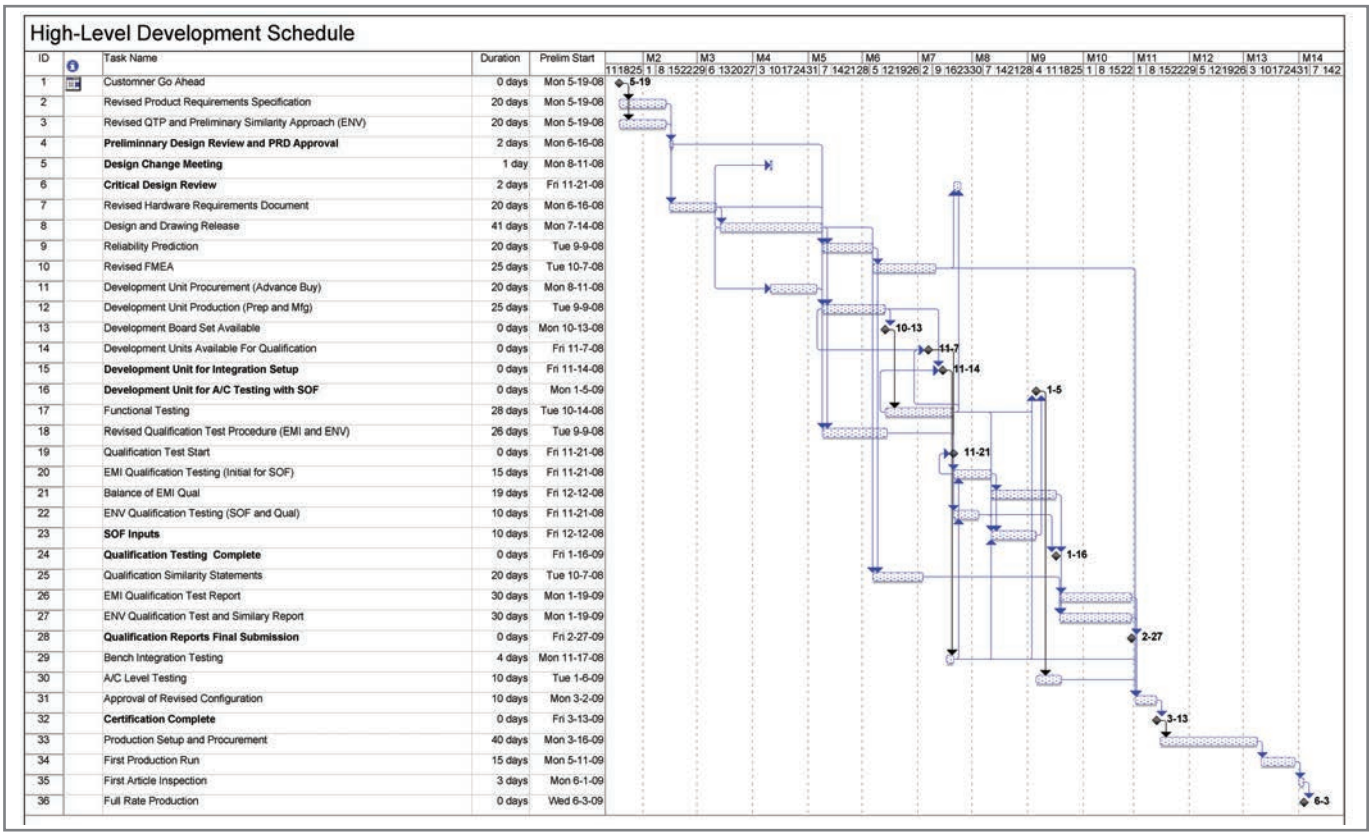
I consider a schedule the most important document. In fact, this is the plan. The schedule in Figure 2 is called a Gantt chart. It graphically shows the work breakdown in tasks. It shows dependencies and supports scheduling and timing of the individual tasks, their resourcing for the most efficient execution, and the shortest critical path to delivery and planning of all activities. I use Microsoft Project. It's reasonably priced and although its interface with the Microsoft Office suite leaves a lot to be desired, it is a competent scheduling tool. With a few workarounds, I find it preferable to more sophisticated competition.

Similar to the computer directory folder structure you may be familiar with, Microsoft Project enables each task to be broken down into subtasks with nearly infinite detail. Dependencies and milestones, progress status, resource leveling, and costs can all be entered into the Microsoft Project schedule. The end result is a well-developed plan. Additional verbiage required by some companies to make the development plan look like a formidable book merely puts into words what can already be gleaned from the schedule.

There are many project-management tools available, from the reasonably priced Microsoft Project to sophisticated programs costing hundreds of thousands of dollars. I have satisfactorily used Microsoft Project for years. I also experienced a monstrosity, tracking each engineer's work until the engineers refused to use it because it forced them to spend more time entering data into it than actually working on the project.

I can't resist making an observation. All these software tools, when used correctly, can make you more efficient, provided you keep in mind they are just tools. They possess no more intelligence than a hammer, and you must never allow these tools to make decisions for you. A competent project manager can do without them, albeit less efficiently. Poor managers equipped with these tools get so impressed with the reams of data they produce, they become convinced of their own invincibility and will fail more often than not.

Preparation of a development schedule is a hard, frustrating job that needs to be undertaken by knowledgeable, experienced senior engineers, especially when it comes to estimating the time needed to design something new. Design engineers often tread on unbroken ground. Consequently, they seem to be allergic to—in their minds—silly questions such as, "How many lines of code will it take?" and "When do you think the hardware will be complete?" And, worst of all, when something does not work, "How long will it take you to fix it?" Common answers to those legitimate questions cannot be printed in this magazine. This is when good engineering managers don't ask. They fall back on their knowledge and



High-Level Development Schedule

ID	Task Name	Duration	Prelim Start
1	Customner Go Ahead	0 days	Mon 5-19-08
2	Revised Product Requirements Specification	20 days	Mon 5-19-08
3	Revised QTP and Preliminary Similarity Approach (ENV)	20 days	Mon 5-19-08
4	**Preliminnary Design Review and PRD Approval**	2 days	Mon 6-16-08
5	**Design Change Meeting**	1 day	Mon 8-11-08
6	**Critical Design Review**	2 days	Fri 11-21-08
7	Revised Hardware Requirements Document	20 days	Mon 6-16-08
8	Design and Drawing Release	41 days	Mon 7-14-08
9	Reliability Prediction	20 days	Tue 9-9-08
10	Revised FMEA	25 days	Tue 10-7-08
11	Development Unit Procurement (Advance Buy)	20 days	Mon 8-11-08
12	Development Unit Production (Prep and Mfg)	25 days	Tue 9-9-08
13	Development Board Set Available	0 days	Mon 10-13-08
14	Development Units Available For Qualification	0 days	Fri 11-7-08
15	**Development Unit for Integration Setup**	0 days	Fri 11-14-08
16	**Development Unit for A/C Testing with SOF**	0 days	Mon 1-5-09
17	Functional Testing	28 days	Tue 10-14-08
18	Revised Qualification Test Procedure (EMI and ENV)	26 days	Tue 9-9-08
19	Qualification Test Start	0 days	Fri 11-21-08
20	EMI Qualification Testing (Initial for SOF)	15 days	Fri 11-21-08
21	Balance of EMI Qual	19 days	Fri 12-12-08
22	ENV Qualification Testing (SOF and Qual)	10 days	Fri 11-21-08
23	**SOF Inputs**	10 days	Fri 12-12-08
24	**Qualification Testing Complete**	0 days	Fri 1-16-09
25	Qualification Similarity Statements	20 days	Tue 10-7-08
26	EMI Qualification Test Report	30 days	Mon 1-19-09
27	ENV Qualification Test and Similary Report	30 days	Mon 1-19-09
28	**Qualification Reports Final Submission**	0 days	Fri 2-27-09
29	Bench Integration Testing	4 days	Mon 11-17-08
30	A/C Level Testing	10 days	Tue 1-6-09
31	Approval of Revised Configuration	10 days	Mon 3-2-09
32	**Certification Complete**	0 days	Fri 3-13-09
33	Production Setup and Procurement	40 days	Mon 3-16-09
34	First Production Run	15 days	Mon 5-11-09
35	First Article Inspection	3 days	Mon 6-1-09
36	Full Rate Production	0 days	Wed 6-3-09

Figure 2—Example of a product development schedule

experience to save the day.

Design planning (i.e., mainly scheduling) is more difficult during these days of concurrent engineering. The specification for an electronic controller can be a moving target for months. In the meantime, the customer and the corporate management insists on obtaining definite plans to budget and to assign resources. The customer wants the lowest price and the shortest delivery time. The customer also wants a guarantee the project will be completed on time. The buyer dreads his employer not being able to ship a $100 million aircraft, while waiting for an embedded controller worth $10,000.

This inability to accurately schedule an engineering development plan can sometimes lead both the developer and the customer to fudge schedules to the point of becoming ridiculous and useless. Once a customer's purchasing manager shook my hand, congratulated me for winning the bid, and added, "By the way, according to our schedule, your delivery is already six months late!" He meant it. For the following three years, I received harassing phone calls every Friday afternoon and monthly "nastygrams" calling me the worst subcontractor they had ever worked with. When we delivered—42 months "behind schedule"—we received a letter of commendation for a job well done.

Next month, I'll focus on the development process, milestones, and design reviews.

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

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:

Design Development (Part 1)
Planning and Tools
by George Martin
***Circuit Cellar* 250, 2011**
You can read and write C code. The next step is to start building something. Follow these tips as you prepare to develop your next product. Topics: Design Planning, C Programming

Design Development (Part 2)
Product Implementation
by George Martin
***Circuit Cellar* 252, 2011**
The second part of this article series digs deeper into the design process by describing how schematic capture software is used to define the hardware portion of a design. Topics: Design Planning, Hardware, Schematic Capture Software

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

Learn to make your products

Rugged, Low-Power, Fast, Small and Easy to Use

at the only conference dedicated entirely to flash memory!

Are you struggling with crucial Solid State Drive (SSD) decisions? Can SSDs resolve your application bottlenecks? How can you maximize SSD performance? Flash Memory Summit will explore new frontiers in enterprise storage and help you make the right choices. Summit highlights include:

Pre-Conference Seminar: SSDs—The Fundamentals
Forums and Tutorials
Nine Keynote Presentations
FMS Theater

Flash Memory SUMMIT

August 21-23, 2012
Santa Clara Convention Center
Santa Clara, CA

REGISTER ONLINE

www.FlashMemorySummit.com

Circuit Cellar Magazine readers: Enter code SPGP and receive a $100 discount!

by George Martin (USA)

Switch Debouncing

Interfacing to a Simple Serial Device

Writing code to debounce switch inputs can help save manufacturing costs, such as the PCB used to hold the parts, the labor required to assemble the parts, and the power used to move the parts. This article explores various types of signaling over a single-input line, beginning with the design of switch-debouncing routines.

The following is true. The names have been changed to protect the innocent. This month, consider the following series of requests. It all starts as you are looking over a project's hardware design. Someone has requested hardware debounce circuits to interface the switch inputs to the microprocessor. That's okay, and these parts work well, but you know you could write the code that would debounce the switch inputs. You know this could save the cost of those parts and other expenses: the PCB area used to hold the parts, the labor required to assemble the parts, and the power used to move the parts. So, you volunteered to make the change.

Why do you need to debounce the switch inputs at all? The microprocessor is digital and can read the switch inputs much faster than a human pressing the switch, so the information will contain many ones (open) and zeros (closed) in sequence. Why is

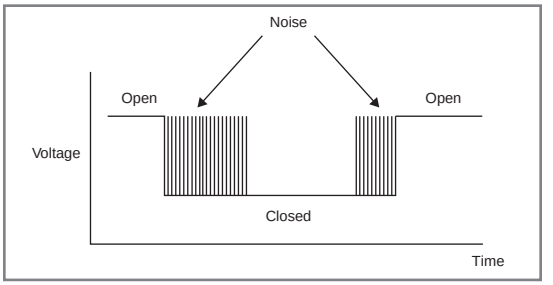


Figure 1—The signals coming from a switch

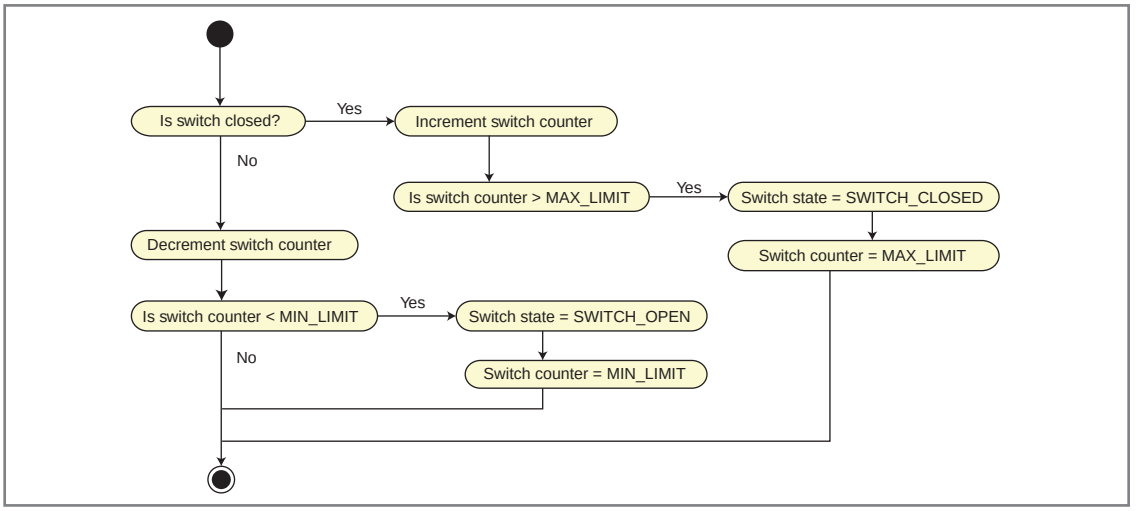


Figure 2—The state design in UML notification

that? The best switches might have gold contacts and a snap action that amplifies the switch movement. The worst switches might have tin contacts that slide against each other at the user provided rate of motion. This sliding action helps the switch contacts remove any corrosion that tends to build up. My company used to implement 15 V and several milliamps for each set of switch contacts. This amount of power kept the contacts clean. As the power consumed by modern microprocessors decreased, the high power and current used to clean the switch contacts are gone. Switch manufactures have responded with different types of constructions as they manufacturer each switch. One common switch contact approach is to use bifurcated contacts. This is a self-cleaning design as corrosion is cleared off the contacts with each press. That's the good news. The not-so-good news is that the switch signal looks like Figure 1.

As you can see, the signal going from switch open to switch closed starts as clearly open and ends as clearly closed. Information is read in between both open and closed. With a fast enough processor, the signal could look like several switch closures instead of just one event. The voltage in Figure 1 is shown on the vertical axis and time is shown on the horizontal axis. It is key to realize that time is the parameter used to get through all the switch noise.

Listing 1—The code loop

```
if ((LastState == SWITCH_CLOSED) &&  (State == SWITCH_OPEN)) {
        then process a switch open event here
             LastState = SWITCH_OPEN
}
if ((LastState == SWITCH_OPEN) &&  (State == SWITCH_CLOSED)) {
        then process a switch close event here
             LastState = SWITCH_CLOSED
}
```

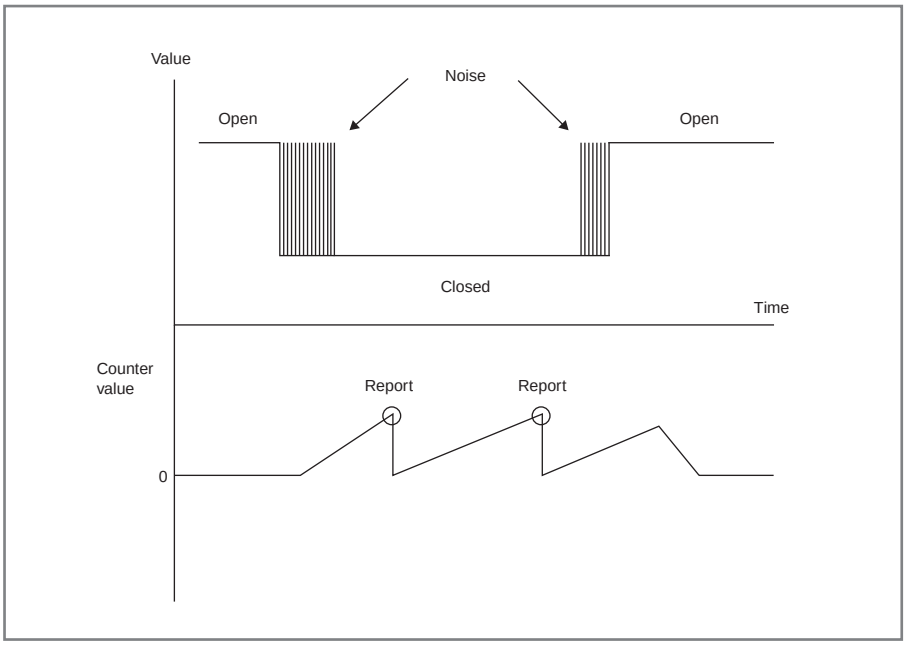


Figure 3—Changing the detection values increases or decreases the code's speed

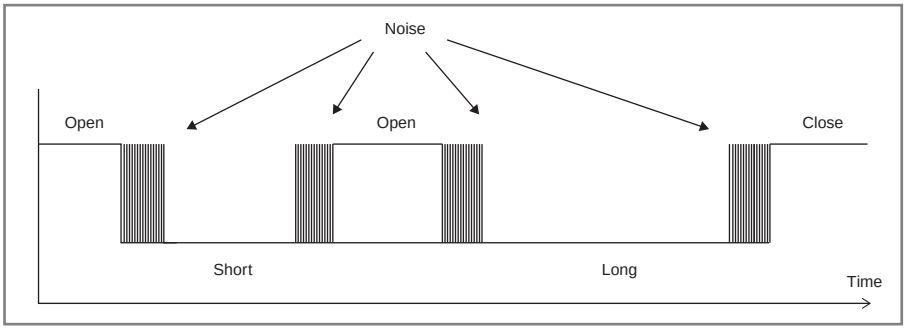


Figure 4—Timing diagram of short and long key closure

SWITCH DEBOUNCING WITH TIME

Consider implementing a design that uses time for switch debouncing. One approach I've used is to have a 10-ms interrupt. In that interrupt, the switch input is read. A counter is decremented if it gives an open condition and incremented if it gives a closed condition. Be careful to watch the ends of the range (0 to 265 for an 8-bit unsigned integer) of this variable. If the counter variable is lower than some low limit, hold that variable at that low limit. If the counter variable is higher than some high limit, hold the variable at that high limit.

The other part of this design is that each switch has a state associated with it. The states I use are:

```
#define
SWITCH_CLOSED 1
SWITCH_OPEN   2
#define
```

I also have a variable that saves the last state. Figure 2 depicts the state design. This routine is called every interrupt (e.g., 10 ms), and the switch timer counter variable eliminates the switch bounce. The state machine helps the code determine state changes. I'm holding the switch counter variable at either the high or low limits. And I usually set these limits not too close to 0 and full scale. The code loop is shown in Listing 1.

TYPOMATIC FUNCTION

Assume the switch has been debounced. You can change the code limits so the code can operate faster or slower. Imagine you're proud of your work and you install it in the redesigned board assembly. Everything is good. Now consider adding a typomatic feature to a switch press. This feature is similar to a PC's keyboard action. If you hold down a switch, the key keeps repeating. This saves the user from pressing the key several times to scroll through

a list or to step through unused entry fields on an entry form. It sounds like a great addition. And if you have a timing diagram of the states, you would see that as the typomatic function is activated, you would need to force the last state to be opened and the current state to be closed. This could be implemented with values of the timer created for debouncing the switch. If the timer runs from 0 to 256, you could use 100 to reset the change from open to closed and keep the timer running. When the timer reaches 150, you could force the states to be `LastState` is opened and `State` is closed. This would inform the code that there is another switch closure. After processing the function, you would need to reset the timer value to, say, 100. The numeric values combined with the interrupt rate would determine the code's speed. You could increase or decrease this speed by changing the detection values (see Figure 3).

KEY PRESS FUNCTION

Now let's consider another scenario. A few days after the code installation, the happy customer gives you a call: Can a short key press and a long key press be detected on a different switch? Figure 4 shows the timing for the design. The hardware is the same, so you still need to debounce the inputs. But the information is not the open/close originally shown. The information is the length of the key pressed. And there are only two types of key presses: short and long. You have different information with this set of requirements. In Figure 1, just detected transitions form open to close and back again. Now, in Figure 4, you need to detect long and short closures and report that to the calling routine. First, you must simplify this requirement so that you only report after a complete open-close-open cycle. And the state machine will keep track of the length of the closed time.

Take a look at the state machine in Figure 5. It looks very similar to Figure 2. Notice a timer was added to measure how long the switch was closed. And you'd need to reset that timer once the switch is open and then save the value for how long the switch was closed. Also be aware that at start up, you'd need to put proper values into these variables. After that, they should take care of themselves. See Listing 2 for the code.

Imagine you detected the switch state changing from closed to open and the state machine measured how long the switch was closed. You were a star. The customer was happy and you delivered and tested the code. But now it's a few days later and you get a call. The code is working well, but the customer requests one simple change. (This is your imaginary scenario. But it actually happened to me.)

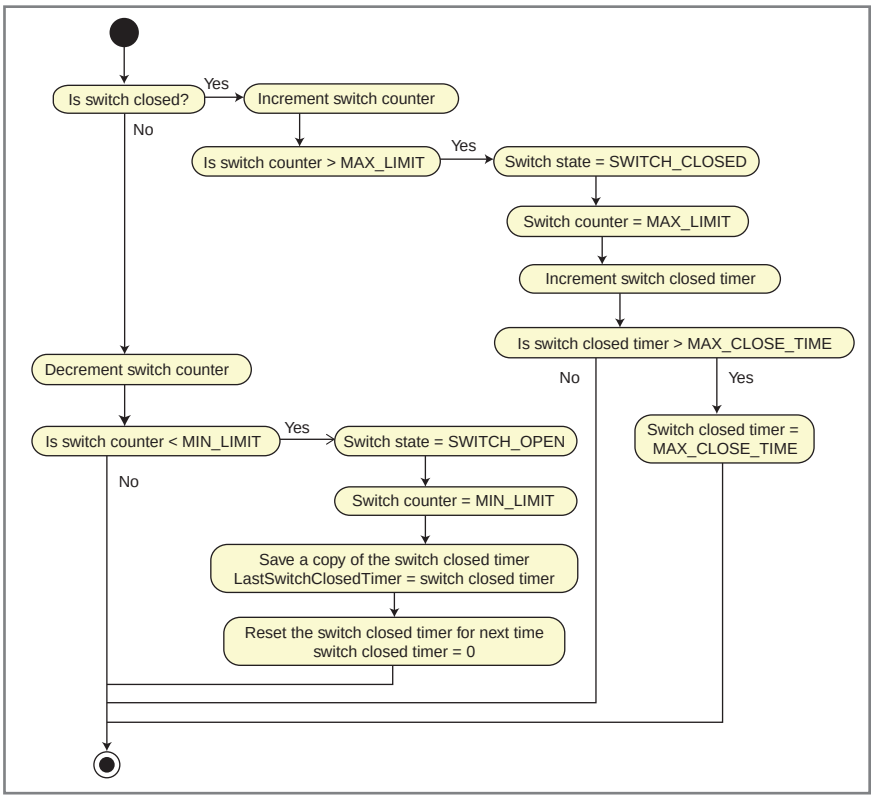


Figure 5—Long-short closure time detection

Listing 2—The code for the variable's proper values needed at start-up

```
if ((LastState == SWITCH_CLOSED) &&  (State == SWITCH_OPEN)) {
SwitchCloseTime = LastSwitchClosedTimer.
        then process a switch close event here
             LastState = SWITCH_OPEN
}
```

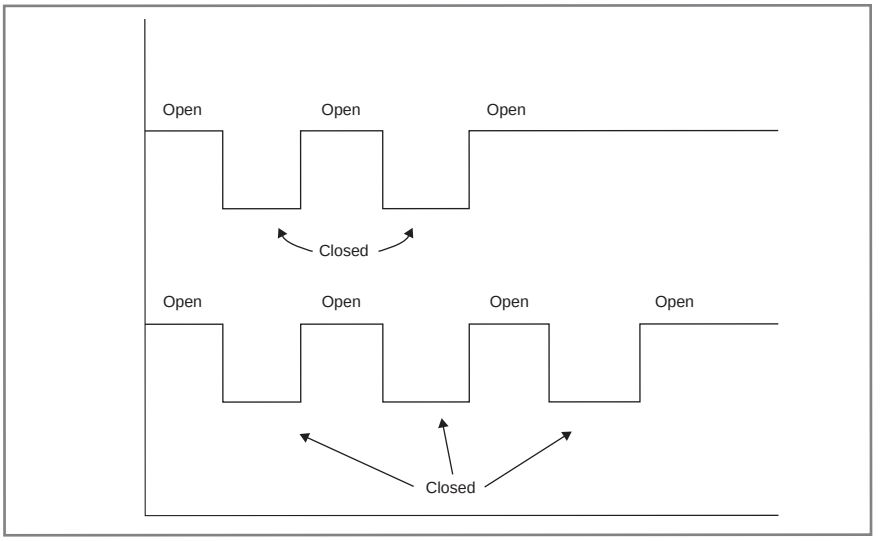


Figure 6—Quick switch closure in succession

HOT CHIPS 24

ADVANCE PROGRAM

HOT CHIPS 24
August 27-29, 2012

A Symposium on High-Performance Chips
Flint Center for the Performing Arts, Cupertino, CA

HOT CHIPS brings together designers and architects of high-performance chips, software, and systems. The tutorial and presentation sessions focus on up-to-the-minute developments in leading-edge industrial designs and research projects. The conference breaks will also feature a student poster session.

Mon. Aug. 27

Morning Tutorial
The Evolution of Mobile SoC Programming — **Khronos**

Afternoon Tutorial
Die Stacking — **AMD, Amkor, SK Hynix, Xilinx**

Tuesday August 28

Microprocessors
- Architecture and Power Management of the 3rd-Generation Intel Core Microarchitecture — **Intel**
- AMD's Jaguar Microprocessor — **AMD**
- Impresa: Efficient Performance on a Fully-Synthesizable Core — **MIPS**

Fabrics & Interconnects
- Swizzle Switch: A Self-Arbitrating High-Radix Crossbar for NoC Systems — **U. Michigan**
- FPGA Augmented ASICs: The Time Has Come — **Solarflare**
- The SwitchX Architecture — **Mellanox**

Keynote 1 Ambidextrous Ecosystems
Mark Papermaster, CTO — AMD

Many Core and GPU
- AMD HD7970 Graphics Core Next (GCN) Architecture — **AMD**
- AMD Trinity Fusion APU — **AMD**
- Knights Corner, Intel's first Many Integrated Core (MIC) — **Intel**

Multimedia & Imaging
- ADI's BF60x Vision Focused Digital Signal Processor SOC — **ADI**
- Visconti2 - A Heterogeneous Multi-Core SoC for Image-Recognition — **Toshiba**

Integration
- Centip3De: A 64-Core, 3D Stacked, Near-Threshold System — **U. Michigan**
- FPGAs with 28Gbps Transceivers Built with Heterogeneous Stacked-Silicon Interconnects — **Xilinx**

Keynote 2 Changes in the Wireless Industry
Marcus Weldon, CTO — Alcatel-Lucent

Wednesday August 29

Technology & Scalability
- Floating-Point Matrix Processing using FPGAs — **Altera**
- An IA-32 Processor with a Wide Voltage Operating Range in 32nm CMOS — **Intel**
- Taking Advantage of SuVolta DDC Technology to Lower Power in Chips — **Suvolta**

SOC
- High Performance and Efficient Single-Chip Small Cell Base Station SoC — **Cavium**
- FSM: a Highly-Integrated Chipset Solution for the Small Cell Market — **Qualcomm**
- Medfield Smartphone - Intel's ATOM Z2460 Processor — **Intel**

Keynote 3 Cloud Transforms IT, Big Data Transforms Business
Pat Gelsinger, COO Infrastructure Products — EMC

Data Center Chips
- POWER7+: IBM's Next Generation POWER Microprocessor — **IBM**
- Xeon E5 2600: Power/Performance Efficiency in the Data Center — **Intel**
- X-Gene: AppliedMicro's 64bit ARM CPU and SOC — **APM**

Big Iron
- SPARC64 X: Fujitsu's New-Generation, 16-Core Processor for the Next Generation UNIX Servers — **Fujitsu**
- 16-core SPARC T5 CMT Processor with Glueless Scaling to 8-Sockets — **Oracle**
- IBM zNext: the 3rd Generation High-Frequency Microprocessor Chip — **IBM**

Please visit us on the web: **http://www.hotchips.org**
or drop us a line via Email: **info2012@hotchips.org**

IEEE

IEEE computer society

This is a preliminary program; changes may occur. For the most up-to-the-minute details, please visit our web site.

acm In-Cooperation

SSCS

A Symposium of the Technical Committee on Microprocessors and Microcomputers of the IEEE Computer Society and the Solid state Circuits Society in cooperation with the Association of Computing Machinery

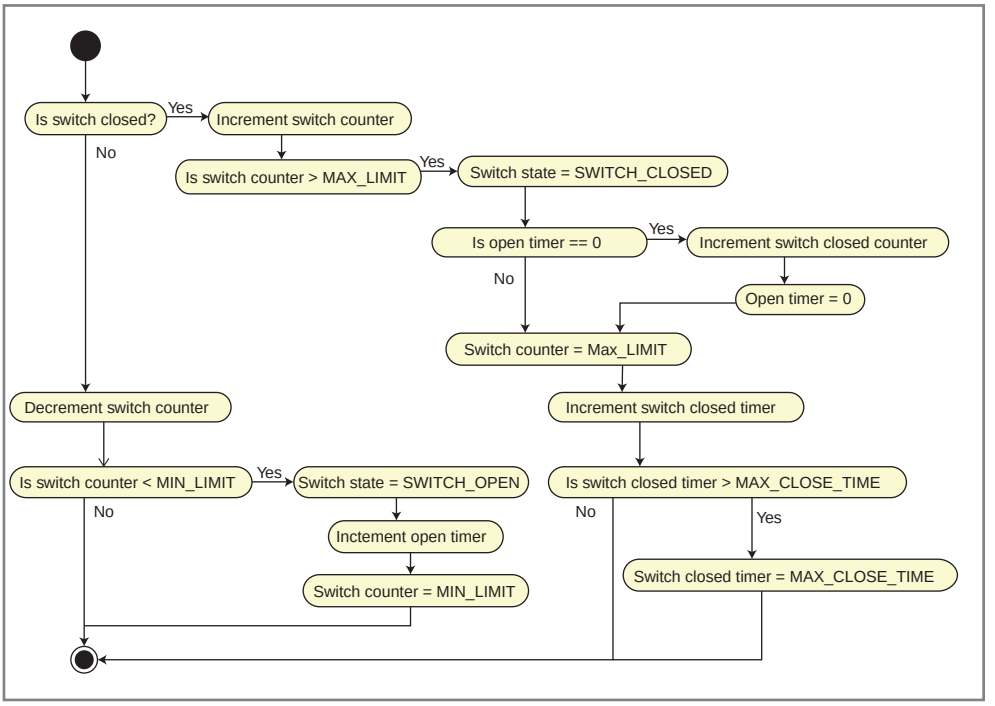


Figure 7—Switch counting

Listing 3—The code outside of the state machine

```
if ((LastState == SWITCH_CLOSED) &&  (State == SWITCH_OPEN)) {
        if (Switch OpenTimer > LIMIT) {
        SwitchClosed Counter is the number of closures.
        Process that fact(number of closures)
        Number of closures = 0        // reset that counter
             LastState = SWITCH_OPEN  // we're all done
}
```

ADDING SUCCESSION COUNTING

The customer would like one of the switches to count the number of times it is pressed rapidly in succession. So, two quick presses would return a value of two and three quick presses would return a value of three. It seems like a simple request, and you agree to do it.

The information you are gathering is changing quite a bit in this request. Take a look at Figure 6. The top line shows two switch closures and should report the value 2. The next line shows three switch closures and should report the value 3. How would you handle this request? What is the information being processed? Remember, you still need to debounce the switch.

To keep this simple, I would think that if you counted switch closures until you got a long switch opening, then you could report the results. The long switch opening is the key. Let's just say that after 1 s of no switch actions you'd be done and could report. Let's also say it wouldn't matter how long the switch was closed. So, you'd need to count each closing, and after a long switch opening, report the results. Take a look at the state diagram in Figure 7. You can see that each debounced closure increments a switch-closed counter. Then the code outside of the state machine would look something like what is shown in Listing 3.

MEASURING TIME

So far, this has been a hypothetical exercise for you. But I've been through it. And there's more. What if you needed to detect the distinctive ringing signals that are available on some phone systems? Normal ringing is 2 s ringing and 4 s quiet. And one special ringing pattern is 1 s ringing, 1 s quiet, 1 s ringing, and 3 s quiet. Both patterns add up to a 6-s overall period. This is a key point, because the timers in the phone system may be either fast or slow. You couldn't rely on just measuring the ringing time and checking it against a limit. You'd probably need to measure all the times and then do a percentage for each as compared to the total time period. I know, because I've been caught trying to just measure time. This design is looking for different information and processing it quite differently than the previous examples. Think about how you might approach the design of the state diagram for this problem.

Now your homework assignment is to detect long and short switch closures and then detect long and short pauses between these events. Sound crazy? Well, think of Morse code. But whatever you do, don't look up "Morse code detector." That would be cheating. (I bet there's even an app for that.) If you dig into the Morse code specification, you'll see that a dot and a dash and a space between letters and a space between words have relationships among them. I'll expect your designs to be handed in before the weekend. See you next time.

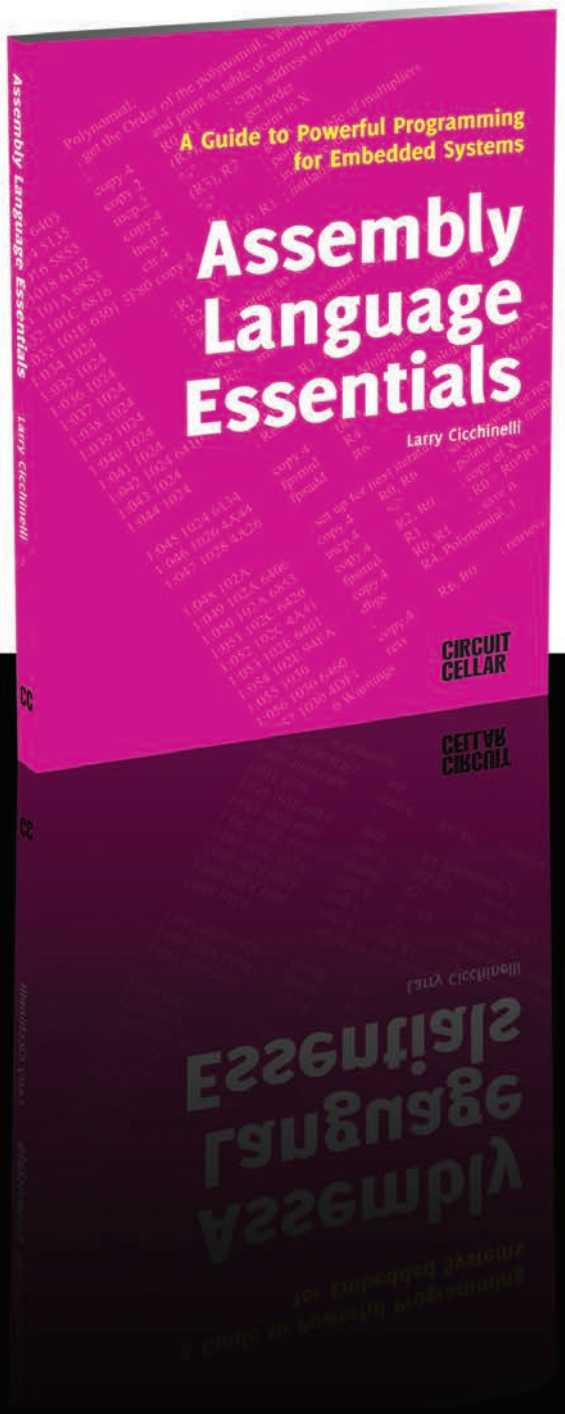
George Martin (gmm50@att.net) began his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and co-founded a design and manufacturing firm (www.embedded-designer.com). His designs typically include servo-motion control, graphical input and output, data acquisition,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

RESOURCE

Tyco Electronics Corp., "Relay Contact Life," 2000, http://relays.te.com/appnotes/app_pdfs/13c3236.pdf.

Assembly Language Essentials

Circuit Cellar's first book, ***Assembly Language Essentials***, is a matter-of-fact guide to Assembly that will introduce you to the most fundamental programming language of a processor.



Author Larry Cicchinelli provides readers with:

- An introduction to Assembly language & its functionality
- Essential terminology pertaining to higher-level programming languages & computer architecture
- Important algorithms that may be built into high-level languages — multiplication, division, and polynomial evaluation
- Overview of Interrupt Service Routines
- Free, downloadable Assembler program …

and more!

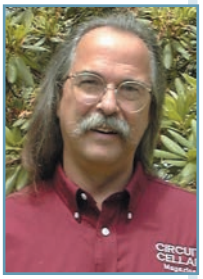
On Sale NOW!

$47.50

WWW.CIRCUITCELLAR.COM • WWW.CC-WEBSHOP.COM

CIRCUIT CELLAR
THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

by Jeff Bachiochi (USA)



Wireless Data Delivery

You can automate a system's data collection process to eliminate the need for manual data entry. This article describes how to use open wireless technology to deliver data wirelessly between a project and an application written to gather data from it.

I recently experienced a personal wake-up call regarding my high blood pressure. For months, I recorded my blood pressure readings morning, noon, and night by writing them into the little "day" blocks of a kitchen calendar. By comparing the data on a day-to-day basis, I noticed a gradual improving trend based on my diet and exercise. But I had to enter the data into a file before I could plot it. Graphing enables you to really picture your data—after all, "a picture is worth 1,000 data points."

The project I described in my last article, "Build an Automatic Blood Pressure Cuff" (*Circuit Cellar* 263, 2012), uses a small air pump to inflate a cuff to 180 mmHg. Then the deflating pressure is monitored to detect the systolic and diastolic pressure points. I look for the point where the heartbeat's pulse pressure overcomes the falling cuff pressure by looking for a pressure increase. By measuring the time between pressure increases, beats per minute (BPM) can be calculated.

The design included a serial port for debugging purposes and a LCD to display the test results. A single push button started the test cycle (it also served as an abort that deflated the cuff, if necessary). Besides handling the user interface, the microcontroller controlled the electric pump, the deflation solenoid,

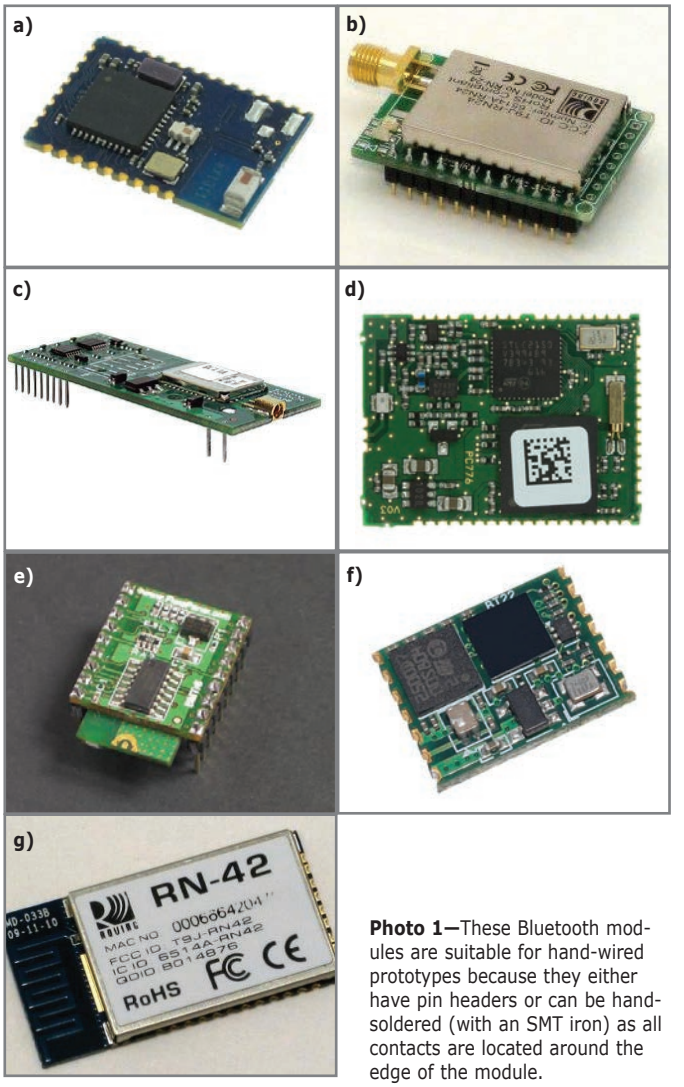


Photo 1—These Bluetooth modules are suitable for hand-wired prototypes because they either have pin headers or can be hand-soldered (with an SMT iron) as all contacts are located around the edge of the module.

and the analog pressure sensor. There is certainly enough horsepower left to add a real-time clock (RTC) and some sort of memory device (e.g., SD card) for logging purposes. This also means additional keys to enable entering names, and so forth. On the other hand, I can make this much simpler by unloading this burden onto my PC or even onto my Palm PDA. This would enable me to get rid of the display, which is one of the big items on the bill of materials (BOM).

The serial debugging port I was no longer using was perfect for adding a wireless Bluetooth module. One of the most useful purposes for Bluetooth is to eliminate a wired serial connection between a device and your PC, PDA, or even cell phone. I'll begin by looking at a few modules and how they might be used.

Photo 1 shows a number of candidates for prototyping use. I define a good candidate as a module that has unhidden connections. And by that, I mean all connections are accessible around the module's perimeter and not hidden somewhere beneath the module. There are a number of modules that use a ball-grid array (BGA) style connection pattern. While this is fine for normal surface-mount technology (SMT) assembly, it is impossible to make these connections with just a soldering iron, so I consider these bad candidates. Some modules have pins that will nicely plug into your circuit, however most are SMT. When I have a prototype PCB made, I often include the SMT pattern for a specific Bluetooth device in addition to a six-pin header.

Using a six-pin header enables me to swap in/out various models provided I use the same pinout to prepare them. Photo 2 shows a couple of modules I hand-wired to a six-pin socket strip. Note: These particular devices require 3.3 V. If your circuit uses 5 V, you will need to provide some level shifting, as some modules do not have 5-V tolerant inputs. The simplest connection uses four of the six signals: power, ground, TX, and RX. The remaining two signals can be RTS and CTS or, if handshaking is not needed, two alternate signals (e.g., reset and link). Using an alternate signal (e.g., link) enables your circuitry to detect when a BT connection has been made.

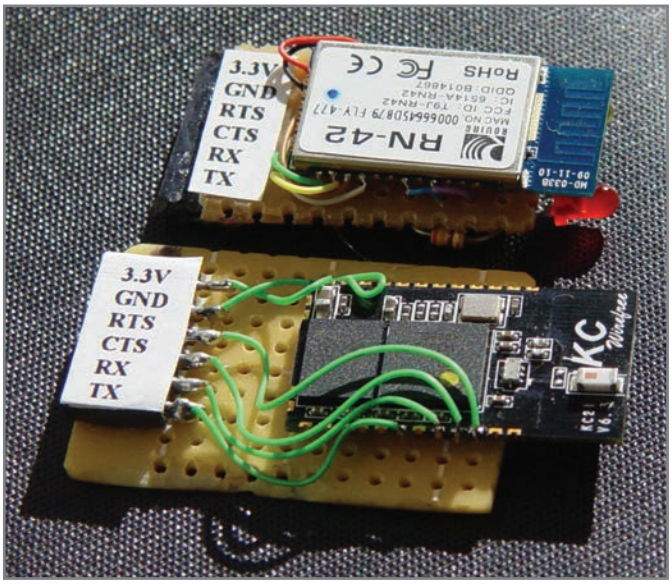


Photo 2—These modules are hand-wired to a six-pin socket, making them user friendly for your design. I also added some LEDs that indicate "link presence" and "activity."

PRAISE FOR THE AT COMMAND SET

Many Bluetooth modules use the AT command set (or something similar) to enable the user to configure the module. You may wish to set a particular baud rate, enable the module to answer with verbose responses, or even turn off a sign-on message. Since these modules are designed to pass data back and forth, they need to know the difference between data and some command you are sending them. There is usually an escape sequence of characters which is recognized as a signal that any following data is not data, but some configuration command. Each manufacturer has its own escape sequence that is chosen as the "least likely"' sequence of characters to be found in data transfer (which would inadvertently cause the module to switch into command mode). The standard sequence for the AT command set used with the Hayes modem is "~~~" (i.e., three tilde symbols in a row). While the idea of escaping into a configuration mode using a special sequence of characters still remains, the actual sequence and even the "command set" itself have become unique to each vendor (see Table 1). You may find it educational to review these before making a commitment to any particular manufacturer, as there may be idiosyncrasies that make one module a better fit for your projects.

Module	Escape sequence
Bluegiga	<esc><esc><esc>
KC-21	$$$
LinkMatik	<esc><esc><esc>
RN-42	~~~~~1 (2, 3, or 4)
SPBT2532	^#^$^%

Table 1—Vendor command sets

For two Bluetooth devices to communicate, they must be paired—swap MAC addresses—if authorization is required by either device. Each device has three forms of control to this operation. The Discovery parameter determines if a device will respond (with MAC) to another's call for "who's there?" The Pairable parameter controls whether a device will accept a request to pair with another device. The Connect parameter can disable any wireless activity. The pairing process may require some kind of authorization such as entering a personal identification number (PIN). Once completed, information about the pair can be stored to simplify future connecting. How does this work from the PC's point of view? Assuming your PC (or laptop) has Bluetooth built in or you've plugged in a USB Bluetooth dongle, there will be a Bluetooth icon in your task bar.

If it's not there, you can find it in the Control Panel's classic view. Clicking on this icon will open a window showing the Bluetooth devices you have already paired with this computer. Photo 3 shows that I have six devices paired on my system: my cell phone, PDA, three Bluetooth serial devices, and a Nintendo Wii controller. Note the Add Wireless Device button. Clicking on this button begins a discovery process that calls out to any Bluetooth devices in the area asking for information. If it finds a device that is not on its list, it displays it and enables you to choose and begin the pairing process. If a

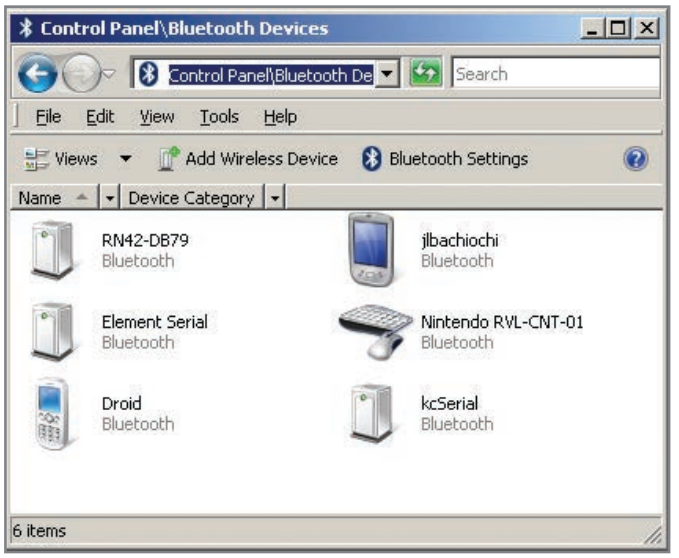


Photo 3—There are six Bluetooth devices paired with my PC. Each device has had its drivers installed, has been assigned a COM port, and is ready to use.

PIN is required, you can get this from the device's documentation. If you can't locate this, try "0000" or "1234," which seem to be two of the most widely used defaults. Once paired, any drivers needed are automatically installed. You will want to take note of the COM number that has been assigned to this pairing. You will need to choose this COM port whenever you try to access the device from within an application program. Here is what happens when I click on a terminal program application. I'm going to use RealTerm for this discussion. If I click on my KC Serial icon in the Bluetooth Devices window and then on the Services tab, I see that virtual serial port COM64 has been installed for this device.

PLUG & PLAY

I used a generic PCB left over from a past article to prototype last month's project. The PCB has an SMT layout for a KC Wirefree KC-21 Bluetooth module and a six-pin header that shows up in the schematic as TTL serial. I can use almost any Bluetooth module I can wire to a six-pin socket. With my KC-21 placed on the header, I can power up the project and *voila*! Wait. It's not working. Yes, there is a change I need to make to the default settings of this particular Bluetooth module. Its default data rate is 115,200 bps. How can I get this done without any custom circuitry? I used this six-pin header to connect to an external circuit, which has a MAX232 and DB9 on it. This level-shifting circuit enabled me to send debugging information to a terminal program running on my PC through a hardwired serial connection. The Bluetooth module could use this to talk to the PC through its TTL connection. But, I assume you don't have access to this and will need another way to alter its default settings. So I'll just let the KC-21 sit on the BPM project for the time being, while it currently has the wrong data rate. Since it is powered from the BPM, it will wirelessly connect to my PC.

If I start RealTerm on my PC and open COM64, it will connect with the module, provided the BPM project is powered on. RealTerm is the same application I used to communicate with the BPM project while connected through the hardwired serial port COM2. I had set up the UART on the BPM's microcontroller to run at 38,400 bps. So the Bluetooth module will not talk with the microcontroller's UART as they stand. I have two options: change the microcontroller's code so it runs at 115,200 bps or change the Bluetooth module to run at 38,400 bps. Note: This has nothing to do with the wireless transmission speed or my PC's baud rate. For argument's sake, I'll say the microcontroller code can't be changed and I'll have to change the default setting in the Bluetooth module. This is where the escape codes come into play. By sending the escape sequence from my PC, I can get control of the remote Bluetooth module. This code sequence must be sent as one continuous block. If the characters are typed into the terminal by hand, too much time will pass between characters for the sequence to be considered legal.

Photo 4 shows the RealTerm screen as it runs on my PC. The initial string of characters "~~~~~3" (in red) was typed in one at a time traveling through the ether to the remote KC-21 mounted on the BPM project. They were not recognized by the module as there was too much time between characters. I resent the ASCII escape string (in green) using the Send ASCII command. This time the remote answered back with <-> [RemoteMode] (in yellow). Now I can enter an AT command to change the data rate, "at configuart 38400." The remote answers with "<-> ConfigUartOk 38400-8-n-1." This tells me the remote has registered my request. I follow up with two more commands, "at messages d" and "at reset." Note that after I disabled messages, the remote no longer replied to the command. The reset command restarts the KC-21 Bluetooth module using my changed default parameters, and it is now ready to talk with the UART at 38,400 bps. Because of this reset, I must close and reopen the connection. And when I enter "?" the BPM project responds with the sign-on message and help commands.

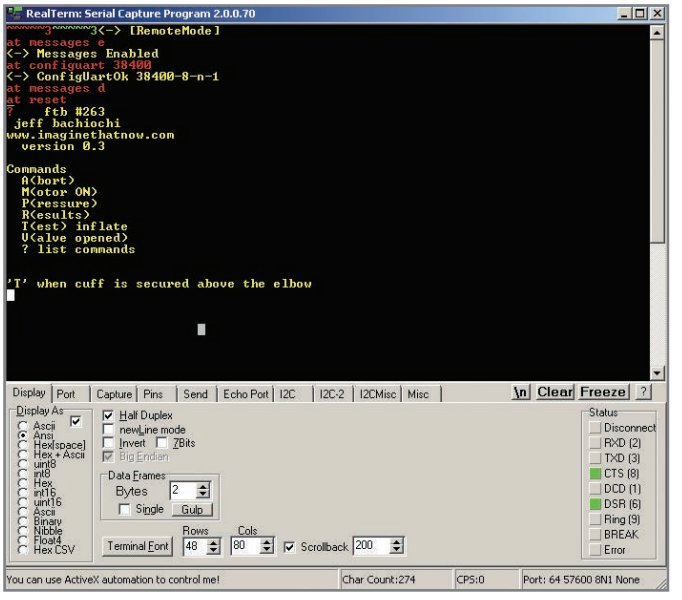


Photo 4—The virtual COM port assigned to my KC-21 Bluetooth module can be selected and connected through a Bluetooth dongle plugged into my PC. Note the first escape sequence (red) wasn't recognized. I had to send the sequence as a string and not individual characters (green), and then the remote module responded. After a few configuration commands, I saved the changes with a reset and, *voila*, I had communication with my project.

BPM APP

Now that I have a wireless connection to the project, I can develop an app to log data. This app must extract important data from the output of the BPM project sent via Bluetooth, display it (systolic, diastolic, and BPM), and enable it to be saved (Save button). The app must also initiate a test on the BPM project (Test button). The mean arterial pressure (MAP) wasn't part of the initial project, so its value isn't transmitted but it can be calculated from the systolic and diastolic pressures using the formula:

$$MAP = \frac{(2 \times \text{Diastolic} + \text{Systolic})}{3}$$

To enable multiple users to log individual data, I need a way to select a user and also take advantage of the PC's RTC so that each log entry is time-stamped. You can refer to the flowcharts in Figure 1 and Figure 2 to see how this application is constructed. Like many of my PC applications, I am using Shoptalk Systems's Liberty BASIC programming language because it's inexpensive and easy to use.

I didn't realize when I created the verbose messages to help in the debug process for last month's project they would be sufficient to support this wireless application. All the data I want to log and the control of the test process were already part of the project. I just needed to add the Bluetooth link. This application scans for available comports and presents the choices. When you choose the corresponding COM (as displayed earlier, when the BT module was paired) the link is established. While the serial port profile (SPP) does not support modem signals that could be used to signal when a BT module has established a connection, there are other outputs on most modules that indicate this. These can be substituted for the CTS and RTS lines on the six-pin connector to enable the microcontroller to be more aware, as the situation warrants.

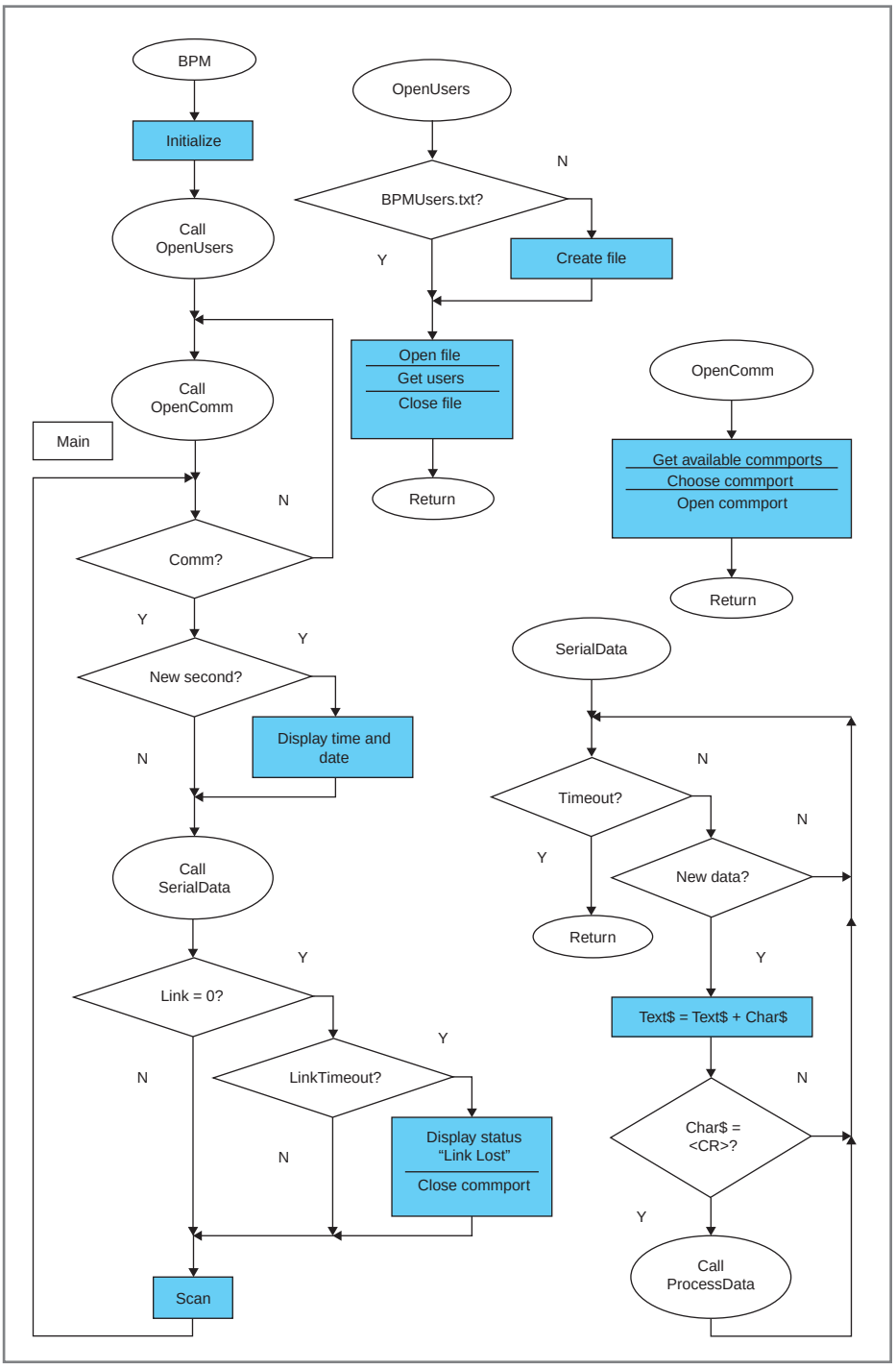


Figure 1—This activity occurs when the BPM application starts.

In this instance, any messages are simply passed on to the PC. And, although I could make use of any of the commands I had set up, only one is needed. Sending the letter "T" will either begin or abort a test sequence. This means I could eliminate the LCD and the push button and most likely use a much smaller microcontroller. Having a local Abort button isn't a bad idea though.

When this application begins, it must check for a text file that holds the user names. Initially, there isn't one. So if the file can't be located, one is created to include two names: User 1 and User 2. This provides the application a couple possibilities to get started. You can choose to use one of these or add your name to the list by clicking on the User tab (see Photo 5). When the Test button is clicked, all values are set to zero, all boxes are unchecked, and a T is sent to begin the test. As messages are received by the application's serial input routine, these strings are searched for those of importance. You can refer back to Fig-

25th Anniversary SPECIAL EDITION

Circuit Cellar Archive on USB

Celebrate 25 years of Engineering Excellence with this Special Edition Circuit Cellar Archive packed with all these incredible features...*and a free gift!*

- Special Anniversary USB drive or our traditional CC GOLD USB drive
  Your choice!
- A USB memory upgrade from 16GB to a whopping 32GB!
- All Circuit Cellar magazine issues in print through date of purchase
- Article Code
- All the details from our previous Design Challenges*
- Two years of ELEKTOR Magazine Issues in PDF format (2010-2011)
- Two years of audioXpress Magazine Issues in PDF format (2010-2011)
- *Free Gift!* Circuit Cellar 25th anniversary hat

Save $25.00 for a limited time! Order today at:
www.cc-webshop.com

* Design Challenge add-ons:
ATMEL AVR Design Contest 2006, WIZnet iEthernet Design Contest 2007, Microchip 16-Bit Embedded Control, Contest 2007 Texas Instruments Design Stellaris Contest 2010, WIZnet iMCU Design Contest 2010

ure 1 and Figure 2 to see which messages are acted upon. When a test is "finished" without an error, clicking the Save button will append the time and date, along with the four reading to a file beginning with whatever user name has been selected. A separate file is kept for each user. Each user's comma-delimited list will easily import into any graphing program (e.g., Excel).

THE GOLDEN RING

There was a time when we could ride on a glorious merry-go-round and those situated on the outer-most colorful steeds could reach out and pluck a ring as they passed a vending machine of sorts. The lucky jockeys who ended up with the golden rings (yes, these were actually painted gold) would earn a free ride. Attaining the golden ring has come to mean "achieving the ultimate." I have some thoughts on what the golden ring might mean to the future of data gathering and analysis.

At this stage, I am forced to take active steps in performing important functions. When you are trying to adjust your weight for instance, you must actively track your intake and exercise, and make the necessary adjustments to reach your goals. This might include meal planning, calorie counting, workout regiments, and analysis and review of progress. There is a certain direct feedback you get from being actively involved, but there is far too much work that isn't directly associated with the actual goal.

What's needed are smarter tools that can work together. This could begin with smart meal planning (making choices that fit within a specific caloric intake); suggesting an exercise program based on meal choices, time and equipment available; or even planning around weather conditions. Monitoring equipment, from the high-tech watch you wear (which might contain a personal ID) to your bathroom scale, should gather data and automatically transfer it to a central repository (possibly your PC).

Applications that run in the system tray or gadgets in your sidebar should be aware of prospective data, gathering it as it becomes available and ready to present it on command. Just as the industrial revolution gave us a bit of freedom from work, smart tools could take us to the next level, releasing us from being slaves to our own technology. We already realized that preventive (and closely monitored) healthcare could reduce the need for emergency care. I am in awe at the diagnostic devices available today thanks to flourishing technology. And, while it may

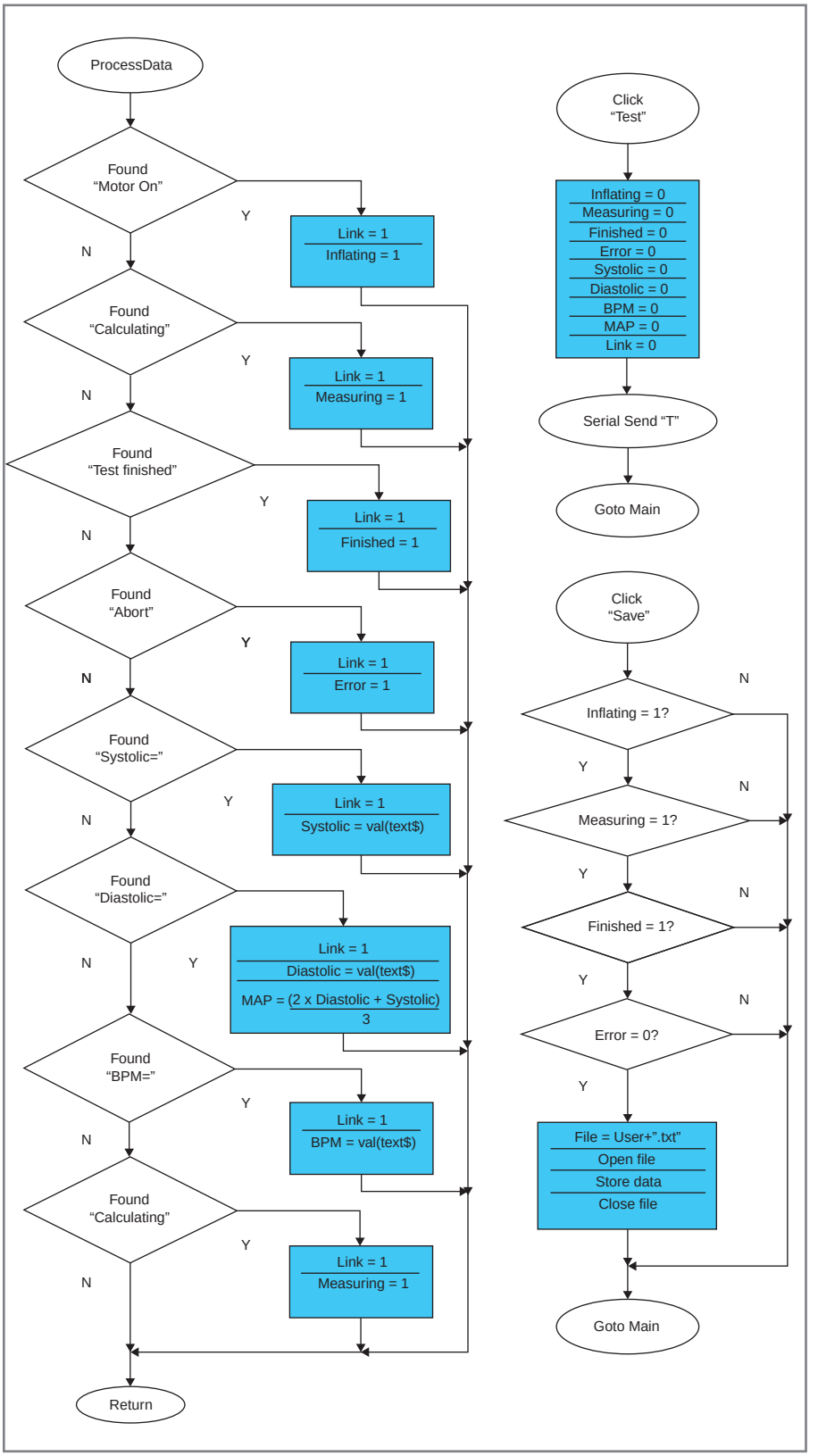


Figure 2—An example of what happens when an event occurs (i.e., when serial data is received or a button is pushed)

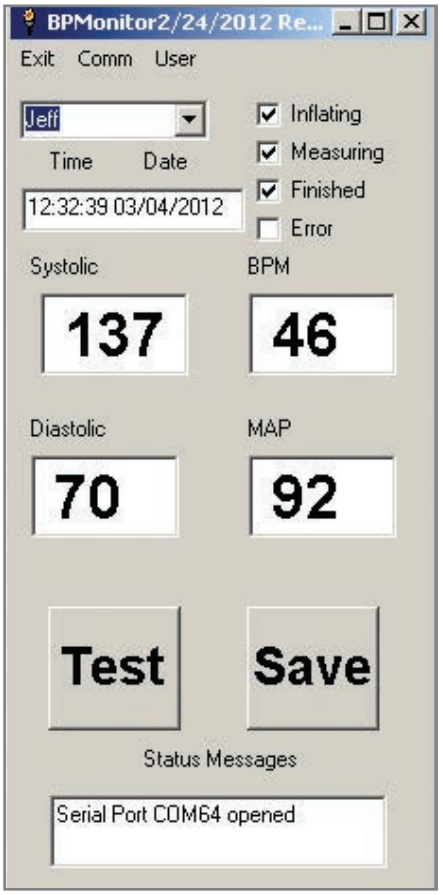


Photo 5—This application, written in Shoptalk Systems's Liberty BASIC, gathers data from my June BPM project. The Bluetooth link now delivers data wirelessly between that project and this application.

be a few years before we have portable MRIs in our medicine cabinet, I believe this day will come. We can either sit back and wait or take on the challenge.

A HEALTH-HAPPY CLOUD

The "cloud" is a hot topic these days. And Microsoft is offering a cloud for preserving health-related data. Microsoft HealthVault is a place where health information can be stored, retrieved, and shared. HealthVault is both an application and device. For instance, if you use a specific pharmacy for prescription fulfillment, you can sign up for the prescription management (application) through your HealthVault account. If you own a Polar exercise monitor (device) its data could go to your HealthVault account. Microsoft also has an SDK available so you can develop a compatible interface and help others achieve their health and fitness goals.

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.*

RESOURCES

J. Bachiochi, "Build an Automatic Blood Pressure Cuff," *Circuit Cellar* 263, 2012.

Bluetooth SIG, Inc., www.bluetooth.com

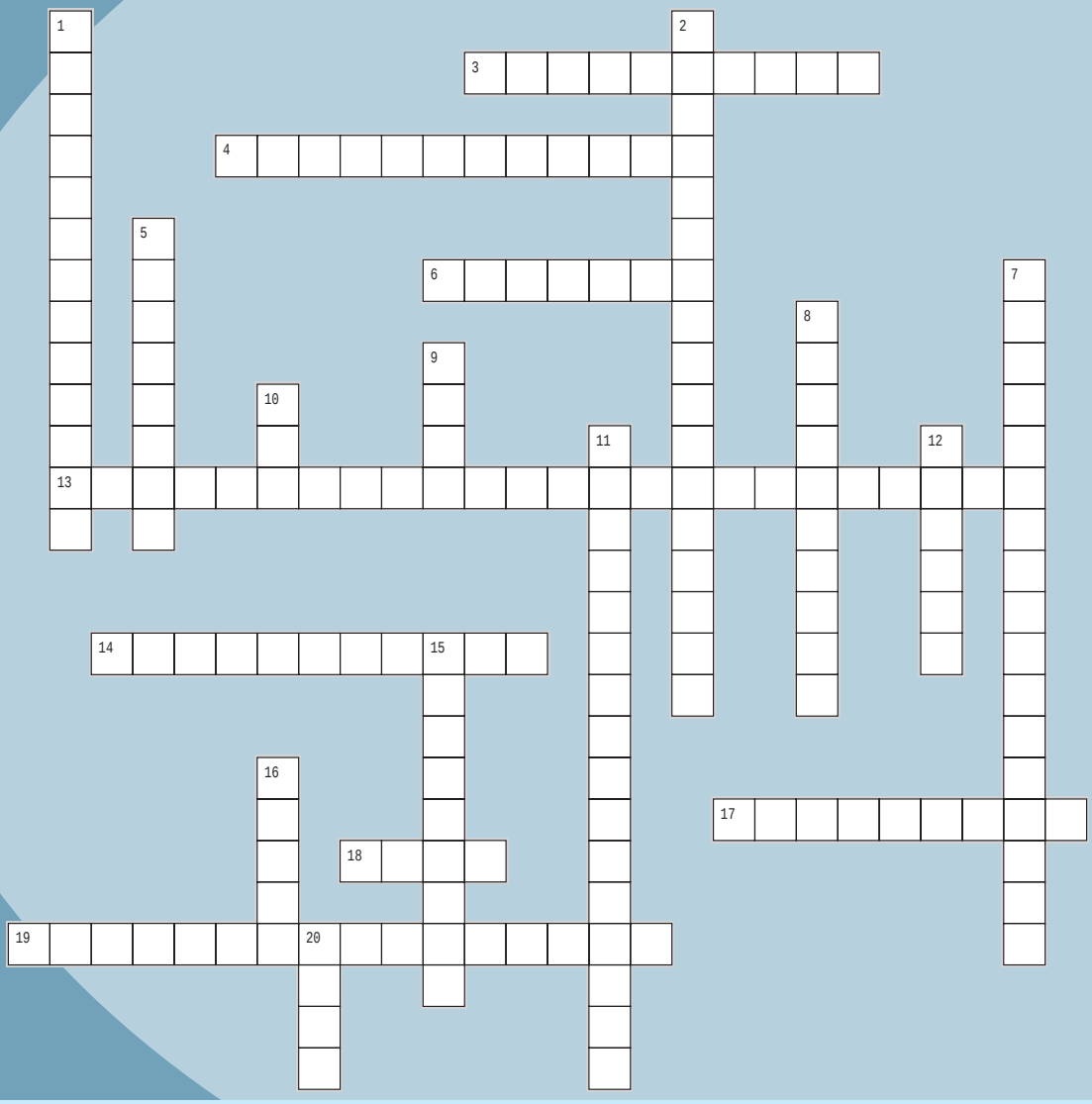
SOURCES

KC-21 Bluetooth
KC Wirefree Corp. | www.kcwirefree.com

HealthVault storage platform
Microsoft Corp. | www.microsoft.com/en-us/healthvault

Liberty BASIC programming software for Windows
Shoptalk Systems | www.libertybasic.com

CROSSWORD



Down

1. A phenomenon that occurs when a vehicle sounding a siren approaches, passes, and recedes from an observer [two words]
2. A device that produces electronic signals [two words]
5. Equals 1/10,000,000,000 m
7. ΔT = 0 [two words]
8. A device that compares two voltages or currents and switches its output to indicate which is larger
9. Organization formed in 1934 by bridge engineer David Steinman
10. Common cause of electronic data corruption and subject of George Novacek's December 2011 *Circuit Cellar* article; acronym [two words]
11. Occurs when crystals acquire a charge after being compressed, twisted, or distorted (e.g., quartz)
12. American electrical engineer (1937–1991) and IC pioneer
15. Circuitry that regulates or provides power to a light source [two words]
16. Symbolized by the 10th letter of the alphabet
20. Unscramble the following: IETEORGSEPSMNYMRLTIAEAT (Hint: It's an acronym)

Across

3. Occurs when an atom or molecule gains either a positive or negative charge
4. Matt Oppenheim's article, "Audio-Enhanced Touch Sensors" (*Circuit Cellar*, May 2012), said one of the stumbling blocks of using this for data collection is that it will try to recharge itself whenever you connect it to a USB port [two words]
6. *Circuit Cellar* interviewee who participated in Motorola's IEEE-802 MAC subcommittee on token-passing access control methods [two words]
13. A method of keeping the world in sync [three words]
14. *Circuit Cellar* published his book about a commonly used computer programming language in 2010
17. A synonym for "circuit cellar"
18. Si plate
19. National Semiconductor's LM385 series is an example of an adjustable one [three words]

The answers will be available in the next issue.

IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

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

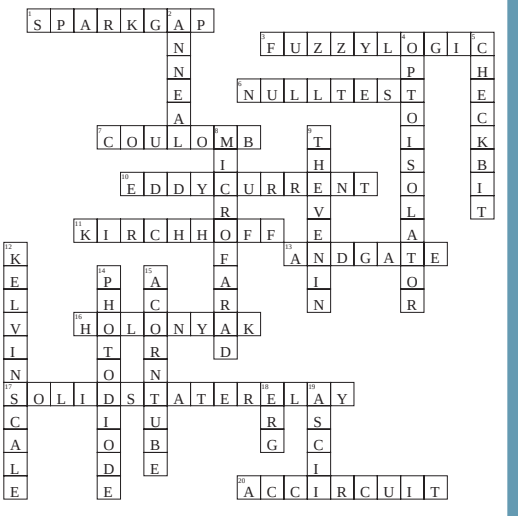
CROSSWORD ANSWERS from Issue 263

Across

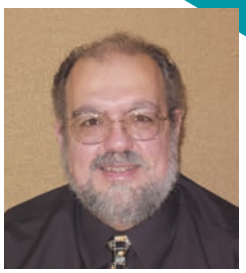
1. **SPARKGAP**—A space in an otherwise closed electric circuit [two words]
3. **FUZZYLOGIC**—Began in 1965 with Lotfi Zadeh's proposal of a certain type of set theory [two words]
6. **NULLTEST**—Cancels a device's signal input by negative feedback from its output [two words]
7. **COULOMB**—One of these is equal to 6.28 × 1018 electrons
10. **EDDYCURRENT**—a.k.a. Foucault currents [two words]
11. **KIRCHHOFF**—Created a law of thermochemistry
13. **ANDGATE**—Relies on truth and logic [two words]
16. **HOLONYAK**—Invented the LED while working at General Electric in the 1960s
17. **SOLIDSTATERELAY**—Doesn't rely on movement or contact to switch electric circuits [three words]
20. **ACCIRCUIT**—A circuit with a current that goes one way and then another [two words]

Down

2. **ANNEAL**—Runs hot and cold
4. **OPTOISOLATOR**—Changes electrical signals to light and back into electrical signals
5. **CHECKBIT**—Adds a bit to make things even or odd [two words]
8. **MICROFARAD**—1,000,000 pF
9. **THEVENIN**—French engineer (1857–1926); augmented Ohm's law
12. **KELVINSCALE**—Temperature scale where 0° = absolute zero [two words]
14. **PHOTODIODE**—Turns light into current or voltage
15. **ACORNTUBE**—A small tube used at very high frequencies [two words]
18. **ERG**—Causes 1 cm of movement
19. **ASCII**—A character-encoding system used to represent text



PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

Fix It or Toss It?

No prophetic diatribes or deep philosophical insights this month. Just the musings of an old guy who apparently doesn't know when to throw in the towel. Let me explain.

I have a friend with a couple LCD monitors he purchased about two years ago. Perhaps due to continuous duty operation (only interrupted by automatic "Sleep mode"), both were now exhibiting some flakiness, particularly when powering up from "sleep." More importantly, if power was completely shut down, as in a power failure, they wouldn't come back on at all without manual intervention. He asked if they could be repaired or must they be replaced.

Since I remembered something about a few manufacturers who'd had a bunch of motherboard problems a while back due to bad electrolytic capacitors, I suspected a power supply problem. Of course, agreeing to look into the problem and figuring out how to get inside the monitors was a whole different issue. Practically all of today's electronics are not meant to be opened or serviced internally at all. Fortunately, my sledgehammer disassembly techniques weren't so bad that I couldn't reassemble them. In the process, I found several bulging and leaking capacitors on the power supply board. After replacing the capacitors, the monitors came right up with no problems.

Power supplies just seem to have it out for me. Recently, I had a wireless router stop working and, after a little diagnosing, I determined that its power supply (an external wall-wart) had failed. While hardly worth my time, I was curious, so I cracked open the sealed case to see just how complicated it was. Sure enough, replacing one scorched electrolytic capacitor and gluing the case back together put me back in business.

All this got me thinking about the relative value of various electronic devices. What is the replace/repair decision line? These $200 high-tech electronic LCD monitors failed because of $3 worth of old-tech components that I was fortunately able to fix. It took time to do the repair that has some value, but it also takes time to shop for and purchase a replacement. There must be better monitors these days for the same price. Should I have told him to toss them and use the opportunity to upgrade?

It's interesting to consider the type of person who repairs stuff like this (being an EE with a fully equipped lab doesn't hurt either). I mean, I do it primarily because I like knowing how things work. Okay, so I'm getting a little carried away after fixing a couple burnt capacitors, but there's still an incredible sense of satisfaction in being able to put something back together and having it work. Since I was a kid, dissecting circuits and equipment helped me understand the design choices that were made, and my curiosity naturally lead me to engineering.

Now, I recognize that people like me who repair their own electronics for curiosity or adventure are very much in the minority. So, what about the average person with a failed piece of $200 electronics? For them, the only goal is getting the functionality back as soon as possible. Do they go to a repair service where it takes longer and involves a couple trips? Worse yet, some things just can't be repaired, and the bad news then is having both the repair "inspection" cost and the replacement cost. I'm guessing that in 99% of typical cases, the no-brainer decision is to toss the failed unit and buy a new one—without ever giving me a chance to tear it apart and play with it.

Let's face it. Taking modern equipment apart to make even simple repairs is next to impossible. The manufacturers use every trick in the design book to minimize the cost of the goods. This means leaving out features that might make end-user repair easier. Cases that snap together (once)—or worse, are heat-welded together—are cheaper than cases with screws or latches. Most board electronics are custom-labeled surface mount devices, everything uses custom connectors, and the short cabling between boards has no slack to swing out subassemblies for access, and so forth. You couldn't even fit a scope probe inside most of this stuff if you tried. Sure, some manufacturers do still put component reference designators in the silkscreen, but I suspect it's so they can repair subassemblies on their production line before final assembly, not make it easier for me to poke around.

Anyway, like I said, there's no prophetic conclusion to be drawn from all of this. I fix stuff because I enjoy the challenge and I usually learn something from it. Even if I can't repair the item, I usually keep some of the useful components and/or subassemblies for experimenter one-off projects or proof-of-concept prototypes. You never know when something in the junk box might prove useful.

steve.ciarcia@circuitcellar.com

FTDI Chip

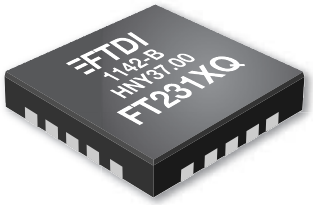
ARE YOU READY FOR THE X-CHIP FACTOR?

INTRODUCING THE X-CHIP USB BRIDGE FAMILY

EXtensive Interface Selection
UART, FIFO, SPI, I2C, FT1248

EXtended Temperature Range

Battery Charger DeteXion



EXpandable Clocking
INTERNAL CLOCK GENERATION
EXTERNAL CLOCK OUT

EXceptional Support

Additional features:

- Royalty free, validated USB driver support
  Windows, Android, Linux & Mac OS
- Low active power: 8mA
- Internal MTP memory for flexibility
- Integrated 3.3V level converter

us.sales@ftdichip.com
www.ftdichip.com


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