Q&A: Embedded Systems Education
LOCATION: Spain
PAGE: 28

SECURITY: Hardware-Accelerated Encryption
LOCATION: United States
PAGE: 48

EE TIPS: Project Configuration Control
LOCATION: Canada
PAGE: 58

CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

SEPTEMBER 2012
ISSUE 266

DATA ACQUISITION

Build an MCU-Based Data Recording System

Miniature Accelerometers for Motion-Sensing Applications

Software & Design File Organization

PLUS

Flowcharting Simplified

// Use a Flowcharting IDE to Write Code
// Utilize Project Options
// Choose an MCU
// Select the Analog Frequency
// Determine Frequency & PWM Outputs

$9.00US $10.00CAN
09>
0 74470 75349 0
www.circuitcellar.com

TASK MANAGER

The Ubiquitous Importance of Data

Regardless of your area of embedded design or programming expertise, you have one thing in common with every electronics designer, programmer, and engineering student across the globe: almost everything you do relates to data. Each workday, you busy yourself with acquiring data, transmitting it, repackaging it, compressing it, securing it, sharing it, storing it, analyzing it, converting it, deleting it, decoding it, quantifying it, graphing it, and more. I could go on, but I won't. The idea is clear: manipulating and controlling data in its many forms is essential to everything you do.

The ubiquitous importance of data is what makes *Circuit Cellar*'s Data Acquisition issue one of the most popular each year. And since you're always seeking innovative ways to obtain, secure, and transmit data, we consider it our duty to deliver you a wide variety of content on these topics. This month, we present both data acquisition system designs and tips relating to control and data management.

On page 18, Brian Beard explains how he planned and built a microcontroller-based environmental data logger. The system can sense and record relative light intensity, barometric pressure, relative humidity, and more.

Data acquisition has been an important theme for engineering instructor Miguel Sánchez, who since 2005 has published six articles in *Circuit Cellar* about projects such as a digital video recorder (*Circuit Cellar* 174), "teleporting" serial communications via the 'Net (*Circuit Cellar* 193), and a creative DIY image-processing system (*Circuit Cellar* 263). An informative interview with Miguel begins on page 28.

Turn to page 38 for an informative article about how to build a compact acceleration data acquisition system. Mark Csele covers everything you need to know from basic physics to system design to acceleration testing.

In "Hardware-Accelerated Encryption," Patrick Schaumont describes a hardware accelerator for data encryption (p. 48). He details the advanced encryption standard (AES) and encourages you to consider working with an FPGA.

Are you now ready to start a new data acquisition project? If so, read George Novacek's article "Project Configuration Control" (p. 58), George Martin's article "Software & Design File Organization" (p. 62), and Jeff Bachiochi's article "Flowcharting Made Simple" (p. 66) before hitting your workbench. You'll find their tips on project organization, planning, and implementation useful and immediately applicable.

Lastly, on behalf of the entire Circuit Cellar/Elektor team, I congratulate the winners of the DesignSpark chipKIT Challenge. Turn to page 32 to learn about Dean Boman's First Prize-winning energy-monitoring system, as well as the other exceptional projects that placed at the top. You can review the complete projects (abstracts, photos, schematic, and code) for all the winning entries at www.circuitcellar.com/contests/chipkit2012.

cj@circuitcellar.com

C. Abate

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, I²C, 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

The "Photo-Pal" Flash Trigger System (Part 1), by Richard Lord

FAT Cache for Fast SD Card Access, by Kerry Imming

Build an MCU-Based RF Door Controller, by Scott Weber

Math Routines for Embedded Systems, by Stuart Oliver

COLUMNS

Mechanical Gyroscope Replacement (Part 1), by Jeff Bachiochi

An Introduction to IIR Digital Filters, by Robert Lacoste

MOSFET Tester, by Ed Nisley

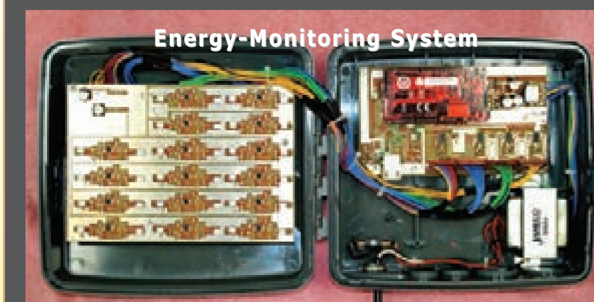
Managing Project Risk, by George Novacek

Concurrency in Embedded Systems (Part 3), by Bob Japenga

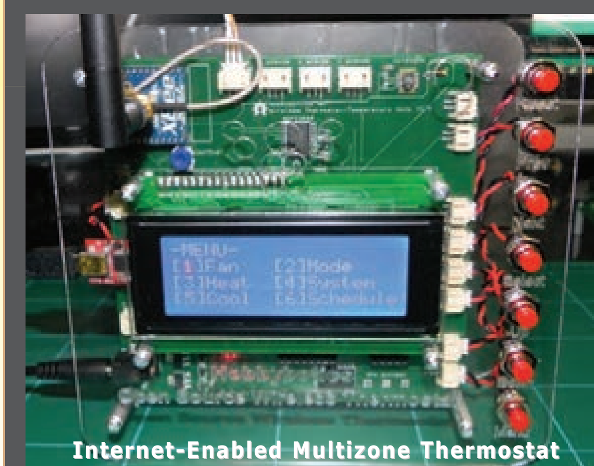
INSIDE ISSUE 266

September 2012 • Data Acquisition

Innovative chipKIT Design Projects p. 32



Energy-Monitoring System



Internet-Enabled Multizone Thermostat

Build a Digital Accelerometer p. 38

Get Started with Flowcode p. 66

THE TEAM

THE NETWORK

audioXpress

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

elektor

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 266 September 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 274920 members in 83 countries.

Not a member yet?

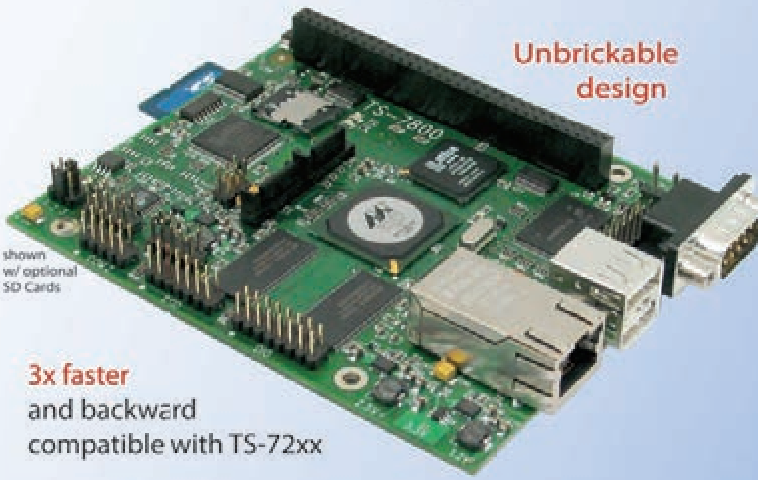
Sign up at www.circuitcellar.com

SUPPORTING COMPANIES

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

$229 qty 100

$269 qty 1

- Low power - 4W@5V
- 128MB DDR RAM
- 512MB high-speed (17MB/sec) onboard Flash
- 12K LUT customizable FPGA
- Internal PCI Bus, PC/104 connector
- 2 host USB 2.0 480 Mbps
- Gigabit ethernet
- 2 SD sockets
- 10 serial ports
- 110 GPIO
- 5 ADC (10-bit)
- 2 SATA ports
- Sleep mode uses 200 microamps
- Boots Linux 2.6 in 0.7 seconds
- Linux 2.6 and Debian by default

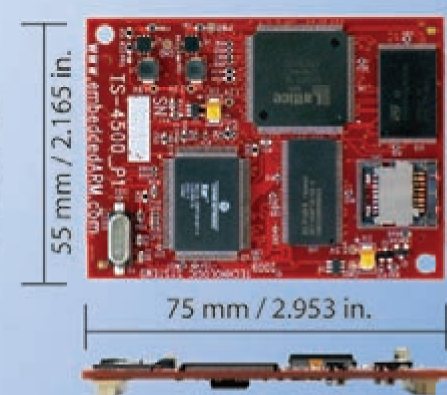
TS-SOCKET Macrocontrollers
Jump Start Your Embedded System Design

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

series starts at $92 qty 100

$139 qty 1



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

- 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

HIGH-RESOLUTION, WEATHER-RESISTANT ULTRASONIC RANGE FINDER

The **HRXL-MaxSonar-WR** sensor is designed for level-sensing applications such as snow, water, tanks, and bins. With a stable 1-mm resolution, the sensor features target-size, temperature, and supply-voltage compensation. The sensor weighs 50 grams and operates from 2.7 to 5.5 V with less than 3-mA current draw.

The HRXL-MaxSonar-WR sensor line is capable of compensating for target-size differences. If an object is large enough to be detected, the sensor will report the same distance regardless of target size. The sensor's noise rejection characteristics will work in the presence of other electrical or acoustic noise sources with higher amplitudes. Most range readings are accurately reported within 5 mm.

The HRXL-MaxSonar-WR's internal temperature sensor enables accurate speed-of-sound temperature compensation. An external temperature sensor can accurately measure the air temperature in the sensor's environment. The HR-MaxTemp, which is the external temperature sensor, is an available option.

The HRXL-MaxSonar-WR comes with the same easy-to-use outputs and standard pin configuration as previous MaxSonar products. In addition to the three standard sensor outputs (RS-232 serial, analog voltage, and pulse width), TTL serial output is now available on select models.

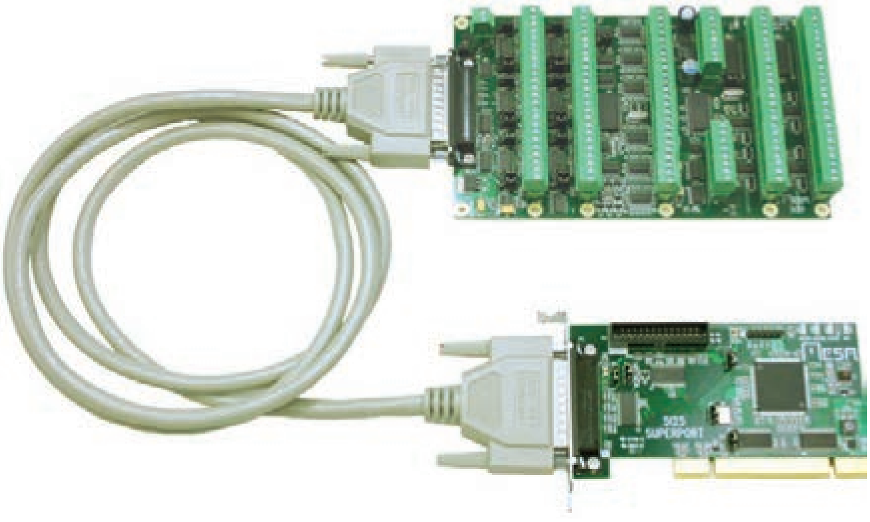
The HRXL-MaxSonar-WR costs **$109.95**.

MaxBotix, Inc.
www.maxbotix.com

ANALOG CONTROL SYSTEM FOR CNC

The **MESA 7I77** card set is a FPGA-based, six-axis analog control system for CNC, industrial automation retrofits, and OEM systems. Six axes of analog outputs and encoder inputs can be used for up to five axis machines, plus spindle control. Encoder inputs can be individually programmable for TTL or differential mode. The PCI or PCIE host interface provides real-time access to the motion hardware.

The control system features motion-related I/O, 32 digital inputs, and 16 digital outputs. These digital I/O points are isolated from the system ground and can use 5-to-32-V I/O voltage. Inputs have a threshold of 0.5 the I/O voltage for high noise immunity. Outputs can supply 300 mA each and are short-circuit protected. I/O can be expanded to more than 400 I/O points with real-time access or up to 12 motion axes. The 7I77 card set is fully supported by LinuxCNC. All FPGA firmware is open source and easily customized to support new and different functions.



The 7I77 set with the PCI host adapter costs **$173** in 100-unit quantities. The 7I77 set with the PCIE host adapter costs **$187** in 100-unit quantities.

Mesa Electronics
www.mesanet.com

DIFFERENTIAL AMPLIFIER DRIVES 18-BIT ADCs

The **LTC6362** is a low-power, fully differential amplifier capable of driving high-precision 16- and 18-bit SAR ADCs at a 1-mA supply current. The amplifier, which features 200-µV max input offset voltage and 3.9 nV/√Hz input-referred noise, is well suited for precision industrial and data acquisition applications.

The LTC6362 has an output common-mode pin with a 0.5-to-4.5-V range and 18-bit settling time of 550 ns with an 8-V_{PP} output step. It is ideal for driving ADCs such as the LTC2379-18 in multiplexed input and control-loop applications. The LTC2379-18 features digital gain compression, which sets its fullscale input range to 10 to 90% of the reference voltage. Combined with the rail-to-rail output stage of the LTC6362, this feature eliminates the need for a negative supply rail, simplifying the circuit and reducing power consumption.

The LTC6362's flexible architecture can convert single-ended, DC-coupled, ground-referenced signals to differential or DC-level shift differential input signals. The amplifier's low-input bias current, low-offset voltage, and rail-to-rail inputs also enable it to be used in a high-impedance configuration to interface directly to sensors early in the signal chain.

The LTC6362 is available in MSOP-8 and 3 mm × 3 mm DFN packages with guaranteed specifications over 0° to 70°C, –40° to 85°C, and –40° to 125°C temperature ranges.

The LTC6362 costs **$1.59** each in 1,000-piece quantities.

Linear Technology Corp.
www.linear.com

COMPACT & PROGRAMMABLE TRANSCEIVER MODULE

The **MRF-900-TCMP** is a low-cost, embedded, self-contained, fully tested, surface-mountable RF wireless module based on the Silicon Labs Si1000 RFIC. The module, which is suitable for any embedded wireless application, is small (0.43" × 0.991"), low power (less than 0.1-µA sleep), and high performance (141-dB link budget).

The MRF-900-TCMP has 100-mW transmit power, –121-dBm receiver sensitivity, and can communicate over distances more than 1 mile in open-field LOS conditions, and more than 1,000' indoors. The module supports OOK/FSK/GFSK modulation and can operate at data rates up to 125 Kbps.

Internally, the module features a high-performance 8051 microcontroller with 64 KB of flash and 4 KB of RAM. Peripherals include: 10-bit analog-to-digital converter, PWM, timer/counter, UART, SPI, I²C, voltage comparators, and programmable current reference. The integrated microcontroller's power performance requires 160 µA per megahertz of operating clock speed.

The MRF-900-TCMP easily handles battery-powered applications. In Sleep mode, the module draws 10 nA with brownout detection disabled and 50 nA with brownout enabled. At full power, in Transmit mode with the microcontroller operating at full speed, the module draws less than 90 mA. In Receive mode, the current is reduced to 22 mA. Using software, the transmitter power is adjustable from 1 to 20 dBm. The module's operating voltage range is 1.8 to 3.6 V.

The MRF-900-TCMP is compatible with the Digital Six Laboratories OpenRF free open-source wireless protocol stack. OpenRF supports point-to-point, point-to-multipoint, and multipoint-to-multipoint networking using single-channel or FHSS modulation. A battery-powered mesh networking mode is currently in development.

The MRF-900-TCMP module also supports the Java-programmable Wireless Kontroller (JwiK) BriK Java run-time environment, which is a free open-source Java VM designed for small microcontrollers used in wireless applications.

The module is available in 433.92-, 868-to-870-, and 902-to-928-MHz versions. A complete line of complimentary antennas and connectors is also available.

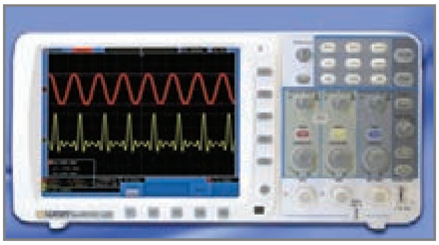
The MRF-900-TCMP costs **$9.90** in quantities up to 1,000 units.

Digital Six Laboratories, LLC
www.d6labs.com

LOW-COST, 300-MHz OSCILLOSCOPE

The **SDS9302** is a low-cost, two-channel, versatile oscilloscope that can be fitted with an optional battery, making it well suited for field or isolated operation.

The oscilloscope's Autoscale feature, which can be used for circuit probing, automatically adjusts the vertical gain, the horizontal time base, or both together. As the probe is moved from point to point on a circuit board, the display auto-adjusts for best trace presentation. It is similar to the AutoSet feature, but instead of being a one-time function,



Autoscale is "hands free" and only active until it is turned off. The FFT function provides an instant display of the signal's frequency spectrum under test.

The SDS9302 can automatically measure and display frequency and peak-to-peak/RMS/mean values. Its cursors are adjustable for individual readings. The oscilloscope's built-in self-calibration capability improves measurement accuracy.

The SDS9302's features include: onboard storage, manual cursor measurements, up to 19 automatic measurements (including frequency), high-speed screen updates, storage for up to 15 waveforms and set-up parameters, a USB interface with PC software, 400-V (DC + AV peak) maximum input, an optional rechargeable battery pack, multi-language capabilities, deep memory, external video-capable trigger, autoscaling, a large 8" full-color TFT LCD, external monitor/projector output, XY mode, auto-set, averaging, math functions, USB output, waveform storage, pass/fail output, and FFT functionality for frequency spectrum display.

The oscilloscope weighs less than 4 lb and features a large 8" 800 × 600 pixel color TFT LCD.

Contact Saelig for pricing.

Saelig Co., Inc.
www.saelig.com

PROGRAMMABLE SWITCHING DC POWER SUPPLIES

The **1685B**, **1687B**, and **1688B** are bench-switching mode DC power supplies. Each model is available in a different configuration of variable output voltage and current. The 1685B offers 1–60 V, 0–5 A; the 1687B offers 1–36 V, 0–10 A; and the 1688B offers 1–18 V, 0–20 A.

All models feature constant voltage (CV) and constant current (CC) modes, bright three-digit LED displays, and rotary encoder control knobs for coarse and fine voltage and current settings. The models also offer built-in overvoltage protection (OVP), overtemperature protection (OTP), and overload protection (OLP) circuitry. With their compact sizes, these switching power supplies increase power density and provide high efficiencies of 80% or more. These DC power supplies are well suited for university labs, R&D, service, and production testing.

Similar to B&K Precision's high-power 1900 series switching supplies, the new models have a front panel auxiliary output and three user-defined voltage and current presets for quick recall of common test parameters. An analog remote control terminal is accessible on the rear. It enables users to connect external variable DC voltage sources or variable resistors to remotely control the output voltage and current, or remotely enable and disable the power supply output.

In addition to the analog remote control function, the 1685B, 1687B, and 1688B models feature a USB interface for remote PC connectivity. Users can easily control their instrument using the included PC software or by sending remote programming commands.

The 1685B, 1687B, and 1688B models cost **$339** each.

B&K Precision Corp.
www.bkprecision.com

LCD FEATURES EASY-TO-USE PROGRAMMING

The **ezLCD-304** is an LCD that provides a graphical user interface (GUI) with an all-in-one modular design. The design combines a 4.3" color LCD, touchscreen, control electronics, memory, I/O, and a mounting bracket, with an easy-to use USB programmable firmware environment. The EarthSEMPL programming language provides users with quick customization of macros, graphical objects, fonts, and images, which reduces the product design cycle.

The ezLCD-304 features 272 × 480 resolution, 65,536 colors, 350-nit brightness, a 500:1 contrast ratio, and a four-wire resistive touchscreen. The LCD's intelligent control module is highlighted by a 16-bit microcontroller, 4 MB of flash memory, USB 2.0, and TTL serial interfaces. The ezLCD-304 operates at a 3.3-V supply voltage, draws less than 100 mA, and provides a –20°-to-70°C operating temperature range. The LCD costs less than vacuum fluorescent displays, STN passive matrix displays, or complex graphical LCD products.

A compatible development kit provides a comprehensive, easy-to-use platform for those designing the ezLCD-304 into new or existing product applications. The kit includes the ezLCD-304, a development board with RS-232, I²C, and RS-485 interfaces, cables, a screwdriver, jumper shunts, and a 3.7-V lithium battery.

The ezLCD-304 costs **$149**. The development kit costs **$249**.

Earth Computer Technologies, Inc.
http://store.earthlcd.com

NPN


Detention Punishment:
Write "I will not play video games in class"
300 Times
10 X = 0
20 Print "I will not play video games in class"
30 X = X + 1
40 IF X < 300 GOTO 20
50 END
President Bush

1-Mb, 80-Mbps SRAMs DEVICES

Microchip Technology has added four new large-density, high-speed devices to its serial SRAM portfolio. The devices use the quad-SPI (i.e., SQI) protocol to achieve speeds of up to 80 Mbps. This provides the zero write-cycle times with almost immediate data movement needed for data-intensive functions (e.g., offloading graphics, data buffering, data logging, displays, math, audio, and video).

The **23LCV512** and **23LCV1024** are two additional devices that provide options for nonvolatile, unlimited-endurance RAM via battery backup. These devices feature fast 40-Mbps dual-SPI (SDI) throughput and low active and sleep currents, making them well suited for applications including meters, black boxes, and other data recorders that require unlimited endurance or instantaneous writes along with nonvolatile storage.

All six devices from the serial SRAM family are available in eight-pin SOIC, TSSOP, and PDIP packages with 512-Kb and 1-Mb density options.

The 23A1024 and 23LC1024 are available now for sampling and volume production. The 23A512 and 23LC512 are expected to be available for sampling and volume production in October.

The four volatile devices cost **$1.16** each in 10,000-unit quantities. The 23LCV512 and 23LCV1024 devices cost **$1.32** each in 10,000-unit quantities.

Microchip Technology, Inc.
www.microchip.com

NPN

Test Your EQ

Problem 1—What's the key difference between infinite impulse response (IIR) and finite impulse response (FIR) digital filters?

Problem 2—Does the fact that the finite resolution of digital arithmetic effectively truncates the impulse response of an IIR filter turn it into an FIR filter?

Problem 3—The following pseudocode represents an implementation of a single-pole, low-pass IIR filter, using 16-bit input and output values and a 24-bit internal accumulator and a filter coefficient of 1/256:

```
# The 32-bit accumulator holds 16 integer
# and 16 fractional bits
$acc = 0x00000000;

# The input value is a 16-bit integer.
$input = 0xFFFF;

# Offset used for rounding the accumulator
# to 24 bits.
$offset = 0x80;

while (1) {
  # acc = (255*acc + input)/256
  $acc -= ($acc >> 8);
  $acc += ($input << 8) + $offset;
  # limit acc to 24 bits
  $acc &= 0xFFFFFF00;
  # output is integer part of acc
  $output = $acc >> 16;
}
```

An implementor of this filter complained that "the output never reaches 0xFFFF." What was the flaw in his reasoning?

Problem 4—The original implementor's solution was to change the $offset value to 0xFF. Why did this work?

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

Contributed by David Tweed

MEMBER PROFILE: Richard H. Lord

Member Status: Richard said he has subscribed to *Circuit Cellar* for at least 14 years, maybe longer.

Technical Interests: Richard's interests include photography, model railroading, and microcontroller projects.

Most Recent Embedded Tech-Related Purchase: Richard's most recent purchase was a Microchip Technology dsPIC30F4013 digital signal controller.

Current and Recent Projects: Richard is working on a Microchip PIC16F886-based multipurpose front panel interface controller.

Member Name: Richard H. Lord

Location: Durham, NH

Education: BS Electrical Engineering 1969, MS Biomedical Engineering, 1971

Occupation: Retired electronics hardware design engineer

Thoughts on the Future of Embedded Technology: "With the ready availability of prepackaged 32-bit processor modules, it's easy to forget there are many applications where 8-bit controllers are more appropriate", Richard said. He continued by saying he gets a lot of enjoyment from the challenge of working within the capabilities and constraints of the smaller microcontrollers.

HuMANDATA®

FPGA / CPLD Boards from JAPAN

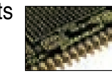
SAVING COST=TIME with readily available FPGA boards

- 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

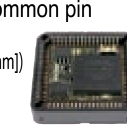
FPGA/CPLD Stamp Module PLCC68 Series

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



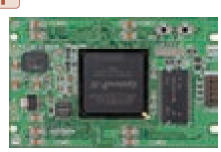
ALTERA FPGA Board

Cyclone IV E F780 FPGA board

ACM-204 series NEW

Cyclone IV E | SDRAM

EP4CE30F29C8N
EP4CE40F29C8N
EP4CE115F29C8N
Credit card size (86 x 54 mm)
RoHS compliant

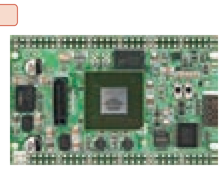


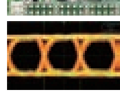
Arria II GX F572 FPGA board

ACM-025 series

Arria II GX | DDR2

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



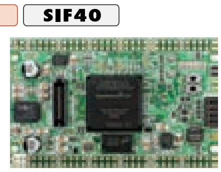


CycloneIV GX F484 FPGA board

ACM-024 series

Cyclone IV GX | DDR2 | SIF40

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

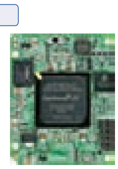


Cyclone IV E F484 FPGA board

ACM-107 series

Cyclone IV E | MRAM

EP4CE55F23C8N
EP4CE75F23C8N
EP4CE115F23C8N
Compact size (43 x 54 mm)
RoHS compliant



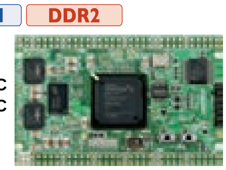
XILINX FPGA Board

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

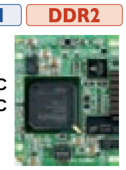


Spartan-6 FGG484 FPGA board

XCM-110/110Z series

Spartan-6 | MRAM | DDR2

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

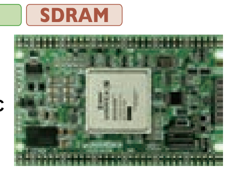


Virtex-5 FFG676 FPGA board

XCM-011 series

Virtex-5 | FRAM | SDRAM

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

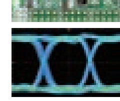


Virtex-5 LXT FFG665 FPGA board

XCM-017 series

Virtex-5 | SDRAM | RocketIO | SIF40

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

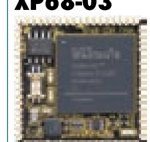


XILINX PLCC68 Series

Spartan-6 PLCC68 FPGA Module

XP68-03 NEW

Spartan-6 | PLCC 68

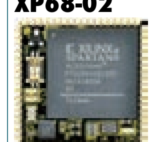


XC6SLX45-2CSG324C
3.3V single power supply operation
On-board oscillator, 50MHz
RoHS compliant

Spartan-3AN PLCC68 FPGA Module

XP68-02

Spartan-3AN | PLCC 68



XC3S200AN-4FTG256C
FPGA internal configuration ROM
Two User LEDs
RoHS compliant

ALTERA PLCC68 Series

Cyclone III PLCC68 FPGA Module

AP68-04 NEW

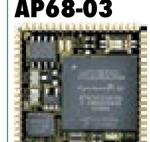
Cyclone III | PLCC 68

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

Cyclone III PLCC68 FPGA Module

AP68-03

Cyclone III | PLCC 68



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

MAX V PLCC68 CPLD Module

AP68-02

MAX V | PLCC 68

5M570ZF256C5N
External Clock inputs
On-board Voltage regulator
RoHS compliant

See all our products, A/D D/A conversion board, boards with USB chip from FTDI and accessories at : www.hdl.co.jp/CC/

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

FEATURE ARTICLE

by Brian Beard (USA)

MCU-Based Environmental Data Logger

You can plan and construct a flexible environmental data logger. This article details a battery-powered data logger design featuring sensors for temperature, barometric pressure, relative humidity, light, and switches. The logger uses any terminal emulation program for control and data transfer.

It all began two years ago, after my employer moved into a new office building with zoned heating. My zone's thermostat is in my neighbor's office, not mine. His office stays a comfortable 74ºF, while mine gets as low as 57ºF. The building custodian's computer printout showed my zone was 74ºF. Without proof my room was cold, he could do nothing. That was the motivation for building my Environmental Data Logger (EDL), which is shown in Photo 1. As I designed it, my EDL became more than just a temperature recorder. It became a sophisticated solution for a variety of environmental data logging applications.

For set up and data download, the EDL connects to a host PC via a serial port or USB-to-serial adapter. You can use any type of terminal software to configure the EDL to record any or all of its inputs, enabling recording sessions from hours to weeks in length. A recording session is the time it takes the EDL to fill its data storage memory with sampled data. During a recording session, the active inputs are sampled at the end of each sample period. A sample period can be as short as 4 s or as long as an hour.

As you read through this article, refer to the sidebar that details the EDL's specifications. I list important information about system power, memory, and inputs.

MCU & CLOCKS

The EDL is based on Microchip Technology's PIC16F873 microcontroller, which has many on-chip peripherals and special features—most of which are used in this application. The PIC16F873 is shown as U1 in Figure 1. The five-pin header J1 is the in-circuit serial programming (ICSP) connector used to program the PIC16F873.

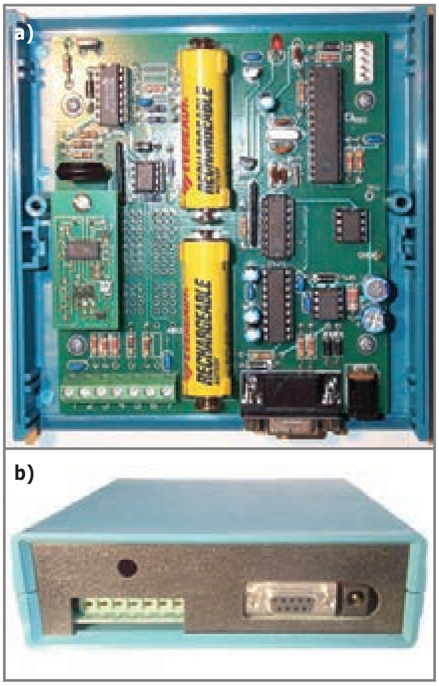


Photo 1a—This is the EDL's circuit board. **b—**This is the back of the EDL.

The PIC16F873 uses two crystal time bases. The 32.768-kHz crystal is connected to Timer 1, a 16-bit counter that generates an interrupt tick every 2 s. This 0.5-Hz interrupt is the EDL's real-time clock (RTC). Timer 1 and the Timer 1 interrupt operate in both Sleep and Normal modes.

The PIC16F873 is in Sleep mode most of the time. It emerges from Sleep mode in response to interrupts from Timer 1 or RB0. When the microcontroller emerges from Sleep mode, the 3.6864-MHz crystal oscillator starts. Program execution begins at the instruction following the Sleep instruction.

SERIAL PORTS

The PIC16F873 has two serial interface controllers. The first is the universal synchronous/asynchronous receiver transmitter (USART). The EDL and host PC communicate via an RS-232 COM port. The USART uses 1 start bit, 8 data bits,

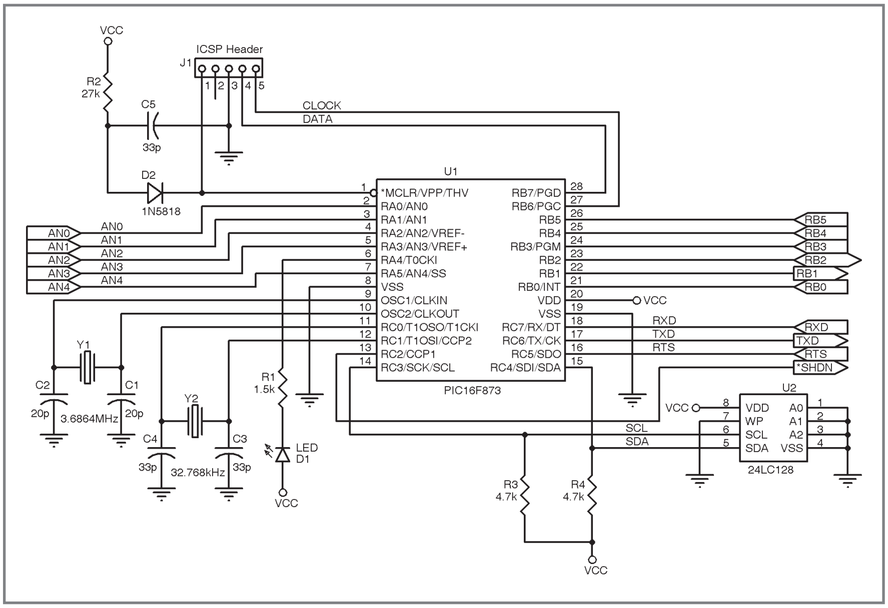


Figure 1—This is the EDL's PIC16F873 and memory circuitry.

So, even if the EDL's batteries die, you can still retrieve the data collected up to that point. The EEPROM in the EDL is in two parts. There are 128 bytes in the PIC16F873 and 16,384 bytes in Microchip's 24LC128 Serial EEPROM. The PIC's EEPROM is rated at 100,000 R/W cycles while the 24LC128 is rated for 1 million cycles. If the EDL was used for one recording session each day, the PIC's EEPROM would last more than 273 years, which is more than enough for most of us. The PIC's EEPROM stores the recording session's start time, sample period, mode setting, and active input flags.

The 24LC128 stores the sampled data from the analog sensors and digital inputs. Each sample occupies 2 bytes, so 8,000 samples can be stored before the memory is full. The amount of time it takes to fill the memory is the maximum recording time, which is a function of the number of inputs sampled and the sample rate. Consider the following examples. If seven inputs are sampled every 4 s, memory will fill in 1 h and 18 min. If four inputs are sampled every 30 min., memory will fill in 42 days and 16 h.

and 1 stop bit. As shown in Figure 2, the USART's transmit (TXD) and receive (RXD) pins connect to a Maxim Integrated Products MAX3222 RS-232 interface chip. A logic zero on the shutdown control pin (SHDN) turns off the charge pumps and disables the transmitters reducing the current drain to approximately 1 µA. The receivers remain active during shutdown, so the RTS signal from the host PC can still be monitored. RTS is sent back to the host PC as CTS, so CTS always echoes RTS.

The second serial controller is the master synchronous serial port (MSSP), which is used in the I²C Serial Bus Master mode to communicate with the I²C EEPROM. In concept, the I²C bus is a simple two-wire bus. The master controls all transfers and supplies the synchronous clock on the SCL line. Data is transferred on the serial data line, SDA. SCL can run as fast as 400 kHz but, because of the EDL's oscillator frequency and power considerations, SCL is set at 92.16 kHz.

Six possible events can take place on the I²C bus: start, restart, stop, read, ACK/NACK, or write. A small subroutine handles each I²C event. Transactions over the I²C bus consist of a series of calls to these subroutines.

EEPROM DATA MEMORY

EEPROM is an ideal storage medium for battery-powered devices because it retains data even when power is removed.

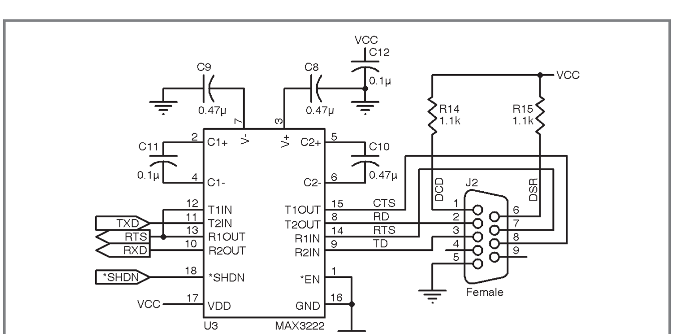


Figure 2—Take a look at the EDL's RS-232 interface circuitry.

POWER CIRCUITRY

The EDL's power circuitry is shown in Figure 3. Power can come from two sources. The first source is the two internal AA batteries, which can be standard alkaline or rechargeable NiCd cells. The second source is an external DC supply, which can be connected at J3.

Normal alkaline battery voltage is 1.5 V, so two batteries provide a maximum of 3 V. NiCd batteries have a 1.2-V cell voltage, so two of them only provide 2.4 V. Because the EDL circuitry is designed to function at a constant 5-V supply, the variable battery voltage must be converted to 5 V. This is done with a Maxim MAX756 step-up DC-to-DC converter. U6 can reliably start with positive voltage inputs

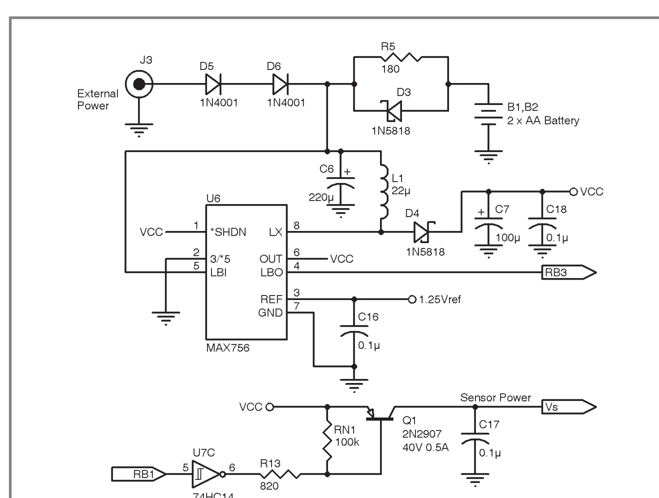


Figure 3—This is the EDL's power interface and control circuitry.

SPECIFICATIONS

Case

PacTec CM5-125 or Simco 150X5

Dimensions:

- 5.08" wide × 5.25" deep × 1.5" high
- 129.03 mm wide × 133.34 mm deep × 38.1 mm high

Power Requirements

Internal:

- Two AA batteries (alkaline or NiCd)

External:

- Coaxial power plug, center +, 5.5-mm OD, 2.1-mm ID
- Any DC voltage between 3.5 and 6 V at 50 mA, 4.5 to 6 V to charge internal NiCd batteries

Indicator LED

Off: normal operation

On steady: memory full

On flashing: low battery

Serial Interface

RS-232 at 9,600 bps on female, nine-pin, D connector

Serial data format:

- 8 data bits, no parity, 1 stop bit (8N1)

Memory

4 KB of flash (program) memory

16,512 bytes of EEPROM (data) memory

Analog Inputs

Temperature sensor: accuracy better than 2% of full scale

Light sensor: relative light intensity

Humidity sensor option:

- Relative humidity (RH) range: 20 to 90%
- Accuracy within ±2%

Barometric pressure (BP) sensor option: uses one external analog input

- BP range: 15 to 115 kPa (2.2 to 16.7 psi)
- Accuracy within ±1.5%

External: two available, one may be used for BP

- Hardware-configurable gain/filter options

Digital (Switch) Inputs

One input where rising edges are counted during each sample period

Two inputs that are read at the end of each sample period

from 1.8 to 5 V. Once it has started, the 5-V output can be maintained with an input as low as 0.8 V. This broad input range, plus isolation diodes D5 and D6, means the external voltage supply can be anything between 3.5 and 6 V. If NiCd batteries are used, they can be trickle-charged by the external source via resistor R5. To ensure the NiCds are charged to their full voltage, the external source should be 4.5 to 6 V. If alkaline batteries are used, R5 should be omitted.

U6 monitors the input voltage at Pin 5, low-battery input (LBI). When LBI goes below 1.25 V, low battery output (LBO) goes low. The microcontroller checks LBO every time it wakes up. After LBO reads low four consecutive times, the EDL switches to Idle mode, which ends sensor input recording. A low-battery condition is signaled by the indicator LED (D1) flashing at 0.5 Hz.

EDL Mode	$V_{BATTERY}$ = 2.5 V		$V_{BATTERY}$ = 5 V	
Waking every 2 s	1.96 mA	4.9 mW	0.91 mA	4.5 mW
Awake, RS-232 on	25.3 mA	63 mW	11.6 mA	58 mW
Awake, RS-232 active	29.1 mA	72 mW	13.3 mA	66 mW

Table 1—These are the results when using a 2.5- or 5-V battery eliminator instead of AA batteries.

POWER MANAGEMENT

With any battery-operated device, power management is an important consideration. The EDL uses several techniques to reduce current drain and extend battery life. In the first technique, the PIC16F873 microcontroller spends most of the time in Sleep mode. This reduces U1's current consumption to approximately 33 µA. In the second technique, the MAX3222 RS-232 interface chip is shut down unless an RS-232 cable is connected to J2. This reduces U3's current consumption to approximately 10 µA. In the third technique, power is applied to the sensors (V_S) only when it is time to acquire data. In the fourth technique, the indicator LED remains off while data is being sampled. It only turns on to indicate conditions that have stopped data sampling.

The data in Table 1 was obtained using a 2.5- or 5-V battery eliminator in place of the AA batteries. As the table shows, the normal current drain at typical battery voltage (2.5 V) is almost 2 mA. The EDL was tested with a pair of new Rayovac alkaline AA batteries with a 2,400-mAh at a 21ºC rated capacity. With a 2-mA current drain, these batteries should last approximately 1,200 h (i.e., 2,400/2). In reality, the batteries only lasted 1,032 h, which is still 43 days. During those 43 days, the EDL was connected to the host PC (RS-232 active) several times for data downloads and reconfiguring. This significantly increased the batteries' average current drain. Use of an external power supply while the EDL is connected to the host PC will greatly increase battery life.

ANALOG-TO-DIGITAL CONVERTER

The PIC16F873 has a five-channel, 10-bit ADC. The ADC is configured so its positive reference is V_{CC} and its negative reference is ground. If the ADC's 10-bit output is X, the corresponding input voltage is:

$$V_{IN} = \left(\frac{X}{1,024}\right) V_{CC}$$

This makes the resolution of the ADC:

$$\left(\frac{1}{1,024}\right)(5\ V) = 4.88\ mV$$

To reduce digital noise, Microchip suggests putting the microcontroller in Sleep mode while the ADC does its conversion. This technique has one big disadvantage. Interrupts other than the end of conversion (e.g., Timer 1 and RB0) can abort the analog to digital conversion. I wrote some test programs to determine which mode and how much averaging provided the best performance. A low-impedance constant voltage source was connected to the ADC input. The standard deviation, which indicates the amount of noise, is shown in Table 2. For a single sample, it's clear that Sleep mode is better. However, averaging

Samples per average	Mode	Averages taken (n)	Standard deviation
1	Normal	2,150	7.592
16	Normal	2,150	1.107
1	Sleep	2,150	4.892
4	Sleep	2,150	1.909
8	Sleep	2,150	1.86

Table 2—You can see the standard deviation in ADC bits from a low-impedance, constant-voltage source, indicating the amount of noise.

ADC Input	Signal
AN0	Temperature
AN1	External input AIN1
AN2	Barometric pressure or external input AIN2
AN3	Relative humidity or external input AIN3
AN4	Light intensity

Table 3—This table shows the ADC's inputs.

16 samples in Normal mode reduces the noise by a factor of four from the single sample in Sleep mode and enables simple interrupt handling. The EDL uses a subroutine that averages 16 sequential samples from a selected ADC channel. The ADC's inputs are listed in Table 3.

TEMPERATURE SENSOR

The EDL uses an Analog Devices AD22100 temperature sensor, which is a three-terminal (TO-92), 5-V supply, ratiometric voltage output sensor with typical accuracy better than 0.5% of fullscale. The AD22100 has a sensitivity of 22.5 mV per degree Centigrade as shown by its transfer function:

$$V_O = \left(\frac{V_{CC}}{5\text{ V}}\right)\left[1.375\text{ V} + \left(\frac{22.5\text{ mV}}{°\text{C}}\right)T\right]$$

The minimum temperature resolution is determined by the ADC's minimum resolution:

$$\left(\frac{4.88\text{ mV}}{\left(\frac{22.5\text{ mV}}{°\text{C}}\right)}\right) = 0.217°\text{C} = 0.391°\text{F}$$

Setting the voltage out of the AD22100 equal to the voltage into the ADC, then solving for temperature, results in T = 0.2172X – 61.11, where T is in degrees Centigrade and X is the ADC value.

LIGHT SENSOR CIRCUIT

The light sensor circuit provides an indication of relative light intensity. Measuring absolute light intensity is difficult even in the best of situations. In any environment, light intensity depends on the light's source, the time of day, shadows of moving objects (e.g., clouds and people) and which way the sensor is pointed. With so many uncontrollable variables, I decided a relative measurement was best.

The EDL uses a cadmium sulfide (CdS) photocell (RP1), which is essentially a light-sensitive resistor (see Figure 4). Its resistance varies over three orders of magnitude, from 1 MΩ in total darkness to 1 kΩ in bright light. RP1 and R6 form a voltage divider whose output follows light intensity. Op-amp U5A is a unity-gain voltage buffer with a high-impedance input that doesn't load the voltage divider. The voltage divider's output is shown by the solid blue circles in Figure 5. The problem with the voltage divider's output is that it goes above and below the light intensity's normal range for human vision. The light sensor's dynamic range should match the range of human vision. This is the curve's center from approximately 8 to 500 kΩ. The output should go from 0 V at 500 kΩ to 5 V at 8 kΩ.

Diode D7 and resistor R7 reduce U5A's output by the diode's forward voltage drop. This brings the voltage close to zero when RP1 = 500 kΩ. To bring the output up to 5 V at 8 kΩ, the voltage is amplified by a factor of 1.3. This is done by U5B, R8, and R9. As shown by the red squares in Figure 5, this voltage is closer to the ideal range.

BAROMETRIC PRESSURE

BP is the pressure exerted by the earth's atmosphere at a particular place and time. BP can vary with temperature, altitude, and weather conditions. BP is an absolute pressure measurement (i.e., its zero-pressure reference is a perfect vacuum). The Pascal (Pa), which is equal to one Newton of force per square meter, is the basic international unit for pressure measurement. A table summarizing

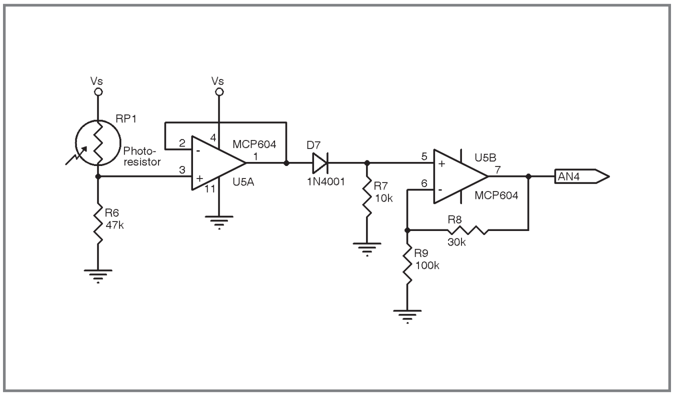


Figure 4—RP1 in the EDL's light sensor circuitry is a CdS photocell.

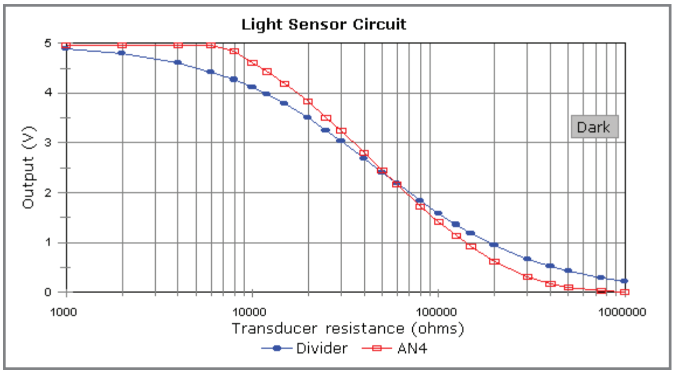
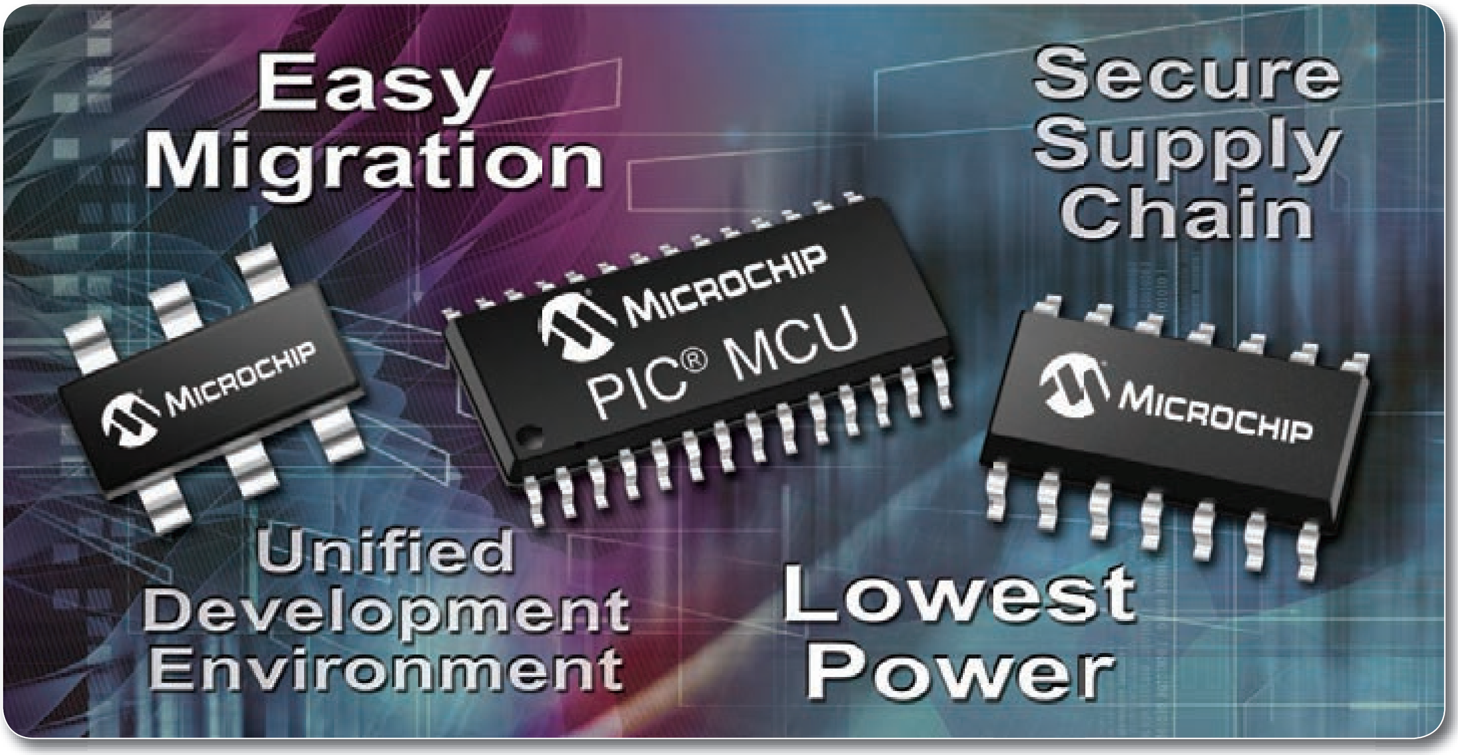


Figure 5—As you can see, the light sensor circuit output is a function of sensor resistance.

The Most Popular 8-bit Microcontrollers

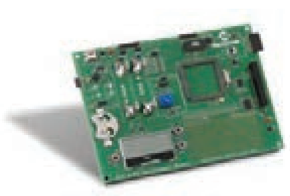


See what makes Microchip the most popular choice for embedded designers:

- Broad portfolio of more than 325 8-bit PIC® microcontrollers
- Easy migration with pin and code compatibility
- Industry's lowest active and sleep power consumption
- Integrated peripherals for USB, CAN and Ethernet with free software
- Interface to the world with LCD drivers and capacitive touch
- Continuous innovation with 70 new 8-bit MCUs in the last two years
- MPLAB® IDE is free, and supports ALL of Microchip's microcontrollers
- Low-cost development tools help speed up prototyping efforts
- Comprehensive technical documentation, app notes and code examples
- World-class 24/7 technical support and training

GET STARTED IN 3 EASY STEPS

1. Purchase the XLP 8-bit Development Board
2. Download free MPLAB® IDE
3. Order samples and start designing!
   www.microchip.com/8bit



XLP 8-bit Development Board (DM240313)

MICROCHIP

Microcontrollers • Digital Signal Controllers • Analog • Memory • Wireless

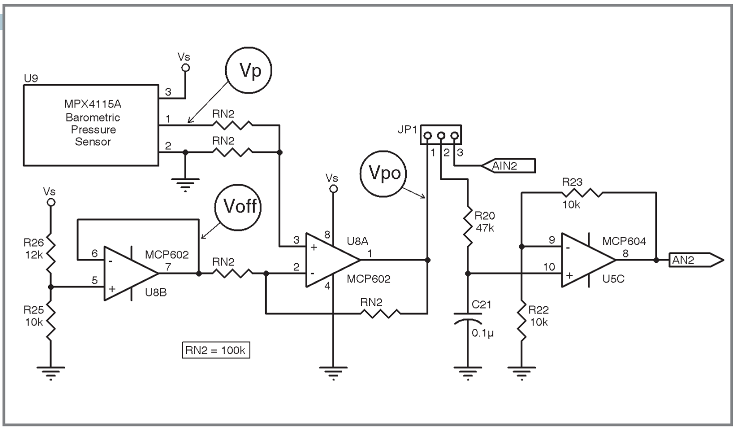


Figure 6—This is the EDL's barometric pressure sensor circuitry.

several common pressure units is available on *Circuit Cellar*'s FTP site.

The EDL uses a Freescale Semiconductor MPX4115A integrated silicon absolute pressure sensor. The MPX4115A is temperature compensated from –40º to 125ºC with a guaranteed maximum error of 1.5% from 0 to 85ºC. With a 5-V power supply, the MPX4115A's output can swing from 0.2 V at 15 kPa to 4.8 V at 115 kPa. However, BP covers a much smaller range. (A table with these results is also available on *Circuit Cellar*'s FTP site.) The widest expected variation in BP corresponds to only 40% of the MPX4115A's measurement range. By only using the portion of the MPX4115A's output that covers normal BP, the EDL can maximize sensitivity and match the ADC's dynamic range. Limiting the EDL to an approximately 65-to-105-kPa BP range does not restrict its use in mountainous regions. On a *standard* day, 65 kPa is equivalent to an altitude of 3,600 m (i.e., 11,800').

The EDL is limited to the desired BP range by subtracting an offset from the output of the MPX4115A then amplifying the remainder. The MPX4115A's output is shown as V_P. Its transfer function is $V_P = (0.009\ P - 0.095)\ V_S$, where P is pressure in kilopascals. The offset voltage (V_{OFF}) is generated by R25, R26, and U8B. R25 (10 kΩ) and R26 (12 kΩ) form a voltage divider with an output buffered by U8B. The equation for V_{OFF} is:

$$V_{OFF} = \left[\frac{10\ k\Omega}{(10\ k\Omega + 12\ k\Omega)}\right] V_S = 0.4545$$

V_{OFF} is subtracted from V_P by the substraction circuit formed by RN2 and U8A. The equation for offset pressure voltage (V_{PO}) is:

$$V_{PO} = V_P - V_{OFF} = (0.009\ P - 0.095)\ V_S - 0.4545\ V_S$$

The BP input uses one of the configurable op-amp buffers, which is configured as a single-pole, low-pass filter (3-dB point at 60 Hz) with a DC gain of two (see Figure 6). The DC voltage to the ADC (AN2) is:

$$AN2 = 2\ (V_{PO}) = 2\ V_S\ (0.00\ 9P - 0.5495)$$

The ADC reads AN2 as:

$$AN2 = \frac{\left(\frac{X}{1{,}024}\right)}{V_{CC}}$$

Combining these equations and solving for P yields (in kilopascals):

$$P = \frac{\left(\frac{X}{2{,}046} + 0.5495\right)}{0.009}$$

The result of this hardware processing is a doubling of the BP resolution from its unprocessed resolution of 0.917 ADC bits per millibar (mb) to 1.85 ADC bpmb. The actual BP range sensed by the EDL slightly exceeds the planned 65 to 105 mb on both ends of the range.

RELATIVE HUMIDITY

Relative humidity (RH) is the amount of water vapor in the atmosphere expressed as a percentage of the total amount the air can hold at the current temperature. The EDL uses an Ohmic Instruments SC-600 daughterboard, which uses a ceramic-resistive humidity sensor. The SC-600 board is mounted above the ELOG1 circuit board. The sensor's resistance changes in response to humidity and temperature in a complex exponential relationship. At 25ºC, the RH is approximated by the equation:

$$RH = 45.25\ VRH - 42.76$$

The RH calculated from this equation must be corrected for temperature according to the following equation, where T is temperature in degrees Centigrade:

$$RH\ (corrected) = RH - 0.7\ (T - 25)$$

EXTERNAL ANALOG INPUT OPTIONS

The input connector has terminals for two external analog inputs, AIN1 and AIN2, which are buffered by op-amp circuits that connect to ADC inputs AN1 and AN2, respectively. The op-amps may be reconfigured to cover a variety of possible inputs by changing passive components.

A single-supply, positive-gain op-amp has the general layout shown in Figure 7a. This layout's transfer function is:

$$V_O = \left[\frac{(R1 + R2)}{R1}\right] V_+ = \left(1 + \frac{R2}{R1}\right) V_+$$

Note the minimum gain is 1. The input can be increased, but it can't be decreased by this circuit.

For maximum flexibility, op-amp buffer's transfer function should be configurable from much less than 1 to much more than 1. This can be done with the circuit shown in Figure 7b. This is the EDL's buffer circuit:

$$V_+ = \frac{[R3]}{(R3 + R4)} V_{IN}$$

Therefore:

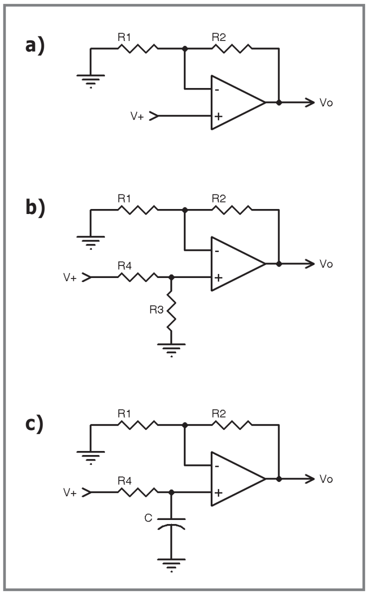


Figure 7—These schematics show the variations on the reconfigurable op-amps for the optional and external analog inputs. **a—**This is the general layout of a single-supply, positive-gain op-amp. **b—**You can use the circuit for maximum flexibility. The op-amp buffer's transfer function should be configurable from much less than one to much more than one, as shown. **c—**This schematic shows the single-pole, low-pass filter with gain.

$$V_O = \left[\frac{(R1 + R2)}{R1}\right]\left[\frac{R3}{R3 + R4}\right] V_{IN}$$

Even though R1–R4 are labeled as resistors, other components could be substituted. Consider the single-pole low-pass filter with gain, shown in Figure 7c, whose transfer function is:

Position	Symbol	Signal
1	GND	Ground, 0 V
2	SWCNT	Counter input, switch or digital in
3	SWIN1	Digital state input 1, switch or digital in
4	SWIN2	Digital state input 2, switch or digital in
5	V_s	Sensor power, on at the end of each sample period
6	AIN1	Analog input 1
7	AIN2	Analog input 2

Table 4—The input connector has terminals for three external switch inputs: SWIN1, SWIN2, and SWCNT.

$$V_O = \left[\frac{(R1 + R2)}{R1}\right]\left[\frac{1}{(1 + R4\,(j\omega C))}\right] V_{IN}$$

SWITCH INPUTS & EXTERNAL CONNECTORS

The input connector has terminals for three external switch inputs: SWIN1, SWIN2, and SWCNT (see Table 4). They are designed to be compatible with mechanical closures (e.g., manual switches, microswitches, or relays) but can also be driven by digital circuitry. Each input is buffered by a CMOS Schmitt-trigger inverter. The input is filtered to avoid switch bounce. The inverter and the pull-up resistor are powered by V_{CC}, which unlike V_S, is always on. The input filter time constants determine the minimum time the switch must be closed

```
ELOG MAIN MENU - Firmware V2.07
1 Download data
2 Set active inputs
3 Set sample period
4 Set clock
5 Display settings
6 Exit
?
```

Figure 8—This menu is displayed when the EDL is connected to a PC running a terminal program.

(7 ms) and open (30 ms).

Inputs SWIN1 and SWIN2 are read at the end of each sample period. No matter how they change during a sample period, it is their state at the end of the sample period that is put in data storage memory.

Input SWCNT is a counter input. Whenever the switch opens, SWCNT goes high and RB0 goes low. This generates an interrupt causing the PIC16F873 to wake from sleep and increment an 8-bit counter. At the end of each sample period, the total count is stored in data storage memory and the 8-bit counter is reset to zero. If the 8-bit counter is full (FF hex = 255 decimal), it will remain at this count and it won't rollover until reset. If SWCNT data consistently goes to 255, the sample period should be shortened.

The external input connector is a seven-position terminal strip located on the left when looking at the rear of the EDL (see Photo 1b). V_S is current-limited by a 10-Ω resistor (see Table 4). This protects V_S from shorts caused by external wiring. V_S can be used to power low-current sensors (e.g., the AD22100 temperature sensor), or it can be used as an end-of-sample-period signal to an externally powered circuit.

EDL OPERATION

The EDL can be configured to record any combination of its seven selectable inputs. The inputs are all sampled at the same rate.

The EDL has an Idle and an Active operating mode. In Active mode, the EDL is recording new data. Because this overwrites old data and the pointers to it, any previously recorded data is lost when the EDL enters Active mode. The EDL can only be placed in Active mode by selecting inputs and sampling time from the ELOG main menu.

The EDL will enter the Idle mode under the following conditions.

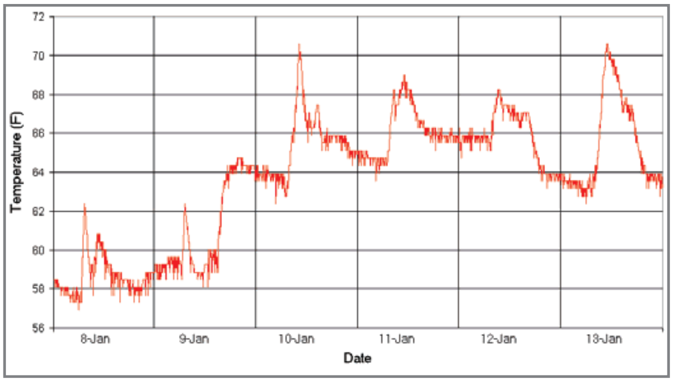


Figure 10—This graph shows the temperature in my work office during a week in January.

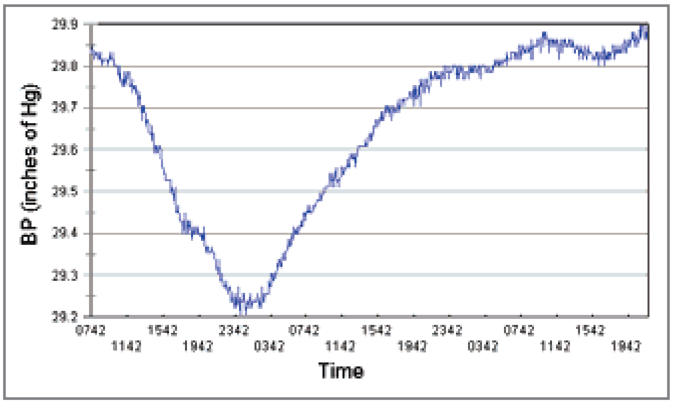


Figure 9—This graph shows the barometric pressure over three days while a hurricane traveled up the eastern coast of the U.S.

After any power interruption, when power is reapplied, the EDL will enter Idle mode. If power was interrupted while in Active mode, the data recorded prior to the interruption will be available. When the EDL detects an RS-232 connection (RTS is ON) it will enter Idle mode. If the EDL was in Active mode before the connection, the data recorded prior to the connection will be available. In Active mode, if the EDL fills the nonvolatile memory, it will change to Idle mode. When the memory is filled, the indicator LED will illuminate. If the EDL is operating on battery power and the battery voltage drops too low, the EDL will enter Idle mode. If the EDL was in Active mode before the voltage drop, the data recorded prior to the drop will be available. When low-battery voltage is detected, the indicator LED will flash.

When the EDL is connected to a PC running a terminal program, you'll see the menu displayed in Figure 8. Option 1 downloads the EDL's data in comma-separated value (CSV) format. The EDL converts all values to decimal for download. As shown earlier, the math needed to compute temperature, BP, and RH is rather complex, and it is best handled by a spreadsheet program (e.g., Excel). The terminal program should capture the download in a file with a CSV extension. Each line in the CSV file contains the data from one sample period. This is an example of one line:

38713,16909,00,370,01,1019,02,708,03,267,04,1004,05,22,06,1

The number representing the date (38713) is based on the standard spreadsheet representation of dates where 36526 is 01-Jan-2000. Simply change the spreadsheet column format to "Date" and the correct date will display.

The number representing time (16909) is the number of 2-s ticks since midnight. Spreadsheets represent time as the fraction of a day since midnight. Because there are 43,200 2-s ticks in one day, the fractional day is (ticks)/43,200.

The next two numbers (00, 370) are the channel numbers for temperature and the temperature value, respectively. Process the Centigrade temperature value using the formula given earlier (T = 0.2172X – 61.11), which can then be converted to Fahrenheit, if desired.

The following pairs of numbers each represent a channel number and the measured value. Each needs to be converted to the desired units using the appropriate formula.

Option 2 enables the user to select any or all of the seven inputs for recording. Note that if RH is selected, temperature should also be selected because it is required to accurately process the RH data.

Option 3 enables the user to select one of nine sample periods: 1 (4 s), 2 (10 s), 3 (30 s), 4 (1 min., 60 s), 5 (2 min., 120 s), 6 (5 min., 300 s), 7 (10 min., 600 s), 8 (30 min., 1,800 s), or 9 (1 h, 3,600 s).

Option 4 enables the user to set the EDL's RTC. Use Option 5 to display and confirm the EDL's configuration. This option displays the current date and time, the sample period, the inputs selected, and the time needed to fill the EDL memory for the selected inputs and sample rate.

EDL APPLICATIONS

Obviously, I can use the EDL for meteorological data gathering. In addition, I can use the AIN1 input for another temperature input (e.g., outdoor temperature) and the Events input for a rain bucket.

I used the EDL last fall when a hurricane crawled up the eastern coast of the U.S. Figure 9 shows the sudden drop in BP as the hurricane passed by my home.

You may be wondering if I was ever able to convince the building custodian that my office was cold. Figure 10 shows the graph that finally did the trick. ■

Brian Beard (info@lucidtechnologies.info) was just starting out in the electrical engineering field when the first microprocessors were developed. He has been an electronics designer and Assembly-level programmer ever since. Brian has worked for the government, in industry, and in academia. Outside of his full-time job, he does some design work and consulting on microcontroller systems.

PROJECT FILES

To download the accompanying information, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/266.

SOURCES

AD22100 Temperature sensor
Analog Devices, Inc. | www.analog.com

MPX4115A Pressure sensor
Freescale Semiconductor, Inc. | www.freescale.com

MAX3222 Transceiver and MAX756 DC-DC converter
Maxim Integrated Products | www.maxim-ic.com

PIC16F873 Microcontroller and 24LC128 Serial EEPROM
Microchip Technology, Inc. | www.microchip.com

SC-600 Signal conditioner
Ohmic Instruments Co. | www.ohmicinstruments.com

Embedded Systems Education

An Interview with Miguel Sánchez

Miguel Sánchez's fascination with electronics began at an early age. In July, Miguel and I discussed some of his early and current microcontroller-based designs, his 3-D printing projects, and how his career path—which started at the School of Chemistry at the Universitat de València—led him to teach serial communications and networking technologies at the university level.—Nan Price, Associate Editor

NAN: Where are you located?

MIGUEL: I live in the Mediterranean city of Valencia, on the east coast of Spain.

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

MIGUEL: I have been an electronics hobbyist since the age of 11, when I built my first radio. My uncle was an electrical engineer, and he was my mentor in that amazing new world. Once I learned about logic gates and microcontrollers, I was hooked. I started my training at a vocational school when I was 14.

NAN: How long have you been designing microcontroller-based systems?

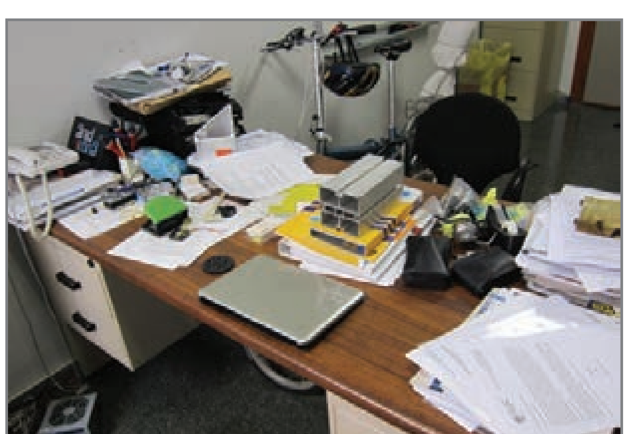
MIGUEL: I started using computers in 1978. I built my first microcontroller project in 1984 during my first year at Universitat Politècnica de València. I haven't stopped designing embedded systems since then.

NAN: Tell us about the first microcontroller you worked with. Where were you at the time?

MIGUEL: Our university's lab had Intel SDK-85 boards you could program in Hex using the built-in keyboard. I guess it wasn't built well. You sometimes lost all your work while typing your code. I learned that schematics were available and a terminal monitor was built in too. So, I built my first microcontroller-based board around an Intel 8085 using the same software that was on the original ROM. But, I changed the serial port delay value so I could use 9,600 bps instead of the original 110 bps on the terminal port. This way, I could do the same labs as my mates, but I could do my work in 8080 Assembler, which was available in Control Program/Monitor (CP/M) computers. At the time, I had an Atari 1040 ST that could run CP/M on top of a Z-80 emulator. Assembly code could be uploaded to my board's RAM memory and later executed using SDK-85 serial monitor code.

I used the 8085's Wait signal to build an additional EEPROM socket in this same board that, with the aid of a 555 timer, was my first EEPROM programmer. I used the Wait signal to delay write operations. In fact, I used this programmer to change the original baud rate to the new one, as I originally did not know that was something I'd want to change later.

My teacher, who is now one of my colleagues, was quite amused with my development and he gave me an A+. I learned a lot about microcontrollers, serial communications, Assembly language, monitor programs, and EEPROM programming algorithms. And, I learned it was not fun to design PCBs with system buses on only one copper layer.



This is Miguel's workstation.

NAN: How long have you been teaching at Universitat Politècnica de València? What courses do you currently teach and what do you enjoy most about teaching?

MIGUEL: I was working as a technician at a nuclear magnetic resonance (NMR) system in the School of Chemistry at the Universitat de València while I was studying computer science. I quit that job to join a small firm designing embedded systems. It took me a few days to realize I made a mistake, as there was nothing for me to learn there and the pay was much less than my previous job (a sacrifice I was only willing to make if I was learning new, cool stuff). Weeks later, in the fall of 1988, I got a teaching assistant job offer from the university where I was studying. I took the job and discovered that teaching was something I enjoyed doing, too.

I started out helping with the servo systems lab, but the next year I was offered a full-time teaching job, which was when I started teaching Computer Networks 101. I was familiar with serial communications at that time, but not very experienced on other networking technologies. You learn a lot by teaching others.

I have learned so much over the years, and I still feel a spark every time I see a student going through the same discovery process I went through many years ago. I guess it is the excitement of the discovery. Although it may not be something new and it is written in a book, each student learns it for the first time, so it is new for them.

I still teach Computer Networks 101, though the name has changed through the different reforms of the CS degree. During the last seven years, I have been teaching in English, as our university recognized the importance of opening up to visiting students and providing our own students with an environment where they can improve their English skills, which are very important in the job market.

"The availability of inexpensive development tools every year—as inexpensive as $10 in some cases—makes it easier for the students to get involved in embedded development. One student and I recently discussed a project to develop CNC control software based on an ARM Cortex-M3 processor using a $15 board."

NAN: You have been involved with the Computer Engineering department (DISCA) since 1988. Tell us about the department and the types of research you and your students do.

MIGUEL: Our department has evolved over the years. As one of the two founders of computer science studies at our university, our topics have evolved too. Our main focus was computer architecture. Another department at the university took care of the software side of the studies. Nowadays, the DISCA department has several research groups working on topics such as real-time operating systems, computer vision, next-generation networks, system on a chip, supercomputers, and network storage systems, among others.

NAN: What "hot topics" do your students find most interesting?

MIGUEL: Only a small fraction of our students are interested in the embedded world. Many students focus on software development. They are not interested in what is inside the box. Artificial intelligence (e.g., pattern matching, machine learning, etc.) seems to be one topic that sparks the interest of many students.

Lately, some of my students have discovered the embedded programming world by building a 3-D printer (not as a class project and not funded by the university). The project made them aware of many subtle details of embedded systems programming.

Another hot topic seems to be smartphone programming. Most students own a smartphone and use it as a tool to perform a real job, whether it is used an RC car remote controller or an RFID reader on a class-attendance monitor.

The availability of inexpensive development tools every year—as inexpensive as $10 in some cases—makes it easier for the students to get involved in embedded development. One student and I recently discussed a project to develop CNC control software based on an ARM Cortex-M3 processor using a $15 board. Sweet.

What we are lacking on campus is an easy and inexpensive way for students to get quality PCBs. The recent agreement to start Valencia's FabLab may help on this front. FabLab is like an Internet cafe that provides a laser cutter, a CNC lathe, or a 3-D printer instead of just networked computers. It's a design and digital manufacturing center that focuses on spreading the use of new manufacturing technologies. The lab is available for local personal, professional, and research initiatives and is part of the global network of fab labs lead by MIT.

NAN: *Circuit Cellar* has published six of your articles. The first one, "Build a Digital Video Recorder" (174, 2005), focused on building a personal video recorder using Linux and a DVB-S digital satellite receiver card. Do you still use the recorder? Have you made any significant changes to the system since it was built?

MIGUEL: I miss that system, as it worked like a champ. But, unfortunately, my provider changed the encryption system, making it incompatible with my DVB-S card. This meant going back to cable TV, which is missing several of the nice features that both TiVo and the system I built had in common.

NAN: You designed a system to simulate strokes on a keypad to trigger modes on an alarm system ("Reverse-Engineered ECP Bus," *Circuit Cellar* 201, 2007). Why did you design it and how have you used it?

MIGUEL: A local company wanted to give new life to old Ademco alarm units. These boards could only be programmed by a serial port socket once a certain service code was typed at the keyboard. I was asked whether an add-on board could be created to make these old boards Internet-enabled so they could be remotely managed and reconfigured over the 'Net.

The first thing I needed to do was to figure out how to simulate the required keystrokes. But I couldn't find any information about the way that bus worked, so I figured that out myself. Later, I thought both the information itself and the way I figured it out might be useful to others, so I approached *Circuit Cellar* editors with a proposal to write an article.

That project ended up as a Rabbit-core powered board that connected the alarm board and the remote access to its serial port. Combined with a virtual serial port on the PC, it fooled the original management software into thinking the PC was directly connected to the alarm board, although it was all happening over the Internet. But the project never made it to the market for reasons unknown.

NAN: In "Three-Axis Stepper Controller" (*Circuit Cellar* 234, 2010), you describe how you built an Arduino-based, platform-independent driver board. Tell us about the design.

MIGUEL: When I discovered the Arduino platform, I was surprised by a few things. First, this development system was not designed by a chip vendor. Second, it was not intended for engineers but for artists! Third, I was shocked because it was multiplatform (which was possible because it was based on Java and GCC) and because none of the other development systems I was aware of were so easy to use. The price was low too, which was a plus for hobbyists and students.

The aim of that project was to show all that to the readers. The idea was also not only to show how to build a stepper controller and to explain the difference between the drive modes and the bipolar and unipolar designs, but to demonstrate how easy it was to work with Arduino.

NAN: Your most recent *Circuit Cellar* article, "Image Processing System Development: Use an MCU to Unleash the Power of Depth Cameras" (263, 2012), describes how you used Microsoft's Kinect motion-sensing device for an interactive art project. Tell us about the project and how you came to be involved.

MIGUEL: My university offers a master's degree in fine arts. I met a professor from the drawing department who had seen a video of my vertical plotter on YouTube and was interested in contacting me, as we worked on the same campus. We became friends and he asked me to help him out with an idea for an installation.

The first approach used an RGB camera, but then Kinect was launched. From what I read on the 'Net, I was convinced it would be a better mousetrap. So, I bought one unit and started learning how to use it, thanks to the hack that had been made available.

The project required gathering visitors' silhouettes and later drawing them on a big wall. The drawing was performed with a properly scaled-up version of my vertical plotter, which, by the way, was controlled by an Arduino board.

I have found working with artists is a lot of fun too, as they usually have a totally different vision than engineers.

Miguel built two 3-D printers that use plastic filaments. The black one printed the blue parts he used for the second one.

NAN: Tell us about your go-to microcontrollers and embedded platforms. Do you have favorites, or do you use a variety of different chips?

MIGUEL: I won't bore you with the many microprocessors I have dealt with, but suffice it to say, my first one was Rockwell's 6502, which powered my first home computer, an Ohio Scientific Superboard II.

I have used Intel's 8051 microcontroller family, different versions of Atmel, and Microchip Technology's PIC microcontrollers, almost always with 8-bit systems. I have tested some 32-bit systems, like mbed or Cortex-M3, but I have not used them yet for any project. Maybe this upcoming CNC controller will be the one. I am currently working with a group of local sculptors to develop a four-axis foam milling machine that could use the extra computing power.

NAN: On your blog, *Fighting with Computers* (http://fightpc.blogspot.com), you post thoughts on topics ranging from Arduino RFID to wireless 3-D printing to Java topology to MEMS devices. Tell us about the types of projects you enjoy working on and blogging about.

MIGUEL: I have definitely enjoyed learning about 3-D printing lately, as it brings together mechanics, computing, electronics, and microcontrollers. And having a 3-D printer at home comes in handy every time you need to create or fix a part.

I blog about the problems I work on, especially when I discover something new (to me) and exciting. On the other hand, it is a self-reminder of where I found a certain tool or a way of solving a problem I was not aware of. I am delighted to discover from time to time that my blog has been useful to people I will likely never meet in person.

I think one key word here is challenge. Every time a new project pops up, it somehow becomes a challenge because you do not know how to solve it. A second key word is purpose. I get hooked whenever I find I can make something work better, faster, and cheaper (but usually not all at once).


ROUTE FASTER !

DesignSpark chipKIT™ Challenge Winners

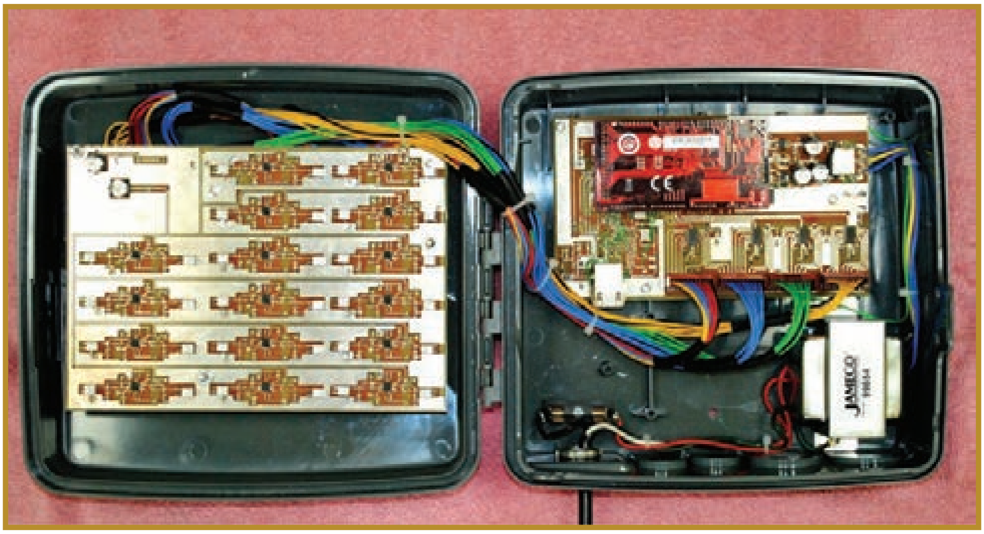
Beginning in September 2011, design engineers from around the globe were challenged to turn their hot ideas into cool solutions using Microchip Technology's chipKIT Max32 development board. While these energy-conscience designers connected in the DesignSpark forums, their projects continued to evolve until the final innovative and eco-friendly project designs were ready for the judges.

The judges reviewed all the entries and scored the projects based on technical merit, originality, design optimization, and quality of an extension card using DesignSpark's PCB tools. The judges' results are now final. Thanks to everyone who participated in this incredible competition and congratulations to all the winners! Enjoy your international recognition and prestige!

First Prize

Energy Monitoring System

The Energy Monitoring System (EMS) provides real-time home electrical usage data to occupants so they can make informed consumption-related decisions. The innovative system features a chipKIT Max32 development board along with two extension boards. A web server provides usage tracking on a circuit-by-circuit basis. The EMS interfaces with a home automation system for long-term monitoring and data logging. The system uses Microchip Technology's MPLAB development environment to utilize custom software written in C. It is integrated with the Microchip TCPIP stack.



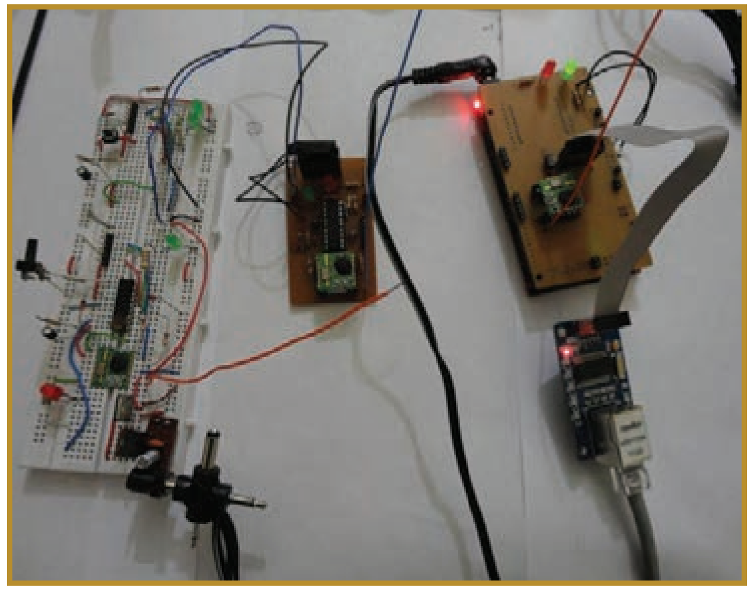
Dean Boman
United States
d.boman@cox.net

Second Prize

Home Energy Gateway

The well-designed Home Energy Gateway enables users to remotely monitor energy consumption and control household devices (e.g., lights). A chipKIT Max32-based embedded gateway/Web server communicates with two types of smart devices in a house: a smart meter that monitors average active real power consumption and several smart plugs in a home area wireless network. Users can monitor the smart plugs and use a web interface to make adjustments to them.

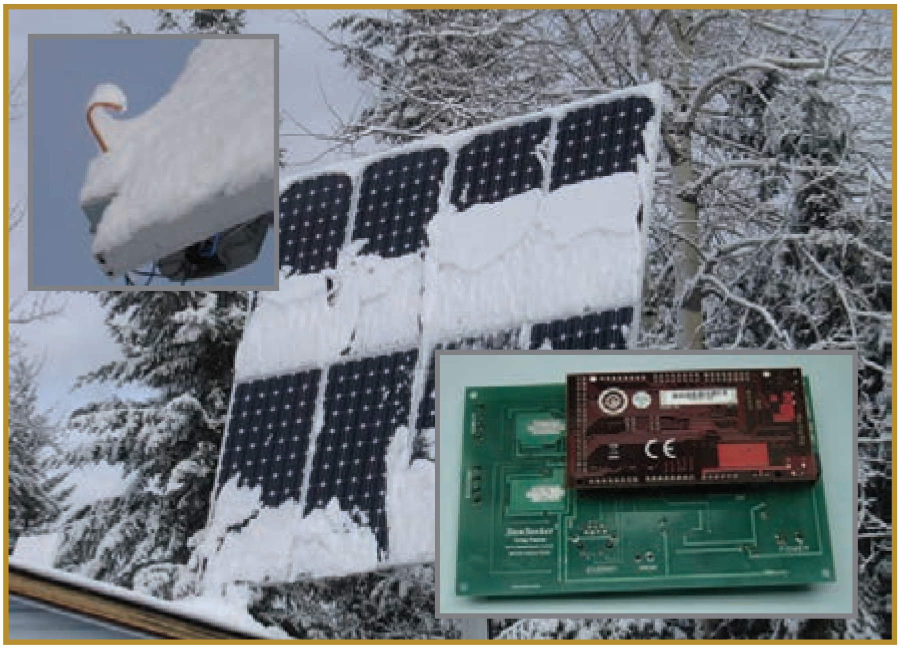
Raul Alvarez Torrico
Bolivia
raul-at@hotmail.com



Third Prize

SunSeeker (PV Array Tracker)

The SunSeeker enables you to control solar photovoltaic (PV) arrays to continually receive the most sunshine for optimal energy conversion. Featuring a Microchip Technology chipKIT Max32, the system is designed to track, monitor, and adjust PV arrays based on weather and sky conditions. It identifies conditions, measures PV and air temperature, compiles statistics, and communicates with a local server that enables the SunSeeker to facilitate software algorithm development and refinement. Diagnostic software monitors the design's motors to show both movement and position.



Graig Pearen
Canada
graig@pearen.ca

www.circuitcellar.com/contests/chipkit2012

Honorable Mention

Wireless Mesh Network Time Server

The solar-powered wireless mesh network time server receives time and date data from a GPS receiver module. This information is rebroadcasted into a mesh network through an XBee module that can be used by any devices attached to the network. A NiCd battery pack powers the server and a small PV panel recharges it. In Normal mode, the server broadcasts the time and date once per minute. If another device on the network requests it, the server switches to Stream mode and the transmission speed increases to once per second. The battery pack's voltage is measured and transmitted in Test mode. The relatively simple software enables you to program in a high-level language (MPIDE) while focusing on time and energy conservation.

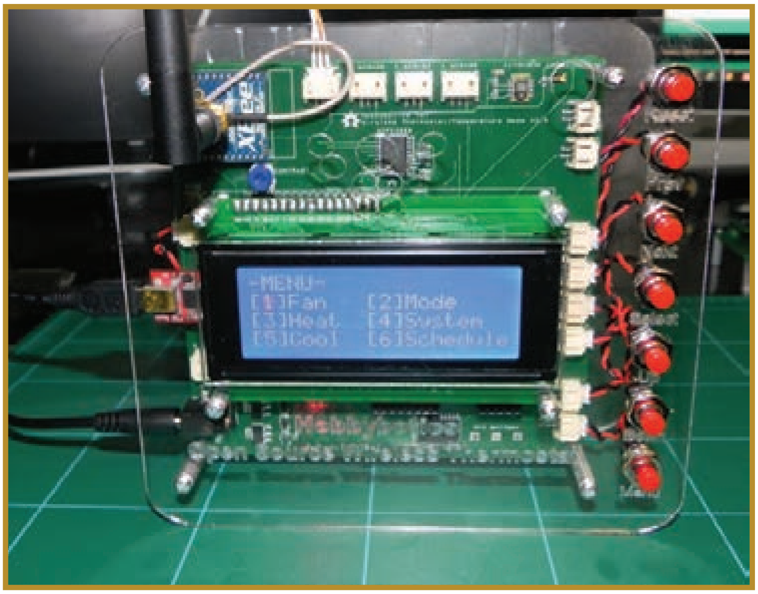
John Schuch
United States
hackersbench@gmail.com

Honorable Mention

Internet-Enabled Multizone Thermostat

An Internet-enabled thermostat provides users with maximum building temperature control. This novel system features three main sections: an XBee shield, a wireless temperature board, and an I^2C-controlled output board. It includes two wireless temperature control boards and two I^2C output boards. A router provides Internet access. The first node comprises a chipKIT Max32, a Max32 Ethernet shield, and an XBee shield (PCB). The second node is the wireless thermostat that controls an HVAC system and receives data from room nodes, which control a motorized damper to regulate each room's temperature.

Curtis Brooks
United States
brooksware2000@gmail.com



Honorable Mention

Eco-Friendly Home Automation Controller

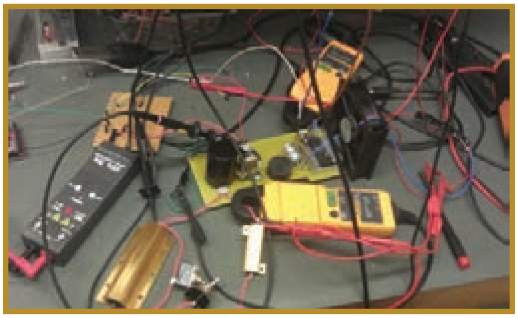
Built around a chipKIT Arduino-compatible board, this well-planned system connects to a custom chipSolar board of the same footprint that provides power from two Li-Ion cells. The board implements an MPPT charger that handles a solar panel's nonlinear output efficiency. Connectivity is provided by a custom chipWireless board comprising a quad band GSM/GPRS modem, an XBee socket, an SD card connector, and companion RTCC. MPIDE was used to write the software, and an SD card is used to log data from sensors and future memory requirements.

Manuel Iglesias Abbatemarco
Venezuala
mhanuel@ieee.org

Honorable Mention

MPPT Boost Converter

Maximum power point trackers (MPPTs) are used to ensure maximum power is transferred to a device from a range of renewable energy input applications (e.g., thermoelectric generators (TEGs), photovoltaic (PV) panels, and inductive power transfer (IPT) systems). An impedance-matching method was developed for a closed-loop digital control system. It provides an MPPT for input applications that have an approximate fixed internal resistance (e.g., TEGs and IPTs). The method was analyzed and used to construct the prototype converter with a microcontroller performing control operations. A modified model might be suitable for input applications with variable internal resistance due to temperature, light or both.

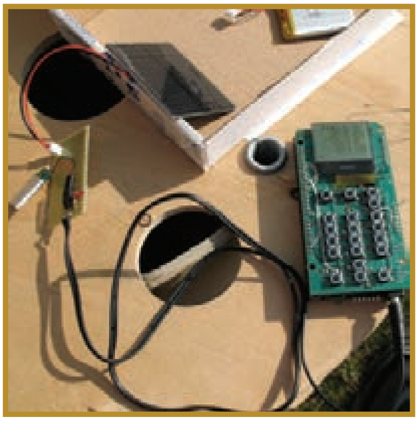


Ian Johnson, Sajjad Lalji, David Weight
United Kingdom
ian.johnson@wattcircuit.com

Honorable Mention

Handheld PIC18 IDE

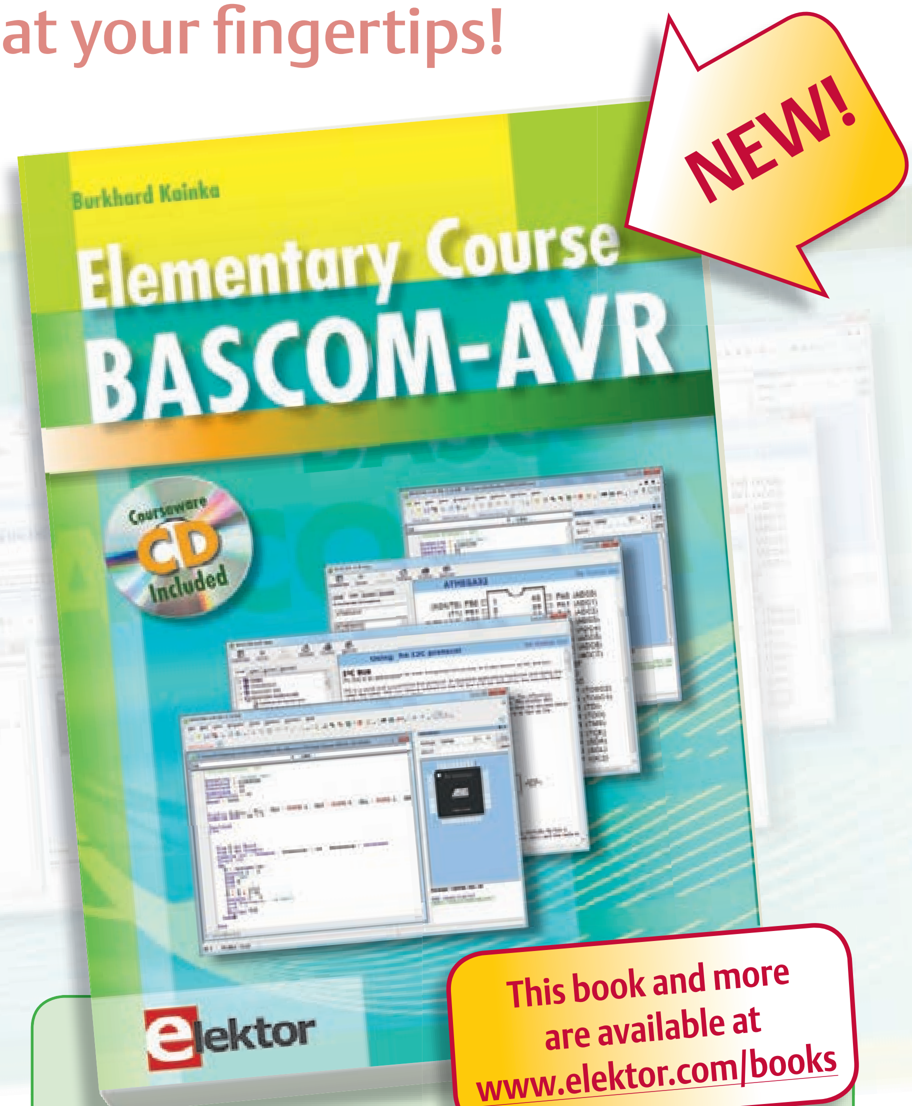
The handheld PIC18 IDE is an autonomous system for creating, editing, and assembling source files for a Microchip Technology PIC18. The process's binary output is programmed into a target PIC18 or debugged at a source level, called PP4. The hardware is simple. It consists of a user interface (LCD and keyboard), data storage, and a programming interface. The handy IDE also includes a BF interpreter for writing and executing scripts in BF language. You can use BASIC with the tag-along BASIC interpreter. The device is solar powered with a Li-Ion cell backup.



Jaromir Sukuba
Slovakia
j.sukuba@gmail.com

Elektor Shop

The world of electronics at your fingertips!



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

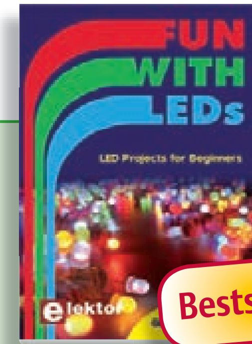
Books

Free Software CD-ROM included

Elementary Course BASCOM-AVR

The Atmel AVR family of microcontrollers are extremely versatile and widely used. Elektor magazine already produced a wealth of special applications and circuit boards based on ATmega and ATtiny controllers. The majority of these projects perform a particular function. In this book however the programming of these controllers is the foremost concern. Using lots of practical examples we show how, using BASCOM, you can quickly get your own design ideas up and running in silicon. The companion CD-ROM with this book provides sample programs and software including BCAVRDMO, AVR STUDIO, LCDTOOLS, and TERMINAL.EXE.

224 pages • ISBN 978-1-907920-11-0 • $56.40



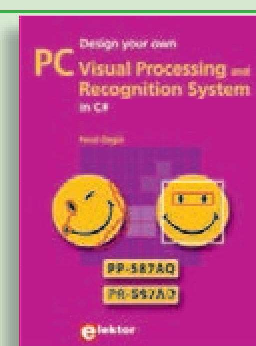
Bestseller!

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. 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



A comprehensive and practical how-to guide

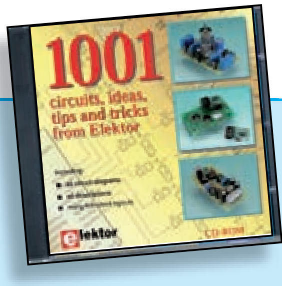
Design your own PC Visual Pro cessing and Recognition System in C#

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

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

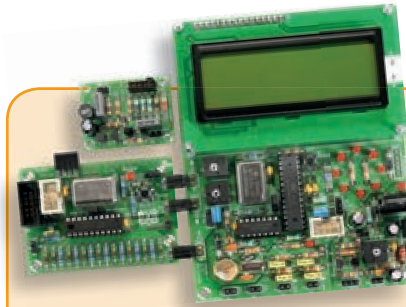
Art.# 120026-91 • $93.30

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

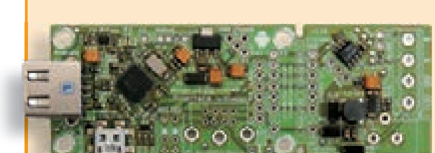


AVR Software Defined Radio

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 Mark Csele (Canada)

Miniature Accelerometers

DIY Acceleration Data Acquisition

Miniature accelerometers are versatile, inexpensive devices that can be used for many motion-sensing applications. This article describes a self-contained recording accelerometer that features a graphical LCD and downloads data to a PC for later analysis.

Recently, inexpensive digital accelerometers—which lend themselves to a host of applications—have become available on the market. Once restricted to applications such as missile guidance, new microelectromechanical systems (MEMS) devices are tiny and may be embedded into applications ranging from digital cameras to game controllers and robotics. Knowing the acceleration, for example simple calculus (by integrating twice) can be used to determine the position.

In this article, I'll detail how I built a portable accelerometer featuring an Analog Devices AXDL312 (see Photo 1). The device can record acquired data, display it on its LCD, and store it in flash memory. You can customize a similar design to meet your needs.

BASIC PHYSICS

Digital accelerometers measure acceleration (or, more correctly, "proper acceleration," which is the physical acceleration an object experiences) resulting from either motion or gravity. We often say an accelerometer measures "g force," but g force is actually not a force. Rather, it is felt due to the force of the earth pushing upward on the observer, preventing him from entering a freefall. A stationary accelerometer reads 1 g. An accelerometer in a freefall (in which an observer is weightless and feels no g force at all) reads zero. These forces are measured in an accelerometer by sensing the movement of a suspended mass—literally measuring the weight of the test mass as it accelerates.

Imagine a simple mass in a vehicle suspended by two springs (see Figure 1). At rest, the mass sits in a middle position, as shown. As the vehicle accelerates forward, the mass appears to move backward. In reality, the vehicle exerts a force on the mass in the forward direction compressing the rear spring, as shown. The mass has inertia, so it resists changes in

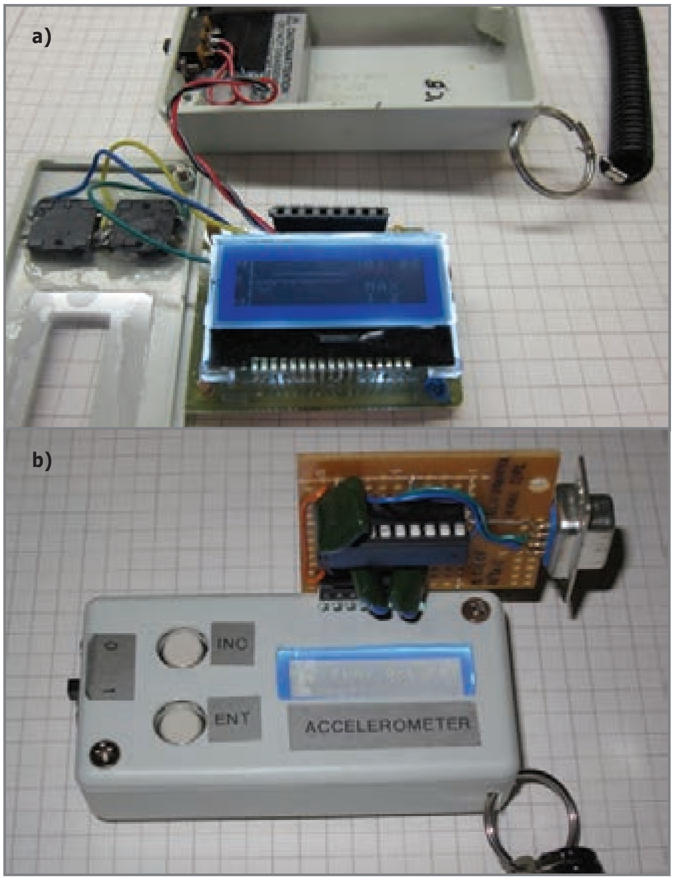


Photo 1a—The project was constructed on a small PCB featuring an eight-pin connector for in-circuit debugger/programmer access as well as connection to the download adapter board. The PCB sits inside a small plastic case along with a rechargeable lithium battery. Two thin push buttons on the front enable user control. **b—**This is the complete project with the serial download adapter. The adapter is installed only when downloading data to a PC and mates with an eight-pin connector on the PCB. The rear of the unit features three powerful rare-earth magnets that enable it to attach to a vehicle.

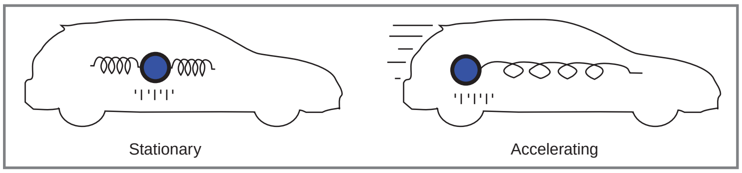


Figure 1—This diagram shows acceleration in a vehicle. As the vehicle accelerates forward, the mass moves toward the rear of the vehicle (from the vehicle's frame of reference). When the vehicle reaches a constant velocity, the mass returns to the center position.

momentum (it "wants" to stay still). If the "mass" was a vehicle occupant, that person would feel as if they were "sinking into the seat." Once a constant velocity is reached, the mass returns to the rest position even though the vehicle is moving (moving, but not accelerating).

In the downward (z-axis) direction, gravity acts on a mass to accelerate it toward the earth. Gravity is often referred to as a "force," but it is really an acceleration. What we feel as gravity is the force the earth exerts upward to keep an object from entering a freefall. Regardless of definition, acceleration is often measured in units of g with one g defined as an acceleration of 9.8 m/s^2.

Accelerometers come in a variety of types and can measure acceleration in one, two, or three axes. In a multi-axis device, a test mass is usually suspended in free space with numerous position sensors around it. These sensors can use inductive, optical, or capacitive sensing.

One may mathematically integrate acceleration to yield velocity, and integrate yet again to yield position. In early missile-guidance systems, this was done on a continuous-time basis using analog computers. Modern digital discrete-time (sampled) systems use numerical methods. Later in this article, I'll provide an example of this type of operation. Measuring acceleration in three axes (and then integrating acceleration from each axis twice to yield position in three axes) could enable you to determine absolute position (i.e., inertial navigation). However, measuring acceleration alone is not enough. Since rotation may occur without an acceleration change (i.e., a vehicle could change direction), you must also measure the angular rate for true inertial navigation. Imagine a three-axis accelerometer (e.g., the one used in this project) with the z-axis perpendicular to the earth and producing a 1-g reading. If the accelerometer was rotated about the plane perpendicular to the earth (i.e., in the x/y axes), no acceleration would be produced and there wouldn't be any indication that the unit's direction had changed.

THE ADXL312 DEVICE

My accelerometer is based on Analog Devices's ADXL312 three-axis digital accelerometer (see Photo 2). Supplied in a 5-mm square lead-frame package, this device can measure acceleration in all three axes (x, y, and z) up to ±12 g and with 13-bit resolution. The device's ultimate resolution is 0.0029 g per LSB.

The accelerometer can measure both dynamic acceleration (acceleration caused by motion) and static acceleration caused by gravity. When stationary, it will always read 1 g in the z-axis (assuming the unit is mounted flat). It can even be used as an inclinometer.

Like a "traditional" accelerometer, it uses a test mass supported by springs. However, in the case of a MEMS structure, the springs and mass are constructed from silicon, mounted directly on a chip wafer. Acceleration is sensed using capacitance between fixed plates and plates on the test mass.

The LFCSP lead-frame package is too small to mount for prototyping, so an adapter is required. As a "quick and dirty" solution, this package was mounted onto a standard eight-pin machined DIP socket used as a header. Epoxied upside down on the header, thin wires were used to attach the leads of the package to the pins (only six leads of the 32-lead package are required since the SPI is used). These wires were hand soldered using a powerful magnifying glass and an ultrafine (0.2-mm radius) tip on a soldering station. The package's bottom pad is actually ground and must also be connected. After it was attached to the header, the entire assembly was covered with epoxy to protect the delicate connections.

The chip is actually mounted upside down in this manner, so if it lies on a flat PCB (or a breadboard during prototyping), the ADXL chip will always read −1 g instead of 1 g due to gravity. (This can be easily corrected in firmware by multiplying this axis by −1 or later when the data is downloaded to a PC, usually to a spreadsheet.)

The ADXL312 uses either I^2C or SPI. It can sample at programmable rates of between 6.25 and 3,200 samples/second. For this project, the SPI is used. The accelerometer uses 28 8-bit registers to set chip parameters or read chip values. A typical exchange between the chip and a microcontroller begins by sending a command (read or write) and a register address (0 to 63) that requires eight clocks. The next eight clocks enable data to be sent to, or read back from, the register specified. Registers are used for device configuration (e.g., sample rate and data format), and for access to acceleration data (with 13-bit numbers separated into two registers with low- and high-byte data).

Figure 2 shows three SPI packets exchanged between the ADXL chip and the microcontroller. In this case, it is a read of

Photo 2—Look how an Analog Devices ADXL312 digital accelerometer measures up to a Canadian penny. The tiny lead-frame package measures only 5 mm on each side.

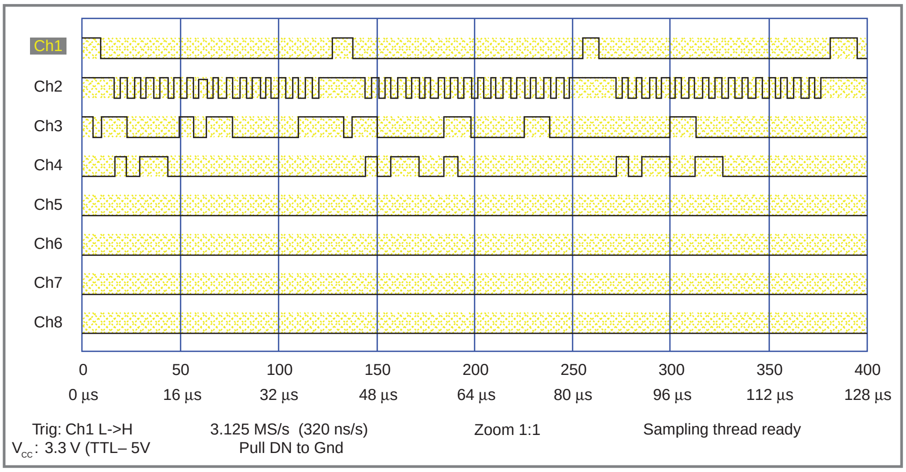


Figure 2—This graph shows the SPI communications between the Analog Devices ADXL313 digital accelerometer and the microcontroller as viewed on a logic analyzer. Three packets are shown: a read to register 0x30 (to check for DATA READY), a read of register 0x32 (x-axis low byte), and a read of register 0x33 (x-axis high byte).

chip registers 0x30, 0x32, and 0x33. Channel 1 is chip select, Channel 2 is clock, Channel 3 is serial data from the chip, and Channel 4 is serial data sent to the chip. In the first packet, 0xB0 is sent to the chip in the first eight clocks (register address 0x30 with bit 7 set high indicating a read command). The chip responds with 0x83 in the last eight clocks (where bit 7 set high indicates data is available from the chip). Reads are then performed on registers 0x32 and 0x33 to read the x-axis acceleration. This example also helps illustrate the logic analyzer's usefulness. During debugging, a faulty piece of code continually sent the same command ("read device ID" command 0x00). The logic

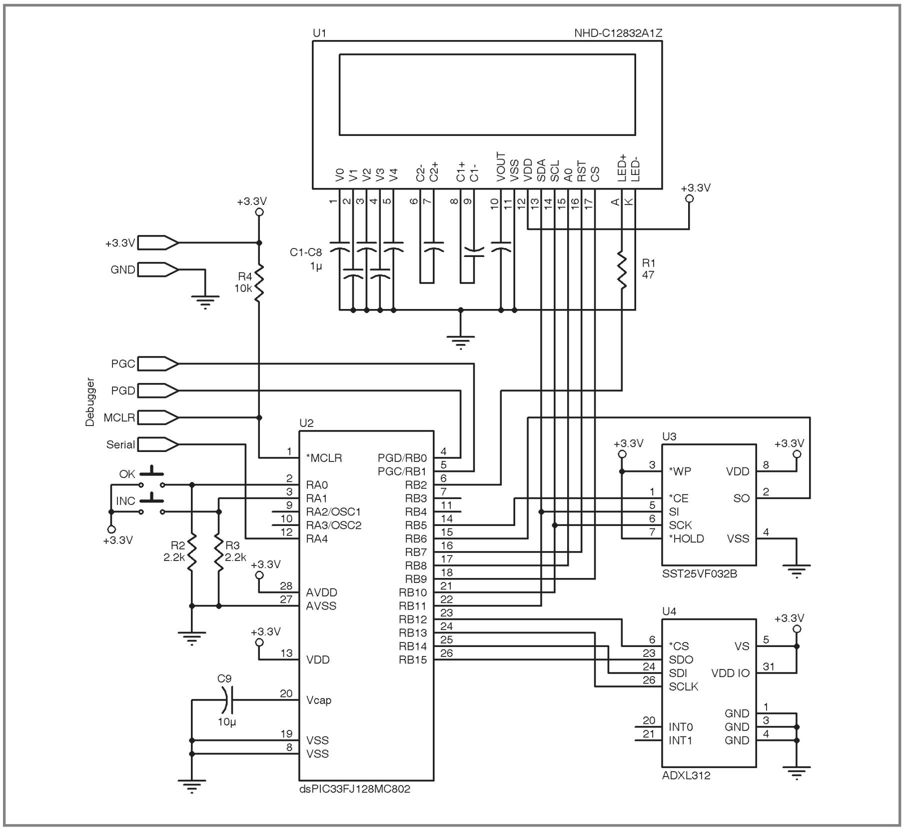


Figure 3—A Microchip Technology dsPIC plays an essential role in the accelerometer's design. The serial download adapter, which uses a Maxim Integrated Products MAX3223 chip, and the MAX882 voltage regulator are not shown.

analyzer—a homebrew unit based on Francis Deck's design from "A High-Speed Logic Analyzer for Windows 95" (*Circuit Cellar* 89, 1997) and updated several times—immediately revealed the nature of this fault.

The chip is quite advanced. It has a number of features that aren't used here but would likely prove useful with higher sample rates, including a FIFO buffer for acceleration samples and programmable interrupt outputs. This buffer enables the accelerometer to interrupt the host processor when the FIFO fills or only when activity (acceleration) is detected above a preset threshold, which enables power-saving modes to be implemented.

PORTABLE SYSTEM

The portable system can acquire data and then later download it to a PC. It incorporates a small Newhaven Display NHD-C12832A1Z-NSW-BBW-3V3 LCD. This chip-on-glass display is inexpensive, offers full graphics capability, and has an easy-to-use serial SPI. The only issue with this display is that it requires eight external 1-µF capacitors (which ultimately occupy board space, although SMT capacitors could be used).

My original plan was to use a Microchip Technology dsPIC microcontroller's on-chip flash memory to store data (using the ability to write directly to flash program memory under program control). However, the 128 KB offered was not enough, so an external flash memory chip was added. I used a Microchip SST25VF032B SPI serial flash, which offers 32 Mb (4 MB) of storage, is packaged in a small eight-pin SOIC package, uses a serial SPI, and is quite fast compared to older flash or EEPROM memories.

I used a Microchip dsPIC33FJ128MC802 digital signal controller that only has two SPIs. Most devices only have one, which was one of the main reasons chose this particular chip. However, I used a total of three SPI devices on this design (ADXL, LCD, and SST flash memory). The ADXL uses a 16-bit mode. The other two devices use an 8-bit mode, so those two share the same interface (albeit with separate chip-select lines).

For the supply, I used a tiny Olympus digital camera battery with a 3.6-V rated output voltage. I used a Maxim Integrated Products MAX882 low-dropout regulator to supply all chips with 3.3 V. It will produce this output voltage with only 3.37 V of input supplied. The entire circuit is very low-power, with only 16 mA measured current drain with the backlight on, and 9 mA when recording with the backlight off. Since the backlight will consume more power than the rest of the circuit, the backlight current was limited to 7 mA instead of the usual 25 mA. The backlight is controlled via an I/O line from the processor so it can be turned off during recording (to keep it from annoying guests on dark amusement rides).

The complete schematic shown in Figure 3 does not include the RS-232 serial interface, which enables data to be downloaded to a PC. That interface, which is on a separate board that plugs into the accelerometer when required, uses a Maxim MAX3223 driver chip, The chip is similar to the ubiquitous Maxim MAX232 but operates at 3.3 V and uses only 0.1-µF capacitors. It also features a shutdown mode to limit

Photo 3—These are the main menu and record screens, as seen on the LCD. **a—**The main menu displays four items. The Record function highlighted. When that item was selected with the OK button, the next available memory slot (05) was selected. **b—**During recording, acceleration data is displayed on a graph along with the maximum value achieved and the elapsed recording time.

current drain when not in use, but this is unused since the download interface is only plugged in as required. The interface is powered from the accelerometer itself (the regulated 3.3-V supply) and uses only a single-transmit channel. Since all modern laptops lack a serial port, an RS-232-to-USB bridge is normally used.

PACKAGING

To keep the unit as small as possible, all components were mounted on a small PCB with the LCD mounted on top. The LCD itself is supported on two thick copper wires used for the backlight terminals. Many components are available only in surface-mount technology (SMT), so it made sense to design the unit to enable direct mounting of such components on the board. However, for this prototype, I used some components already in the DIP adapter sockets (e.g., the ADXL312 chip). Using a through-hole component dictated how thin the unit could be built. A much thinner unit could be built using all SMT parts.

The PCB, along with the lithium battery, is encased in a small plastic box as shown in Photo 1a. The completed unit is shown in Photo 1b. The unit features two user push buttons (two thin push buttons glued to the top cover) used for increment and enter functions for a simple menu-driven interface. These push buttons are recessed from the front surface to prevent accidental presses.

A rectangular window was cut into the box for the LCD, as well as eight holes for access to an I/O connector on the PCB (to which the male connector on the download adapter mates, and through which access is provided to pins for in-circuit programming and battery terminals for charging, since removal is difficult).

Once completed, the prototype unit produced a lot of noise. With the unit sitting stationary on the workbench, the −1-g value (negative, since the accelerometer chip was mounted upside down), which is normally around −350, randomly varied up to 50 LSB values. The electrical noise was drastically reduced by adding both a 0.1-µF capacitor across the power leads directly under the ADXL chip and another 10-µF tantalum capacitor right beside the ADXL. The result is a noise reduction by an approximate factor of five! When laying out the PCB, be sure to include thick ground leads and adequate supply decoupling.

SOFTWARE

The LCD is graphical with a resolution of 32 dots across × 128 dots high. It is static requiring no refresh signal, and features a SPI with a basic command set enabling positioning of the write pointer to display memory, writing of display data, and so forth. When displaying text, the display is logically arranged (in software) as four lines each eight pixels high. Character generation is supplied by a lookup table derived from the video section of an old Ohio Scientific (circa 1979) computer rearranged for column patterns as required for this LCD (as opposed to video generation, which is accessed as rows).

When first started, the system verifies the accelerometer and flash memory chips are both working by reading each's device ID. The user is then presented with a menu with four options: record (REC), PLAY, download (D/L), and format (FMT), as shown in Photo 3. Menus are navigated with two user push buttons, ENTER and INC. Pressing INC cycles between options with the currently selected option displayed in inverse video. Pressing ENTER selects that option.

The REC function acquires data from the accelerometer chip, displays it as a graph on the LCD, and stores it to flash memory for later download. Although samples are acquired at a 25-Hz rate and with a total range of –12 to 12 g, the graph is updated only once per second and only the –3 to 3-g range is displayed. The graph (which consumes more than half of the total display width) fills from left to right. When it is full, it simply recycles from the left, constantly updating during record. The right side of the display shows the elapsed time and the maximum acceleration recorded (in g, found by dividing the acceleration by the 1-g value or approximately 345).

The flash memory chip is quite large (with byte addresses ranging from 0x00000000 to 0x003FFFFF) enabling storage of multiple "runs." Memory is allocated as logical 128-KB blocks called "slots." On format, the entire memory is cleared to 0xFF. When the record function is selected, the code examines each memory slot for availability with the first available memory slot identified by the first few bytes of that slot being 0xFF. (When a memory slot is used, the first byte is 0x00.)

Samples are then taken and stored into flash as 8-byte structures as follows: sequence number (2 bytes); x-axis value, 16 bits (2 bytes); y-axis value, 16 bits (2 bytes); and z-axis value, 16 bits (2 bytes).

The sequence number identifies the sample time and ensures data integrity during download. When a run is recorded to a new memory slot, the first byte written is sequence number 0x00, identifying that slot as used. Values read from the accelerometer are stored as 16-bit numbers in the 0xF000 to 0x0FFF range in little-endian, two's-complement format. With a 128-KB memory slot and a sample rate of 25 Hz, each slot can hold just over 10 min. of data. A total of 32 such slots are available with the 4-MB flash memory. When the user presses ENTER during recording, that function is terminated leaving all unused memory initialized to 0xFF (then the sequence number 0xFFFF identifies the run as complete).

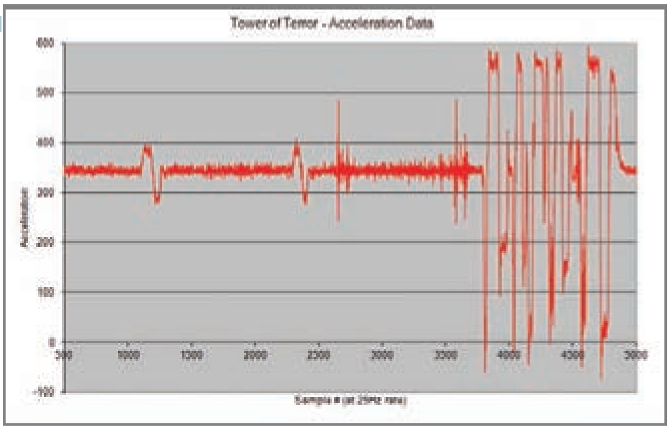


Figure 4—This graph shows "raw" acceleration data from Disney World's Tower of Terror attraction. Two small bumps between 1,000 and 2,500 take riders to an intermediate floor where the ride car is loaded into the main drop shaft. The main drops for the ride begin just after sample 3,500.

Sampling is also terminated when a memory slot is full (after 10.5 min. of recording).

Another function available to the user from the main menu is data playback, which simply graphs the first few bytes of stored data in a run. Given the small size of the display, useful analysis cannot be carried out "in the field," so this feature is primarily used to ensure data was actually collected. When playback is selected, the user must select a slot using the INC key (unlike record, which automatically selects the next available slot to use).

Finally, a download feature (e.g., playback) enables the user to select a memory slot to download via a 115.2-kbps serial link to a PC. Upon setting up a pointer to the beginning of the selected memory slot, data is downloaded to the PC in comma-separated value (CSV) format with the format <Sequence #>,<X>,<Y>,<Z><CR> for each data point where the first four numbers are in signed decimal format followed by a carriage return. For example, a typical data record might be "02528,00009,–00342,00007," meaning sequence number 2528 has an x acceleration of 9 (0.026 g), a y acceleration of –342 (–1 g), and a z acceleration of 7 (0.02 g). Hence, the x and z axes have an acceleration of nearly zero, and the unit is stationary with the y axis facing vertically. This format can be logged to a file by a terminal program and directly imported into a spreadsheet enabling graphing and further data analysis.

TESTING & APPLICATION

I took the accelerometer on a family trip to Disney World in Florida for "research" and testing. One of the amusement park's most interesting attractions is the Tower of Terror, in which riders are dropped 13 stories at velocities exceeding freefall. Figure 4 is a graph showing the acceleration recorded on the ride. (The graph shows the raw data from the accelerometer as a function of a sample number with only the z axis shown.) Upon starting the ride, the accelerometer (at rest) reads 1 g, as expected. Two small lift sequences take riders to a middle floor (known as the "fifth dimension" sequence) where the ride car is loaded into the main drop


Imagine, Design, EARN!
club JAMECO®
You design an electronics project.
We'll turn your bright ideas into cash!
HAVE A GREAT PROJECT IDEA?
Design an electronics project at ClubJameco.com, identify the components, write step-by-step instructions, and that's it. We'll do everything else!
Start Earning Now!
www.ClubJameco.com
Copyright © 2012, Jameco Electronics. All Rights Reserved. 1355 Shoreway Rd., Belmont, CA 94002 USA Info@ClubJameco.com

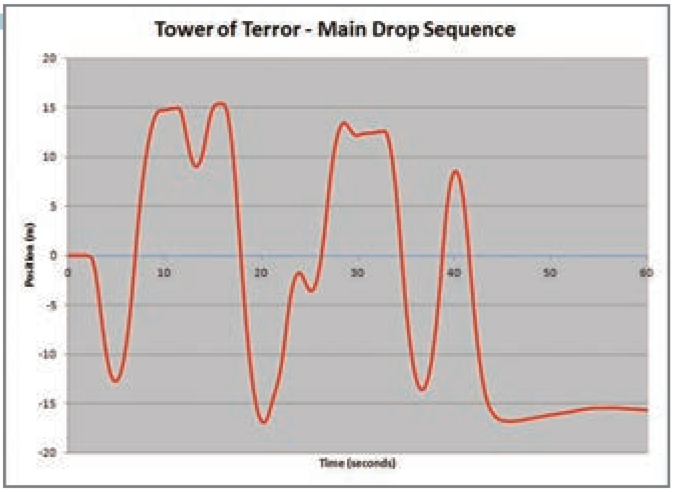


Figure 5—This graph shows the Tower of Terror position data. Acceleration data was converted to units of m/s^2 (and sample number to units of seconds). Acceleration is then numerically integrated twice to yield position.

shaft. The main drop sequence then begins with the acceleration on several drops reaching below 0 g (i.e., riders feel truly "weightless" as they are quite literally "pulled downward").

The graph in Figure 4 can, of course, be calibrated in real units of time by dividing the sample number by the sample rate of 25 Hz, and acceleration in units of g by dividing the number provided by the accelerometer by the constant number read while the unit is at rest. Acceleration can also be expressed in unit of m/s^2 by multiplying the figure in g × 9.8 (where 9.8 m/s^2 represents 1 g).

Having expressed acceleration in real units, simple physics can be used to derive the position (in the single-axis example here, the height of the ride car at any time) by integrating acceleration twice. Assuming the unit is at rest when started, the acceleration that produces motion is the acceleration as read on the device minus the effect of gravity (1 g)—the reading taken when the device is stable and at rest (i.e., reading a consistent value of 1 g). Integrating acceleration yields velocity. In older systems with an analog accelerometer, this was performed by analog integrators. However, in a digital system, samples are not continuous, so a numerical integration is used where values are computed at discrete intervals of time. (You might recall that in calculus, Δt approaches zero and becomes infinitely small whereas the opposite is true here.)

The velocity at any instant is then $v = v_0 + a\Delta t$ where v_0 is the previous velocity (starting at rest, but presumably increasing as time progresses), a is acceleration at that instant, and Δt is the time interval (1/sample frequency). The position at any instant can be similarly computed as $d = d_0 + v\Delta t$, where d_0 is the previous position and v is the instantaneous velocity as computed from the previous equation.

Data from the Tower of Terror ride is easy to analyze given that only one axis (z) is significant. Figure 5 shows the resulting positional data. The complete spreadsheet containing acceleration data and integrations is available with other project files on *Circuit Cellar*'s FTP site. The start position in the graph is assumed to be zero but it is, in fact, 17 m above the ground. There are several large drops in the ride, including one massive drop of more than 32 m.

Also evident in this plot is a phenomenon known as "drift" in which small errors in acceleration lead to large errors in velocity and even larger errors in position. This is seen at the end of the plot in Figure 5 (between 45 and 60 s) where the accelerometer was stationary, but the position appears to change. The primary cause of such drift is a DC bias or offset error, for which this accelerometer chip has built-in compensation simply by programming an offset register. (This feature was not used in this test run, which revealed the issue.) Other problems that may affect analysis accuracy include noise (noise was still evident in the signal from a stationary device) and sample rate (which can be set as high as 3,200 Hz).

While the application presented here was designed to demonstrate the use of the accelerometer in a "typical" physics application, the acceleration measurement can be used in many other applications, including: tracking critical shipments, motion sensing for game control, true inertial navigation (when combined with gyros, for which purpose MEMS gyros are also available), and analysis of mechanical systems. ■

Mark Csele (mcsele@computer.org) is a professor at Niagara College, Canada. A scientist and professional engineer (PEng), he has taught courses in embedded control systems and has worked on various industrial control engineering projects. Mark is the author of Fundamentals of Light Sources and Lasers *(Wiley, 2004) and a previous contributing author to* Circuit Cellar. *His article, "DSP-Based Color Organ: Use the Convolution Technique to Create High-Performance Filters" was featured in issue 249, 2011.*

PROJECT FILES

To download the code, as well as test data from several amusement rides, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/266.

RESOURCES

F. Deck "A High-Speed Logic Analyzer for Windows 95," *Circuit Cellar* 89, 1997.

SiloWorld, Systems Test Operations Support Group, "Theory of Operation & Mechanization of the Typical Inertial Guidance System," 1959, www.siloworld.net/DOWNLOADS/TheoryOfOperationOfInertialGuidanceSystem.pdf.

SOURCES

ADXL312 Digital accelerometer
Analog Devices, Inc. | www.analog.com

MAX882 Linear regulator and MAX3223 and MAX232 RS-232 transceivers
Maxim Integrated Products, Inc. | www.maxim-ic.com

dsPIC33FJ128MC802 Digital signal controller and SST25VF032B SPI serial flash
Microchip Technology, Inc. | www.microchip.com

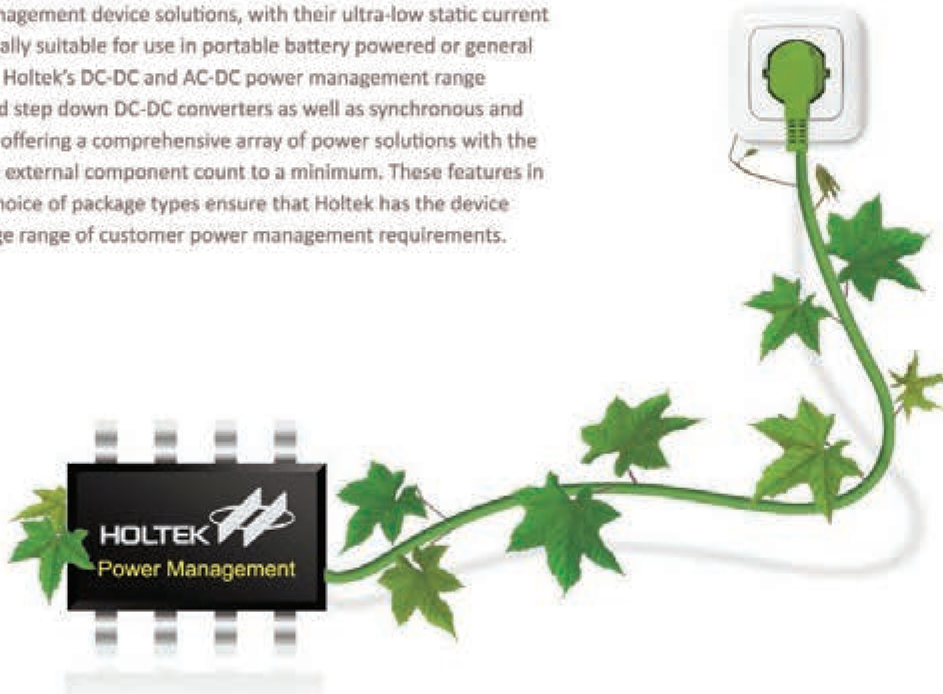
NHD-C12832A1Z-NSW-BBW-3V3 LCD module
Newhaven Display International, Inc. | www.newhavendisplay.com

HOLTEK
www.holtek.com

Ultra Low Standby Current

Advanced Eco-friendly Power Management Solutions

Holtek's wide-ranging power management device solutions, with their ultra-low static current specifications, make them especially suitable for use in portable battery powered or general AC powered electronic products. Holtek's DC-DC and AC-DC power management range include a full range of step up and step down DC-DC converters as well as synchronous and asynchronous AC-DC converters, offering a comprehensive array of power solutions with the added benefit of still keeping the external component count to a minimum. These features in combination with an extensive choice of package types ensure that Holtek has the device range to meet the needs of a huge range of customer power management requirements.



- Exceptionally low power consumption
- High input voltage range
- High DC-DC conversion efficiency
- High functional integration power IC solutions
- Huge range of high efficiency linear and switching regulator solutions
- Multiple package types for a wide range of customer application requirements

Series	Part No.	Features
PFM Asynchronous Step-up DC to DC Converter (1)	HT77xx/HT77xxA	Asynchronous DC-DC converter step-up voltage solutions for portable products - 100mA~200mA
PFM Synchronous Step-up DC to DC Converter (2)	HT77xxS/HT77xxSA	Synchronous DC-DC converter step-up voltage solutions for portable products - 100mA~200mA
PFM Synchronous Step-up DC to DC Converter (3)	HT77S1x	High conversion efficiency step -up voltage solutions for portable products - 300mA~350mA
Step-down DC to DC Converter	HT7465/HT7466	High input voltage, step-down DC to DC converters – output current 2A/3A
Charge Pump DC to DC Converter	HT7660	Charge pump type negative V_{DD} voltage converters
Off-Line AC to DC Converter	HT7A3942/HT7A6003/ HT7A4016/HT7L2102/ HT7L2103	Low power off-line AC-DC Converters – Energy Star 2.0 compliant
Non-Isolation AC-DC LED Lighting Driver	HT7L4091	Low power non-isolated AC-DC LED lighting drivers

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

by Patrick Schaumont (USA)

Hardware-Accelerated Encryption

Hardware acceleration can give slow embedded software a helping hand. Using modern field-programmable gate arrays (FPGAs), you will find the possibilities of integrating hardware with software are endless. This article describes how a hardware accelerator improves the performance of an encryption software routine by an order of magnitude. The technique with AES encryption on a low-cost DE0-nano FPGA kit will also be demonstrated.

As flexible as it is, software has a fundamental limit. Software is sequential in nature, as processors execute instructions one at a time. The execution speed of software is, therefore, limited by a single instruction's execution speed. High-end processors use many tricks to work around this bottleneck—from using superscalar pipelines that simultaneously handle multiple instructions to multicore designs that run sequentially as parallel threads. Embedded processors, however, don't have the luxury of superscalar or multicore design. They run on a diet of resources, in terms of power and in terms of computational elements. To deal with an embedded processor's performance barrier, something different can be done. A specialized hardware acceleration module can be built and integrated with the processor. This makes the design very efficient at one particular task, while still having the general-purpose performance for regular computations.

Modern field-programmable gate arrays (FPGAs) are well suited to this scenario. They come with a design environment that supports hardware design in Verilog or VHDL, and they support embedded processor designs. After a processor is configured in an FPGA, it can execute embedded software written in C. Such processors are called softcores, as they are configured in the FPGA, together with other regular hardware modules.

Figure 1 compares a traditional (software-only) embedded design flow with one that uses a hardware accelerator. In the traditional flow (Figure 1a), the designer writes the application in C for a processor, which happens to be configured in an FPGA. With a hardware accelerator (Figure 1b), the design flow is different. First, the designer analyzes the application (a C program), to identify the critical parts that are most suitable for hardware acceleration. Next, the C program is broken into two parts. One part remains plain C code, and runs on the embedded processor. The performance-critical part, however, is converted into hardware and implemented as a hardware module attached to the processor. Obviously, the result of partitioning is application-dependent. A different application yields a different accelerator.

In this article, I'll use a common cryptographic algorithm, the advanced encryption standard (AES). I will build a hardware accelerator for AES encryption and decryption that significantly outperforms the (optimized) software implementation of AES. First, let's take a detailed look at AES in order to understand its structure and its potential for hardware acceleration.

AES CIPHER

AES is an algorithm for bulk data encryption. An encryption algorithm creates an intractable combination of input data bits and a secret key, which can only be undone using a matching decryption algorithm that operates with a matching secret

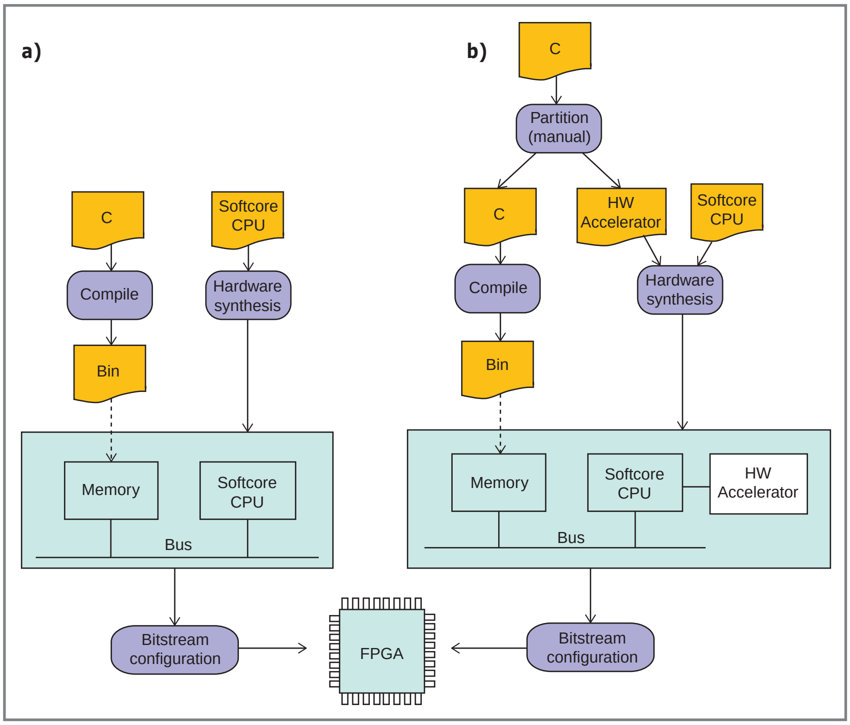


Figure 1—These diagrams show two different embedded processor design flows using an FPGA. **a—**A C program is compiled for a softcore CPU, which is configured in an FPGA. **b—**To accelerate this C program, it is partitioned into a part for the software CPU, and a part that will be implemented as a hardware accelerator. The softcore CPU is configured together with the hardware accelerator in the FPGA.

key. AES comes in three variants (AES-128, AES-192, and AES-256), which use different length keys. In AES-128, a block of 128 data bits is combined with a key of 128 data bits to provide an encrypted or decrypted result of 128 bits. AES-128 is a symmetric-key algorithm, which uses the same key for encryption and decryption. I'll focus on AES-128 for now. The References section at the end of this article includes pointers to the detailed specification of AES and all its variants.

Figure 2 shows how AES-128 operates. It is an iterative algorithm that consists of 10 iterations or rounds: nine regular rounds and one final round. Each round has a similar structure and consists of a substitution and a permutation step, followed by a roundkey addition. A substitution in a block cipher changes the symbol set's ordering. A plain-English substitution, for example, would be to substitute every letter from the original alphabet for a different letter. A permutation in a block cipher means symbols will be moved around to different positions in a block. A plain-English permutation, for example, could reorder every group of five characters from 1-2-3-4-5 into 5-2-3-4-1. The References section also includes links to the precise definition of the substitutions and permutations used in AES. For AES to work well, the substitution/permutation steps must be efficiently implemented. I will return to this point later when I discuss the hardware accelerator.

AES encryption's real secrecy comes from the periodic additions of a (secret) roundkey. Every round uses a roundkey, and they are all derived from the input key. This operation is called the key schedule or key expansion. If many data blocks need to be encrypted with the same key, it is beneficial to precompute all the roundkeys and store them. On the other hand, if the key changes with every data block, precomputation is less useful. For AES, computing the roundkeys has about one quarter of the computational complexity of an encryption round. Hence, when performance is a concern, deciding when to compute the roundkey schedule is a relevant question.

AES CIPHER C IMPLEMENTATION

The AES software implementation I used for this article is an optimized open-source implementation commonly found in AES implementations for Internet protocol suites such as SSH and SSL. Encrypting a block of 128 bits (i.e., 16 bytes) requires first computing a key schedule, then calling the encryption function:

```
AES_KEY keyexp;
unsigned char key[16], plaintext[16],
ciphertext[16];
AES_set_encrypt_key(key, 128, &keyexp);
AES_encrypt(plaintext, ciphertext,
&keyexp);
```

For the AES hardware accelerator, I'll focus on the AES encryption function (`AES_encrypt`), and the decryption counterpart (`AES_decrypt`). Listing 1 shows `AES_encrypt`. This listing illustrates an AES encryption round's operations (line 17–36 and 43–62), and the final round (line 65–84). The AES state's 128 bits are captured in four 32-bit variables, `t0` through `t3`, and `s0` through `s3`. The precomputed roundkeys are stored in an array `rk[ ]`. During regular rounds, the AES substitution/permutation operations are implemented with four lookup tables, `Te0[ ]` through `Te3[ ]`. These lookup tables are called T-boxes. Every T-box takes one byte of the state as input and produces 32 bits of output data. The output state is obtained by adding together all T-box outputs with the roundkey. For example, the first output state variable, `t0`, is

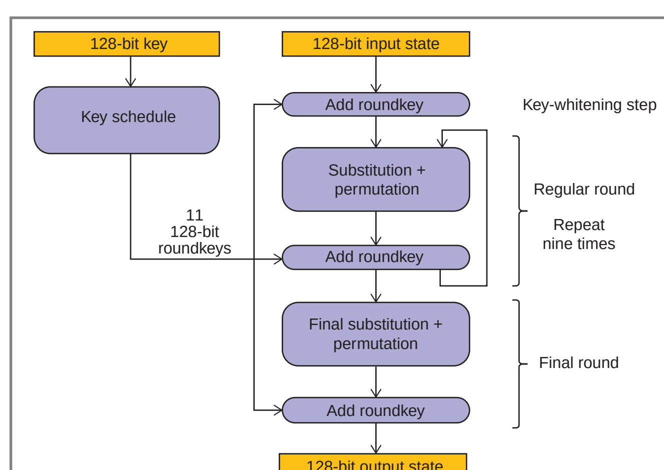


Figure 2—Encryption/decryption in AES-128 combines a 128-bit key with a 128-bit input through a series of 10 rounds. Each round adds a roundkey and mixes up the result through a substitution/permutation block. The roundkeys are derived from the 128-bit input key. Each round uses its own roundkey.

Listing 1—This is a T-box implementation of AES encryption.

```
void AES_encrypt(const unsigned char *in,
                 unsigned char       *out,
                 const AES_KEY       *key) {

    const u32 *rk;
    u32 s0, s1, s2, s3, t0, t1, t2, t3;
    int r;

    rk = key->rd_key;
    s0 = GETU32(in     ) ^ rk[0];
    s1 = GETU32(in +  4) ^ rk[1];
    s2 = GETU32(in +  8) ^ rk[2];
    s3 = GETU32(in + 12) ^ rk[3];

    r = key->rounds >> 1;
    for (;;) {
        t0 = Te0[(s0 >> 24)       ] ^
             Te1[(s1 >> 16) & 0xff] ^
             Te2[(s2 >>  8) & 0xff] ^
             Te3[(s3      ) & 0xff] ^
             rk[4];
        t1 = Te0[(s1 >> 24)       ] ^
             Te1[(s2 >> 16) & 0xff] ^
             Te2[(s3 >>  8) & 0xff] ^
             Te3[(s0      ) & 0xff] ^
             rk[5];
        t2 = Te0[(s2 >> 24)       ] ^
             Te1[(s3 >> 16) & 0xff] ^
             Te2[(s0 >>  8) & 0xff] ^
             Te3[(s1      ) & 0xff] ^
             rk[6];
        t3 = Te0[(s3 >> 24)       ] ^
             Te1[(s0 >> 16) & 0xff] ^
             Te2[(s1 >>  8) & 0xff] ^
             Te3[(s2      ) & 0xff] ^
             rk[7];

        rk += 8;
        if (–r == 0) {
            break;
        }

        s0 = Te0[(t0 >> 24)       ] ^
             Te1[(t1 >> 16) & 0xff] ^
             Te2[(t2 >>  8) & 0xff] ^
             Te3[(t3      ) & 0xff] ^
             rk[0];
        s1 = Te0[(t1 >> 24)       ] ^
             Te1[(t2 >> 16) & 0xff] ^
             Te2[(t3 >>  8) & 0xff] ^
             Te3[(t0      ) & 0xff] ^
             rk[1];
        s2 = Te0[(t2 >> 24)       ] ^
             Te1[(t3 >> 16) & 0xff] ^
             Te2[(t0 >>  8) & 0xff] ^
             Te3[(t1      ) & 0xff] ^
             rk[2];
        s3 = Te0[(t3 >> 24)       ] ^
             Te1[(t0 >> 16) & 0xff] ^
             Te2[(t1 >>  8) & 0xff] ^
             Te3[(t2      ) & 0xff] ^
             rk[3];
    }

    s0 =  (Te4[(t0 >> 24)       ] & 0xff000000) ^
          (Te4[(t1 >> 16) & 0xff] & 0x00ff0000) ^
          (Te4[(t2 >>  8) & 0xff] & 0x0000ff00) ^
          (Te4[(t3      ) & 0xff] & 0x000000ff) ^
          rk[0];
```

Listing continued on p. 51

Listing 1 continued from p. 50

```
70    s1 = (Te4[(t1 >> 24)       ] & 0xff000000) ^
71         (Te4[(t2 >> 16) & 0xff] & 0x00ff0000) ^
72         (Te4[(t3 >>  8) & 0xff] & 0x0000ff00) ^
73         (Te4[(t0      ) & 0xff] & 0x000000ff) ^
74         rk[1];
75    s2 = (Te4[(t2 >> 24)       ] & 0xff000000) ^
76         (Te4[(t3 >> 16) & 0xff] & 0x00ff0000) ^
77         (Te4[(t0 >>  8) & 0xff] & 0x0000ff00) ^
78         (Te4[(t1      ) & 0xff] & 0x000000ff) ^
79         rk[2];
80    s3 = (Te4[(t3 >> 24)       ] & 0xff000000) ^
81         (Te4[(t0 >> 16) & 0xff] & 0x00ff0000) ^
82         (Te4[(t1 >>  8) & 0xff] & 0x0000ff00) ^
83         (Te4[(t2      ) & 0xff] & 0x000000ff) ^
84         rk[3];
85
86    PUTU32(out     , s0);
87    PUTU32(out +  4, s1);
88    PUTU32(out +  8, s2);
89    PUTU32(out + 12, s3);
90 }
```

computed as:

```
t0 = Te0[(s0 >> 24)     ] ^
     Te1[(s1 >> 16) & 0xff] ^
     Te2[(s2 >> 8) & 0xff] ^
     Te3[(s3    ) & 0xff] ^ rk[4];
```

This expression selects the fourth, third, second, and first byte from each of s0, s1, s2, and s3, and uses these as index into four T-boxes Te0, Te1, Te2, and Te3. The final round (line 27–46) uses only a single T-box, Te4, and the output state is obtained by masking individual bytes from the T-box output:

```
s0 = (Te4[(t0 >> 24)     ] & 0xff000000) ^
     (Te4[(t1 >> 16) & 0xff] & 0x00ff0000) ^
     (Te4[(t2 >> 8) & 0xff] & 0x0000ff00) ^
     (Te4[(t3    ) & 0xff] & 0x000000ff) ^ rk[0];
```

The analysis of these expressions is relevant for the hardware accelerator's design, which will require the hardware equivalent of AES_encrypt. For example, in hardware, selecting a byte out of a word is easy, and is implemented with simple wiring. It does not require any shifting or ANDing! Also, a hardware implementation does not need to store the lookup tables Te0 through Te4 in main memory. It is much easier to implement these as dedicated lookup tables (i.e., ROM). Optimizations such as these, in combination with the parallel nature of hardware (as opposed to sequential software), will ensure the hardware accelerator computes AES encryption much quicker than the software implementation.

ALTERA DE0 NANO & NIOS II SOFTCORE CPU

I will use a small, low-cost FPGA prototyping kit to demonstrate the AES hardware accelerator. Terasic Technologies's DE0 Nano development and education board costs $79 and is a convenient and easy way to start with FPGA prototyping. The DE0 Nano

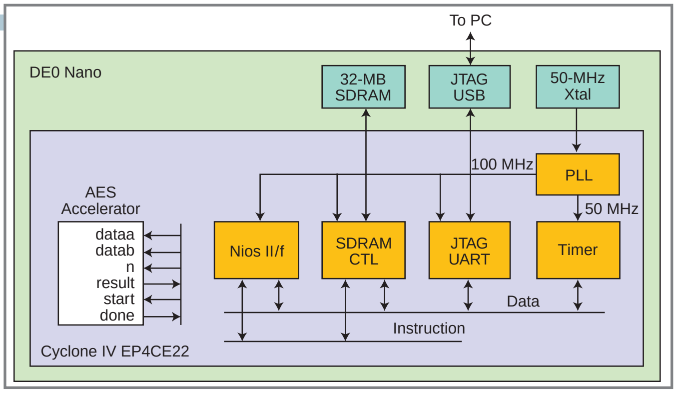


Figure 3—The DE0 Nano kit contains an Altera Cyclone IV FPGA and several peripherals including 32-MB SDRAM. For this project, the FPGA is configured with a Nios II/f embedded core, an SDRAM controller, a JTAG UART, and a Timer module. On-chip PLL generates a 100-MHz clock for the CPU, the memory controller, and the JTAG UART, while the Timer module is driven from a 50-MHz clock. The AES hardware accelerator is attached through the Nios II/f processor's custom-instruction interface.

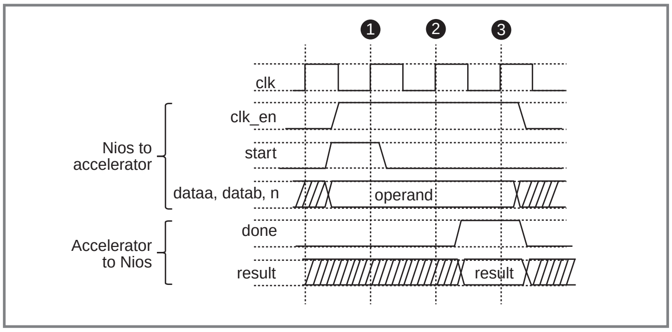


Figure 4—This graph shows the timing of the Nios II/f custom-instruction interface. At clock edge 1, the operands (dataa, datab, and n) are available to the hardware accelerator and execution is started. At clock edge 2, the hardware accelerator is computing. At clock edge 3, the hardware accelerator returns the result to the Nios II/f core and signals completion with a done pulse. A custom-instruction interface can take a variable number of clock cycles.

includes an Altera Cyclone IV FPGA, 32-MB SDRAM, an off-chip crystal oscillator, a JTAG USB interface, and several other peripheral components. The kit is powered through USB.

The FPGA is designed with a baseline configuration including a Nios II/f microprocessor, an SDRAM controller, a UART, and a Timer module (see Figure 3). The CPU, SDRAM controller, and UART run a 100-MHz clock, while the Timer runs a 50-MHz clock. The Nios II/f core is a pipelined RISC processor with dynamic branch prediction, configurable data/instruction cache, and dedicated datapath elements (e.g., a hardware multiplier/divider and barrel shifter). The processor's expected performance is close to one instruction per clock cycle (at 100-MHz clock). On his webpage, *Running µClinux on Terasic DE0-Nano Altera Board*, Tony Frangieh from Virginia Tech's Bradley Department of Electrical and Computer Engineering, demonstrates how to run µClinux on this configuration. This is a nice experiment for our readers with an interest in µClinux!

I will concentrate on adding a hardware accelerator to this baseline configuration. I used a custom-instruction interface on the Nios II/f core. This interface enables a designer to add new hardware to the Nios II/f processor's pipeline. To the software, the new hardware looks like a new instruction. The new instruction accepts three operands: two 32-bit fields `data` and `datab`, and an optional 8-bit field `n`. The new instruction's output is returned through a 32-bit field `result`.

Figure 4 illustrates the timing of the custom-instruction interface. Besides the operand and result fields, three control signals (`start`, `done`, `clk_en`) synchronize the hardware accelerator to the software program. The `start` signal indicates when the Nios II/f issues a new custom instruction to the hardware accelerator. At clock edge 1, the operands are available and the Nios II/f asserts start. The hardware accelerator starts the computation and obtains a result by clock edge 3. The hardware accelerator then returns the computation's result and asserts `done`. This interface can easily

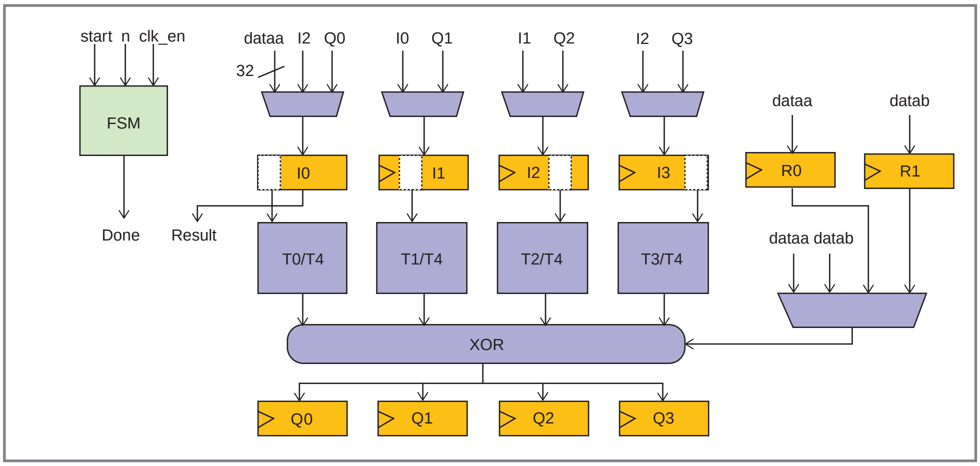


Figure 5—This is the structure of the AES hardware accelerator. Plaintext is loaded in registers I0 to I3. Ciphertext is produced in registers Q0 to Q3. The encryption operation takes four clock cycles and uses four T-box tables (T0 to T4) in parallel. After one round, the Q registers are copied to the I registers and the next round can start. The accelerator is connected to the custom-instruction interface (data, datab, result, n, clk_en, start, and done).

Instruction	Operands			Result	Latency	Function
	n	dataa	datab	result	(cycles)	
LOAD	1	input	–	I0	1	Load new input word
UPDATE	2	–	–	I0	1	Move Q to I
RK	3	Rk0	Rk1	I0	1	Load two roundkey words
ENC	4	Rk2	Rk3	I0	4	Encrypt one round
DEC	5	Rk2	Rk3	I0	4	Decrypt one round
ENCLAST	6	Rk2	Rk3	I0	4	Encrypt last round
DECLAST	7	Rk2	Rk3	I0	4	Decrypt last round

Table 1—This is the custom instruction definition for the AES hardware accelerator.

10th International System-on-Chip (SoC)

Conference, Exhibit & Workshops

October 24 & 25, 2012 – Hilton Irvine/Orange County Airport

www.SoCconference.com

EARLY BIRD REGISTRATION IS NOW OPEN!

Don't Miss Out!

Platinum Sponsors

Microsemi

intel

UC Irvine Extension

- Table-Top Exhibit (Free Passes)
- Memory Trends & Technologies
- Analog & Mixed-Signal Designs
- Green Chips
- Sub 20nm Design Challenges
- 3-D ICs
- Heterogeneous Multi-Core Designs
- Multicore Software Development
- SoC Design & Verification
- Many Door Prizes

- SOI vs. CMOS
- FPGAs –Trends & Designs
- Multicore SoC Platforms
- Memristors
- Network-on-Chips (NoCs)
- Informative Panels
- Mixed-Signal SoCs
- IEEE Student Design Contest
- Networking Opportunities
- And Much More. . .

Keynote Speakers

Altera
Dr. Mike Peng Li,
Fellow, Altera Corporation.

HP
Dr. Janice Nickel,
Cognitive Systems Laboratory at Hewlett Packard Laboratories.

Microsemi
Jim Aralis,
Chief Technology Officer (CTO),
and Vice President of R&D.

JPL
Dr. Reza Ghaffarian,
Jet Propulsion Laboratory,
California Institute of Technology.

UCI
Dr. Payam Heydari,
University of California Irvine.

Promote Your Company, Technology, Products & Services at
The Most Informative, Targeted & Affordably Priced
Chip Design Conference, Exhibit & Workshops of the Year!

Selected Participating Companies & Universities

For More Information or Questions, Please Contact the SoC Conference Organizing Committee at:
SoC@SoCconference.com or (949) 851-1714

www.SavantCompany.com & www.SoCconference.com accommodate variable instruction latencies so a hardware accelerator can use additional clock cycles, if needed. Of course, when the hardware accelerator is computing, the Nios II/f pipeline will be stalled. Therefore, custom instruction latencies should be kept small.

AES ENCRYPT/DECRYPT HARDWARE ACCELERATOR

To design the hardware accelerator, the C function shown in Listing 1 must be mapped into hardware. In addition, the hardware interface must match the custom-instruction interface shown in Figure 4. Figure 5 shows the AES hardware accelerator's structure. A set of 32-bit input registers, I0 to I3, is loaded with plaintext. Next, a round's output is computed and stored in a set of 32-bit output registers Q0 to Q3. The computation is done through T-box tables implemented as hardware lookup tables. Each table is a 256-element array of 32 bits. The datapath shown in Figure 4 can compute the output of one round in four clock cycles. In each clock cycle, one output word is computed and stored in one of the Q registers (Q0 to Q3). Computing a Q register also requires a roundkey. Half of the roundkey bits are stored in R0 and R1 before a round computation starts. The other half is provided as input data to the encryption instruction (dataa, datab). When all Q registers are computed, they are copied back to the I registers to start the next round. Furthermore, the roundkey registers need to be updated with two fresh roundkey values. Thus, a minimum of six clock cycles (4 + 2) is needed for a single round's data handling. This is still a significant speedup over the software implementation of an AES round (lines 17–36 in Listing 1).

The hardware accelerator is controlled using a finite state machine, which is driven from the Nios II/f through the custom-instruction interface. When a suitable instruction code is decoded through the inputs n, clk_en, and start, the FSM will initiate execution of the selected custom instruction. Table 1 shows the set of custom instructions implemented by the hardware accelerator. The instructions take care of data movement between Nios II/f and the hardware accelerator (LOAD, UPDATE,

Listing 2—This listing shows a custom instruction version of AES encryption.

```
void AES_encrypt_CI(const unsigned char *in,
                    unsigned char *out,
                    const AES_KEY *key) {
    const u32 *rk;
    u32 s0, s1, s2, s3;
    int r;
    rk = key->rd_key;

    // Key Whitening Step
    s0 = GETU32(in     ) ^ rk[0];
    s1 = GETU32(in +  4) ^ rk[1];
    s2 = GETU32(in +  8) ^ rk[2];
    s3 = GETU32(in + 12) ^ rk[3];

    // Load AES State
    CUSTOM_INS(LOADINS, s0, 0);
    CUSTOM_INS(LOADINS, s1, 0);
    CUSTOM_INS(LOADINS, s2, 0);
    CUSTOM_INS(LOADINS, s3, 0);

    // Round 1
    rk += 4;
    CUSTOM_INS(RKINS,  rk[0], rk[1]);
    CUSTOM_INS(ENCINS, rk[2], rk[3]);

    // Round 2..9
    for (r=0; r<8; r++) {
      rk += 4;
      CUSTOM_INS(UPDATEINS,  rk[0], rk[1]);
      CUSTOM_INS(ENCINS,     rk[2], rk[3]);
    }

    // Last Round
    rk += 4;
    CUSTOM_INS(UPDATEINS,  rk[0], rk[1]);
    CUSTOM_INS(ENCLASTINS, rk[2], rk[3]);

    // Retrieve AES state
    s0 = CUSTOM_INS(UPDATEINS,  0,     0);
    s1 = CUSTOM_INS(LOADINS,    0,     0);
    s2 = CUSTOM_INS(LOADINS,    0,     0);
    s3 = CUSTOM_INS(LOADINS,    0,     0);

    // Return Result
    PUTU32(out     , s0);
    PUTU32(out +  4, s1);
    PUTU32(out +  8, s2);
    PUTU32(out + 12, s3);
}
```

Function	Timing 512 B/1 KB O0	Timing 4 KB/4 KB O0	Timing 512 B/1 KB O3	Timing 4 KB/4 KB O3	Unit
Encrypt (Software Ref)	15452	4890	9692	2988	Cycles
Encrypt (HW Accelerator)	1622	1132	894	454	Cycles
Encrypt speedup	9.5x	4.3x	10.8x	6.6x	
Decrypt (Software Ref)	16550	5208	9744	3266	Cycles
Decrypt (HW Accelerator)	1648	1132	566	454	Cycles
Decrypt speedup	10.0x	4.6x	17.2x	7.2x	
Encrypt key schedule	3960	2464	4924	1006	Cycles
Decrypt key schedule	30548	12074	17548	5922	Cycles

Table 2—This is the state timing for the AES software and the hardware accelerator. The same design executes with two different Nios II/f configurations (1-KB cache versus 4-KB cache) and two different compiler optimization levels (O0 versus O3). Against the best software setting (4-KB cache and O3 optimization level), hardware acceleration still provides a speedup of 6.6 for encryption and 7.2 for decryption.

and RK), and they take care of encryption and decryption (ENC, DEC, ENCLAST, and DECLAST). The AES-128 decryption algorithm is very similar to encryption. It is easy and useful to create a hardware accelerator that can do both decryption and encryption. The accelerator is developed in Verilog and integrated into an Altera Quartus project. The project files are available on *Circuit Cellar*'s FTP site.

An actual encryption or decryption sequence needs to be assembled using the custom-instructions listed in Table 1. Listing 2 shows how the Nios II/f will control the hardware accelerator for encryption. Each call of the form `CUSTOM_INS(n, dataa, datab)` will access the hardware accelerator to execute instruction ID `n` with operands `dataa` and `datab`. Given that Listing 1 and Listing 2 have the same functionality, the hardware accelerator will offer a significant speedup compared to the AES encryption's software version.

PERFORMANCE ANALYSIS

When designing an embedded system's hardware accelerator, you must know how fast the overall design is and how many additional resources will it cost to evaluate the result. In my case, speed is expressed in the number of clock cycles to compute a full AES-128 operation, and cost is expressed in extra FPGA utilization for the accelerator functionality. To measure speed, I count clock cycles using the design's timer. Because the Nios II/f has cache and branch prediction, I cannot take just a single encryption function call's timing. A better strategy is to execute the encryption several times, measure each instance, and take the average or the steady-state value. Sample timing measurement code looks as follows:

```
for (i=0; i<10; i++) {
  t1[i]=alt_timestamp();
  AES_encrypt_CI(txt, dout, &keyexp);
  t1[i]=alt_timestamp()-t1[i];
}
```

After execution, `t1[ ]` contains a list of 10 timing values and the last few values represent the steady-state value.

Table 2 shows the AES encryption and decryption's measured execution times, in software and using hardware acceleration. The different columns represent different processor/compiler configurations. I used two different cache sizes on Nios II/f. One has a 512-byte data cache and a 1-KB instruction cache. The other has a 4-KB data cache and a 4-KB instruction cache. I also used two different C compiler optimization levels: no optimization (O0) and best-case performance optimization (O3). Table 2 shows hardware acceleration provides speedup factors for 4.3x to 10.8x for encryption, and 4.6x to 17.2x for decryption. This performance improvement comes at the cost of additional FPGA resource usage. Table 3 shows the full system's logic complexity (as shown in Figure 3), with and without the hardware accelerator and for the different cache configurations. This shows the overhead of AES hardware acceleration in the complete system is relatively minor, compared to the performance improvement.

	512 B/1-KB Cache	4 KB/4-KB Cache	Unit
Baseline design	3665	3681	Logic elements (LEs)
	12	18	Memory blocks (M9K)
Accelerator design	4452	4468	Logic elements (LEs)
	26	32	Memory blocks (M9K)

Table 3—This is the resource usage for the baseline design and for the hardware acceleration design. FPGA resource consumption is measured in logic elements (LEs) and on-chip memory blocks (M9K).

GIVE FPGA A TRY!

There you have it—a hardware accelerator for data encryption. If you have never worked with an FPGA, I recommend you try it. The latest generation of FPGA design environments is very good, and offers many bells and whistles that keep embedded designers happy. Professionally, I use this component extensively for teaching, for hardware-oriented projects, and for firmware-oriented projects such as this one. In my next article, I plan to continue on the same platform and implement a true random number generator (TRNG). I will show that a TRNG cannot be implemented with software alone, and that true randomness can only be achieved in hardware.

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.

RESOURCES

Altera Corp., "Nios II Custom Instruction User Guide," 2011, www.altera.com/literature/ug/ug_nios2_custom_instruction.pdf.

———, "Embedded Design Handbook," 2011, www.altera.com/literature/hb/nios2/edh_ed_handbook.pdf.

T. Frangieh, "Running μClinux on Terasic DE0-Nano Altera Board," Configurable Computing Wiki, Virginia Tech University, 2011, www.ccm.ece.vt.edu/twiki/bin/view/Main/LinuxOnNIOS2InstallationDE0Nano.

National Institute of Standards and Technology (NIST), Computer Security Resource Center, "Federal Information Processing Standard 197: Announcing the Advanced Encryption Standard (AES)," 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

SOURCES

Quartus II design software and Cyclone IV FPGA
Altera Corp. | www.altera.com

DE0-Nano development and education board
Terasic Technologies, Inc. | www.terasic.com.tw

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:

—

Hybrid Computing on an FPGA
by Bruce Land
***Circuit Cellar* 208, 2007**
This article explains how to simulate the parallel functions of an analog computer on an FPGA. Topics: FPGA, CPU, Differential Analyzer

—

Designing with the Nios (Part 1)
Second-Order, Closed-Loop Servo Control
by George Martin
***Circuit Cellar* 167, 2004**
This article describes how a Nios was used to design a second-order, closed-loop servo control. Topics: FPGA, Nios, Motion Control

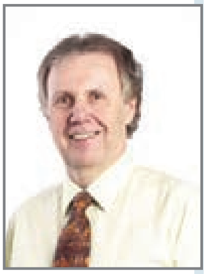
—

Designing with the Nios (Part 2)
System Enhancement
by George Martin
***Circuit Cellar* 168, 2004**
In the first part of this article series, the groundwork was laid for a Nios-based two-axis motion control system. This article discusses velocity feedback and preps you on the software needed to drive it. Topics: FPGA, Nios, Motion Control

—

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

by George Novacek (Canada)



Project Configuration Control

Family Tree Drawing & Document Archiving

Family Tree drawing is an indispensable tool for maintaining project configuration control, document number assignment, project progress tracking, and document archiving. Documents and drawings are easily accessible, especially during design changes.

Configuration control is an important undertaking during an engineering development project. Family Tree drawing is one control method. Fundamentally, a Family Tree drawing is an organizational chart-like breakdown that lists documents and drawings needed in the course of the project.

TAKING ROOT

At the beginning of any project, engineers need to develop a preliminary schedule and cost estimation. If developed concurrently, a Family Tree drawing can help visualize the scope of work and increase time and cost estimate accuracy. Figure 1 is a typical Family Tree drawing. There are no rules guiding its form. Possibilities are endless and companies often have their own preferences. I like mine. It's easy to understand and navigate. Documents and drawings are usually designated by numbers, not names, and a Family Tree drawing helps quickly find them and identify which related documents will be affected if a design change occurs.

Let's say I am starting a project called Controller. I need to assign it a model number. I like to use thousands, so in this case, let's say, 5000. The drawing with this number is at the top of the Family Tree, showing the final assembly of the product as shipped. Then, numbers are assigned to individual groups: 5100 to hardware; 5200 to software; 5300 to configuration control and quality assurance (QA) plus reliability, maintainability, and safety documents (RM&S); 5400 to acceptance test procedure (ATP) and all the integration and testing issues; and, finally, 5500 to mechanical parts (e.g., cabinet, packaging, etc.) I reiterate that the structure is not carved in stone and can be customized, based on the project and the documentation needed. It is a summary of the complete documentation for the given product. Program management (PM) and customer correspondence can also be included in separate groups, but the PM commonly keeps files away from engineering.

Based on experience, I can develop a preliminary list of documentation. For example, the hardware group will always contain a system black box diagram with all the external interfaces, then a white box diagram for the controller, comprising functional blocks (e.g., power supply, I/O, memory, etc.). This list is followed by the schematics. If the hardware is to be developed according to DO-254 or any other standard, the list of necessary documentation is usually provided by the standard or the customer. Each subassembly's bill of material (BOM) is often entered. Each Family Tree block can have branches to other blocks to show supporting documentation, sub-subassemblies, and so on. There are no limitations as long as consistency is maintained. The Family Tree drawing is a living document that should only be modified by one authorized person. That person is also responsible for entering new documents and their numbers.

SOFTWARE DOCUMENTATION

Software documentation starts with the executable code at the top level and is followed—in this case, in compliance with DO-178 standard—with software aspects for certification (SAFC), system requirements (SYRD), software requirements (SRD), software design description (SDD), source code, and so forth. The last item is frequently the

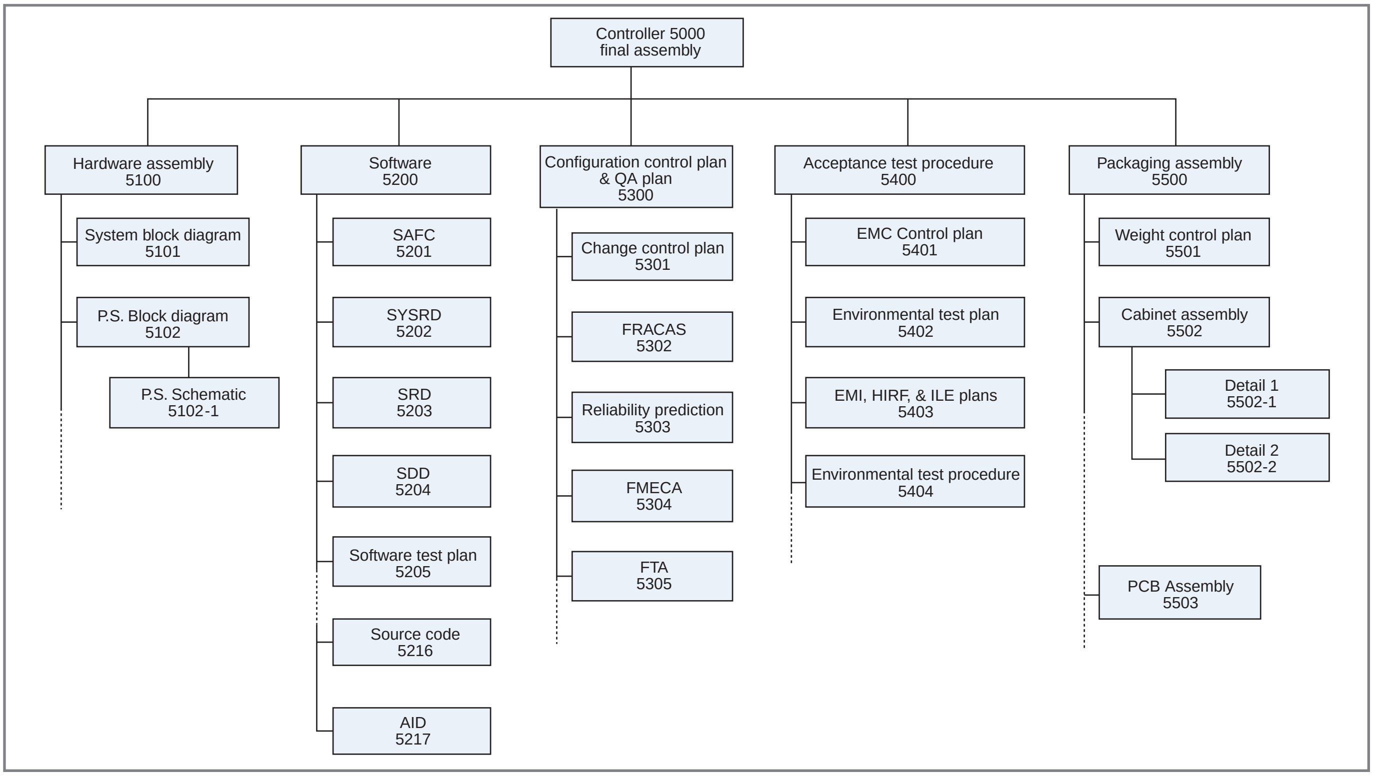


Figure 1—This is an example of a Family Tree drawing.

altered item drawing (AID), which defines the memory or the microcontroller chip to be loaded with the firmware. For example, it identifies the IC by its full purchasing number plus the medium containing the code. It also gives the cyclical redundancy check (CRC) value to verify code accuracy.

I have always resisted providing customers with schematic diagrams, background calculations, and source code. First, the product has been developed with a lot of experience and proprietary know-how, which is not paid for by the customer. There is rarely a legitimate need for the customer to have the design data—you are the hired expert! Second, once given away, the data always finds its way to the competition regardless of nondisclosure or other confidentiality agreements. And third, access to diagrams and the source code awakens many budding designers who work for the customer. They don't have design or financial responsibility for it, yet they tie up engineers in time-wasting arguments about their design suggestions. Showing the customer what you're doing at design reviews is one thing. Allowing the customer to take home and pick at the design details often comes back to haunt you.

Category 5300 includes the configuration control plan and QA plan (sometimes combined into one), which are usually generic company documents that are modified as needed for the project. Other documents may be included in this branch, such as reliability prediction, failure mode and criticality analysis (FMECA), and fault tree analysis (FTA).

TESTING & PLANNING

Two important documents most customers and certificating agencies want to review are the change control plan and the fault reporting and corrective action system (FRACAS) plan. The change control plan defines how design changes will be processed, including who has the authority to initiate a design change, how it is reflected in the version control, who establishes whether a recertification or a user information bulletin is needed, who is liable, and so forth. FRACAS, on the other hand, specifies how field problems are tracked and resolved. It defines how they are reported, who is responsible, and what the process is if the resolution requires a design change.

RM&S analyses have to be generated at the project's beginning and updated concurrently with the development. Based on the black box (system block diagram) and white box (controller internal block diagram), an experienced engineer can estimate the reliability and assign it to the block-level FMECA. Then, a decision can be made how to proceed to satisfy the customer's reliability and safety requirements.

Similarly, electromagnetic compatibility (EMC) issues need to be addressed early in the program by the EMC control plan. As I mentioned before, while engineers often find these analyses boring and time consuming, if they are not maintained, you may fail to meet the design goals and cause future headaches.

The acceptance test procedure (ATP) is a key document in the test and integration group 5400. Every product is subjected to it before being shipped. The ATP results must show every specification requirement is satisfied. The work on the ATP should start during the compliance statement preparation. Whenever responding "compliance by test," consider how the test will be performed and if it can be performed. Otherwise, you may discover that some requirement is impossible or too difficult to test. Most ATPs today are performed by automatic testers. The software running the ATP often needs the same level of QA as the product.

The EMC control plan focuses on

> **"Each Family Tree block can have branches to other blocks to show supporting documentation, sub-subassemblies, and so on. There are no limitations as long as consistency is maintained. The Family Tree drawing is a living document that should only be modified by one authorized person."**

susceptibility as well as ways to control unwanted electromagnetic emissions. Packaging, shielding, filtering, and PCB layout all need to be addressed in advance. The plans preparation is mainly based on prior experience. Performance verifications under EMI, high-intensity radiated field (HIRF), indirect lighting effects (ILE), environmental conditions, accelerated testing, and others are defined in their respective plans to satisfy the specification. They are followed by procedures and results reports. Similarly, all hardware and software integration testing plans, procedures, and results come under this umbrella, although all software-specific tests may be in the software group.

Mechanical issues (e.g., packaging) belong in the 5500 category. In addition to mechanical drawings, the 5500 category may also include reports (e.g., weight and its reduction plans, electrical loading, etc.).

BRANCHING OUT

The Family Tree drawing is an indispensable tool that facilitates the preparation of a customer proposal, development of the work breakdown, schedule, cost, and resource planning. It is equally instrumental in configuration control, progress tracking, and archiving. The drawing should be hanging on the wall for everyone to see which documents and drawings have been finished, what their revision numbers are, what numbers the new documents have been assigned, and which documents are to be prepared next.

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

RESOURCES

G. Novacek, "Accelerated Testing," *Circuit Cellar* 262, 2012.

———, "Project Development (Part 1): Plans, Schedules, and Task Management," *Circuit Cellar* 264, 2012.

———, "Project Development (Part 2): Development Process, Milestones, and Design Reviews," *Circuit Cellar* 265, 2012.

by George Martin (USA)

Software & Design File Organization

A simple directory structure can track your project's correspondence, datasheets, specifications, hardware and software designs, and simulation results. This article describes one method of tracking information using a mechanical design project.

I am envious of some of the recent engineers whose interviews have been featured in *Circuit Cellar*. In these interviews, photos of the designer's labs look so neat and clean. Mine is a mess. I suspect some clean up occurred for the photographs, but maybe not. While my hardware situation might not be so well organized, my software is kept rather neat and tidy. This article describes how I keep track of it all. The value in my technique extends to a mechanical design project and explores how a big software suite might fit into all this.

PROJECT ORGANIZATION

When you have a nontrivial project to keep organized, correspondence, datasheets, specifications, hardware designs (schematics), software designs (code), meeting notes, simulation results, and many other pieces of information belong to the product. I start it all with a simple directory structure (see Figure 1).

If this file layout looks obvious, then I think it's a good design. Let's look into this structure and see how it works. This directory structure is on my D drive. That way, I can back it up (copy everything) to another drive in the network or copy it to a CD/DVD ROM. I typically have more than one project for a each customer. These projects can span decades. And several projects may be active at the same time. Under a project, the next level of folders varies

```
1. CustomerName
   a. Project 1
           i. Correspondence
                  1. My correspondence
                  2. From John Q
                  3. From Bill Z
           ii. Documentation
                  1. TI
                  2. Freescale
                  3. ON Semi
           iii. CODE
                  1. TI Compiler
                        a. Prototype
                               i. Latest
                               ii. V00-00
                               iii. V00-01 Released 12May2012
                        b. Production
                               i. Latest
                               ii. V01-00
                  2. A different compiler
           iv. ProtelDXP (Schematics)
                  1. Latest
                  2. Rev A
                  3. Rev B
           v. Simulation
           vi. Scope Dumps
   2. Project 2
```

Figure 1—This is an example of a simple directory structure.


2013 International CES
THE GLOBAL STAGE FOR INNOVATION
PRODUCED BY CEA
When new ideas collide, the charge is unmistakable.
TUESDAY, JANUARY 8—FRIDAY, JANUARY 11, 2013 * LAS VEGAS, NEVADA * REGISTER AT CESWEB.ORG
CES
REGISTER NOW

```
1. MyCompanyName
  a. Manufacturer # 1
        i. Product A
        ii. Product B
        iii. Product C
  b. Manufacturer # 2
        i. Product D
        ii. Product E
        iii. Product F
```

Figure 2—This is an example of a directory used to differentiate project details.

greatly with each project. Some projects involve simple schematics or artwork. Others are more complicated and will have many folders (e.g., documents) at this level, which can be organized in many ways. If you only have one device from Texas Instruments (TI), it's probably not worth giving it a separate folder. If you're analyzing ADCs, you probably want a folder for all the different converters. These could hold datasheets, application notes, and test results.

I use Altium's Design Explorer (DXP) integration platform for schematics, parts lists, and artwork. This program is easy to use. I simply make a directory for each revision and place the appropriate file in the appropriate directory. When it comes to software, the story is much different. If you have more than one processor on a project, you will have more than one development environment. You need to track revisions in each of these environments so they will be compatible with the compiler toolset.

Today, I believe every compiler comes with its own integrated development environment (IDE). But that's where the friendliness ends. The problem is, if you make a new code revision and put it in a new folder, some of the tools won't let you easily work with the new folder. I've adopted a method that seems to work with most of the compilers and their IDEs. I create a project in the "Latest" directory in the directory structure shown in Figure 1. I do this with the standard "Create a New Project" command from inside the IDE. Then, whenever I release code, I copy the latest directory into a directory (named "V00-00" in my example). That version can be released and preserved forever. As releases continue, the version numbers are increased. On more complicated projects, my version numbers are V00-00-00. This gives me finer control over the changes, and the compiler and IDE know where to find the code to compile. It's in the latest folder.

If I need to roll back to a previous version, I save where I am, delete all the files in the latest directory, and copy the older version into that latest directory. Version control systems would do this more automatically, but not all compilers and IDEs support version control systems.

DETAIL DIFFERENTIATION

I recently needed to communicate with a sheet metal design and fabrication shop about a series of products it was designing for me. These products had much in common. It soon became clear we could get confused with all the details. The designer did not know the products and they started to blur together. I adopted the directory structure shown in Figure 2.

We created a directory with my company's name. All of *my* products are contained in this directory structure. We then created two folders; one for each of the two manufacturers. All of my products are available in one of the two manufacturers' directories. In each manufacturer's directory, we created folders for each of my products (by my part number).

The designer could put all the mechanical design information in each of the product folders. He was using SolidWorks, which had a preferred directory structure. I could now refer to my designs by my part number and the designer knew exactly which part I was talking about. A little organization can make life simpler. Perhaps I should clean my desk.

"If you have more than one processor on a project, you will have more than one development environment. You need to track revisions in each of these environments so they will be compatible with the compiler toolset."

```
1. C:TexasInstruments
     a. BLE-CC254x-1.2
         i. Accessories
               1. Drivers
               2. HexFiles
         ii. Components
               1. Ble
                    a. Controller
                    b. hci
                    c. host
                    d. include
               2. Hal
                    a. common
                    b. target
                    c. include
               3. Osal
                    a. common
                    b. mcu
                    c. include
               4. services
         iii. Documents
         iv. Projects
               1. ble
                    a. Blood Pressure
                    b. TimeApp
```

Figure 3—This is the directory structure for the Bluetooth BLE toolset standard install.

LIFE IN THE REAL WORLD

I have several computers all with much the same set of tools installed. This setup gives me several benefits. First, if I get stuck and am waiting for an answer from one customer, I can switch to a different machine and work on a different project. Second, I can easily back up (copy) the project directory described in Figure 2 on several machines. So, if one machine has a problem, I can use a different machine to work. Today's equipment has fewer problems than 10 or 20 years ago, but they do crash.

Finally, it's easy to make a backup of all the customer's work, put it on a CD/DVD or USB memory device, and send it out.

PROBLEM SOLVING

I've been working on a Bluetooth low-energy (BLE) project that uses an IAR Systems compiler. IAR makes great compilers, and the CPU was an 8051, so it's been around for a while and it is stable. The complete design package uses a stack to manage the BLE radio and a simple real-time operating system (RTOS) to perform the design's real-time portion.

So, for a small CPU, this is a really big toolset. A lot of code

```
1. CustomerName
      a. ProjectName
            i. Code
                  1. TI-BLE-CC254x-1.2
```

Figure 4—This is an example of a Texas Instruments tools-based directory structure.

comes with the compiler. Simple example projects compile up to 100 KB of code. The directory structure for the standard install of this toolset is shown in Figure 3.

All the example projects that run on the evaluation board are in the /projects/ble directory. These projects use the files located in the BLE-CC254*x*-1.2/components directory. I add my project work to the projects' directories on the C drive. But, I lose the technique for revisions and whenever TI releases a new version of the toolset, I need to somehow start over.

At first, I started to create a new project directory on the D drive as I've been describing. But all the proper links to the files needed to be defined and included. I soon realized this was a difficult, if not impossible, approach.

I then asked TI if I could install the tools in my project directory. The company confirmed this would work. So, I now have a directory structure that looks like the one shown in Figure 4.

Then, in the TI-BLE-CC254*x*-1.2 directory, I started my directory structure using different versions. If TI comes out with a new compiler, I will place it as another entry in the Code directory. I then copy my code from the 1.2 compiler version to the new compiler version. I've preserved all the intricate links the tools have built into them.

I hope you found this discussion useful. Rather than show you the results and give you a pile of schematics and code files, I'm trying to build the foundation to help you succeed. My methods are just one approach. If you find better ones, let us know.

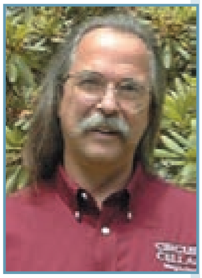
George Martin (gmm50@att.net) began his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and co-founded a design and manufacturing firm (www.embedded-designer.com). His designs typically include servo-motion control, graphical input and output, data acquisition, 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.

SOURCES

Design Explorer (DXP) integration platform
Altium, Ltd. | www.altium.com

Bluetooth low energy (BLE)
Bluetooth, Inc. | www.bluetooth.com

by Jeff Bachiochi (USA)



Flowcharting Made Simple

Use the Flowcode Flowcharting IDE to Write Code

It is easier to code if you first create a flowchart, which is a diagram showing a step-by-step process. With a flowchart, you still have to write a machine-recognizable application. But what if your flowchart could create code? This article describes a flowcharting IDE that writes code for many types of microcontrollers.

Even if you're wearing a mandatory helmet and flotation jacket, paddling can be a dangerous sport. One of the main rules to remember if you capsize—and you will—is to "go with the flow." This means you will struggle swimming directly upsteam or even swimming toward the closest shore. Instead, use the current and aim toward the shore a bit further downstream. The river's flow will guide you safely to your destination.

This is how I feel when I get stuck writing an application. If I haven't drawn out my ideas prior to writing any code, I find I'll often get capsized and need to stop and gather my thoughts. Flowcharting always helps me regain my bearings and get headed in the right direction. Flowcharting is a graphical way of expressing the steps necessary to achieve a goal. Every flowchart contains two basic graphical shapes: the square (i.e., process) and the diamond (i.e., decision). You've likely used a flowchart in the past, so I'm not going to dwell on the process.

If you've spent any amount of time flowcharting your application, you may have asked yourself why? After all, flowcharting doesn't get your application written. Its graphical nature is more of a visual aid or guide to how your code might be written. And, after it's finished, you must rewrite it in a machine-recognizable form. What a waste of time. Actually, I believe spending a little extra time laying out your plan of action (i.e., flowcharting) before beginning to code makes the coding easier. But, what if your flowchart could actually produce your code? ARM, AVR, and PIC microcontroller users now have this ability thanks to Flowcode, an integrated development environment (IDE) from Matrix Multimedia, Ltd. (MML).

FLOWCODE IDE

MML divided its process into four areas: design, simulate, test, and deploy. All operations are controlled through a common IDE that consists of a single screen and includes a number of toolbars and window panes. The screen capture in Photo 1 describes Flowcode.

The main toolbar enables you to handle files and control the simulation and download functions. The icon toolbar contains supported drag and drop flowchart graphical icons for your application. The component toolbar contains supported components that can be connected to your microcontroller. These are grouped into categories of similar functions. I've vertically repositioned this normally horizontal toolbar to enable the screen to be squished without losing functions on the right side of the screen.

The flowchart pane (in the center) shows the application's main process for my project. To the right is a properties pane, which shows the application's items. At the bottom of the screen is a front panel view of the application items. This can be modified to simulate a mockup of your application's actual layout. Not shown are a number of other panes you can enable showing microcontroller pinout as well as variables and call stack

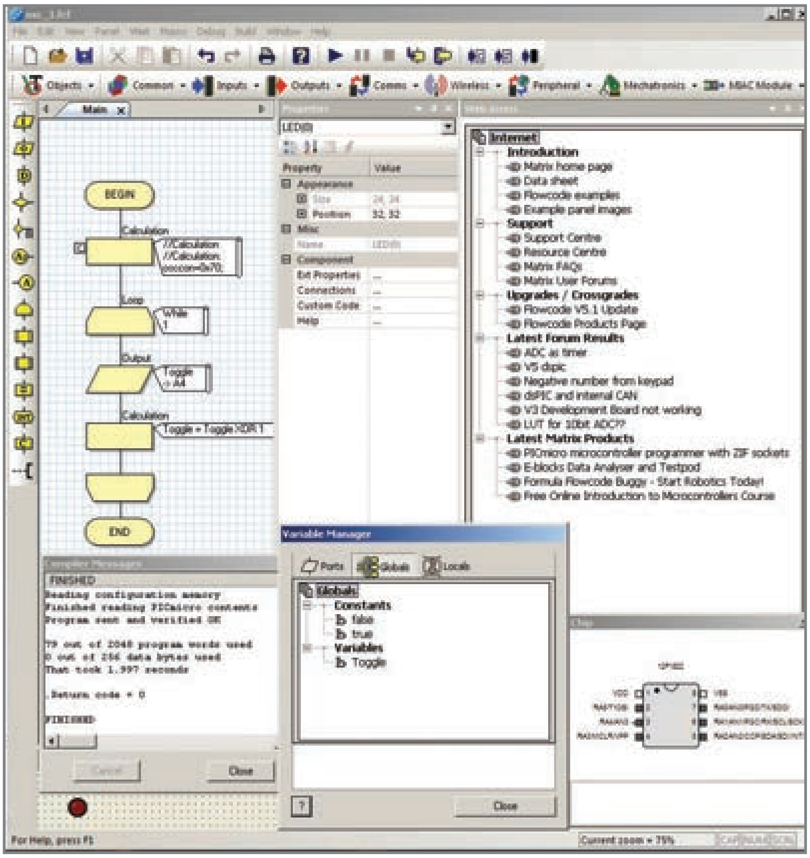


Photo 1—The IDE can be used to reach all of Flowcode's operations. I've opened many of the various operations' panes. You can see the basic flowchart in the upper left pane and the compiler message pane in the lower left.

you've chosen and also how to bend the rules (i.e., use some C or Assembly instructions to bypass the standard code).

Getting started can be daunting. I say this because, after choosing a microcontroller via "Project Options" (see Photo 2), you can attempt to operate with the default settings but you may need to set some specifics (e.g., a particular clock speed). If you click on the "Configure Chip," there are 17 more options that, to a non-programmer, are enough to get you to say "uncle." An incorrect choice could stump a new user for hours. For instance, if you don't know your oscillator selection's requirement, your code may never get executed because it doesn't have a running clock!

In this instance, the default value is for an external 19.6608-MHz crystal, which is on MML's E-blocks multiprogrammer board. The default oscillator selection is for an external oscillator, which must be changed to use the EB-006's crystal. Confusion aside, I have selected the internal oscillator with PLL. While the clock speed box has no selection for the maximum 32-MHz clock speed, I can type it in. However, it doesn't use this information to set up the OSCCON register (which needs to be set to 8 MHz for the PLL to bump it up to 32 MHz.) So, when I build and download the project, it runs 16 times slower than expected (OSCCON defaults to 500 kHz). Flowcode does not handle OSCCON (and many internal registers) directly, you

panes used within your application and during simulation.

All toolbars and panes can be enabled/disabled and docked or left floating, so you can customize your own personal workspace. Flowchart's real magic begins when you "build" a project that has a completed flowchart. This process takes all the information you've given it and produces C code based on the flowchart. Most programmers are familiar with the process of compiling C code into Assembly code and assembling and linking those files into a single HEX file, which is the process needed to properly place the code within your microcontroller. But I'm getting ahead of myself. Let me take you through the process I used to try this on a simple design.

PROJECT OSC

I occasionally need a clock source to stimulate circuits. Sometimes I need a slow repeating trigger, an audio source, or some clocking frequency. Let's take an eight-pin PIC and see what might be practical. I'll start the project using Microchip Technology's PIC12F1822 microcontroller. You can purchase this for approximately $1.30 for one unit. Under the build menu's "Project Options" you can select from an extensive parts library. One note here, the marketing department says, "Flowcode software allows those with little to no programming experience to create complex electronic systems in minutes." While this might be true for most basic designs, to accomplish anything outside of the sometimes limited options, you need to know the specifics of the chip

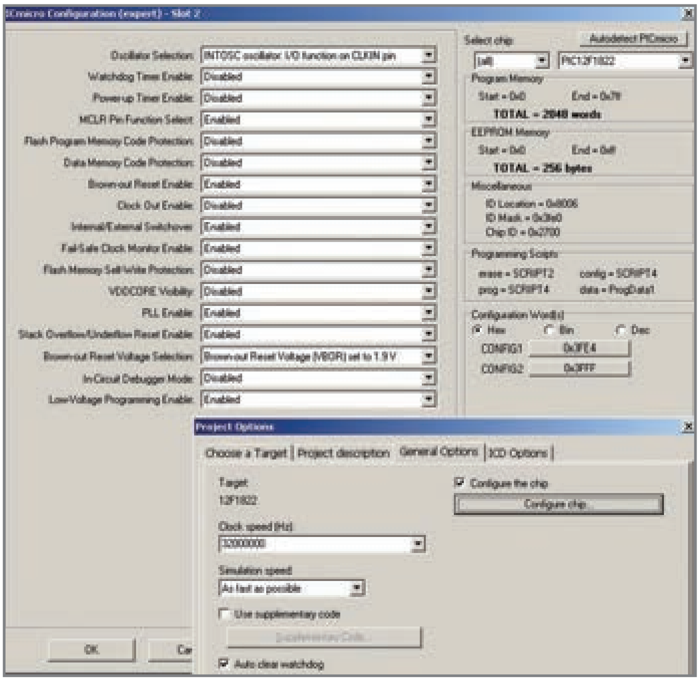


Photo 2—One the first steps is to choose and configure your design's microcontroller. This can be done through the "Project Option" menu item. Clicking on the "Configure Chip" brings up a complicated group of options (i.e., background pane) you will have to navigate (a necessary evil for any programming language).

must add C code (or Assembler statements) to the application to do this (refer to Photo 1).

Even before any code is programmed into a microcontroller, you can use the debugger to simulate this code's execution. This process is controlled by the main toolbar's Start, Pause, Stop, Step Into, and Step Over buttons. Your selection in "Project Options" controls the execution (step) speed. The call stack and variable panes that pop up during debugging help you determine what's actually occurring. Not much is happening with this inline code. The steps within the `while` loop write the `Toggle` value (a Boolean variable) to the output bit PORTA.5 and use the XOR operator to calculate a new `Toggle` value. This repeats endlessly since the `while` test for continuing will always be "True" (the constant "1"). An output device (e.g., the LED), adds some visual feedback on the state of an output. This is especially useful when single stepping, otherwise, due to persistence of vision, fast execution may look like it's on or off all the time.

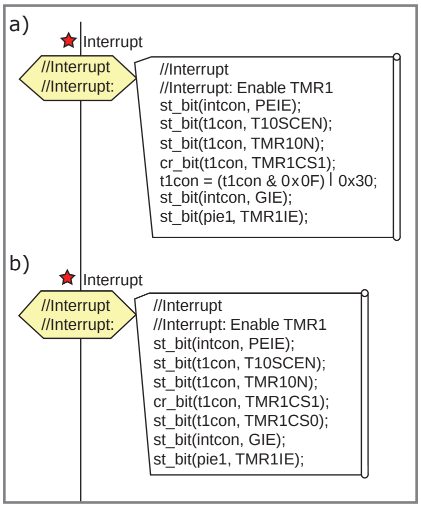


Figure 1—The Timer 1 interrupt is responsible for the output state (i.e., frequency). Calculated values must be reloaded into this counter to vary the time between interrupts. Flowcode's stock component does not provide this option. Another clock source must be selected to achieve maximum timer resolution. These changes must be done through code customization. I've added a second interrupt to the flowchart so you can compare the stock code (**a**) to the changes I've made (**b**).

Once you are happy with the simulation, you can put the code into your chip. Aside from a programmer, the HEX file prepared during the build process is all you need. You can use the E-blocks USB PICmicro multiprogrammer board or Microchip's MPLAB and any of its programmers, if you already own one. When I program this into my PIC12F1822 and reset the device, I can see an approximately 200-kHz output on PORTA.5. Each half cycle is approximately 2.5 µs, which is about 20 instructions (2.5-µs half cycle time/125-ns execution time). This is expected, as higher-level languages produce extra code as language support. You can view the C and Assembly code generated by Flowcode right from the IDE.

VARIABLE OSCILLATOR

Let's give this design some range. If I only toggle a bit, there is a 200-kHz maximum frequency. Adding code only slows things down. So, let's design for a 100-kHz maximum. I also want some way to adjust this frequency. I'll use a couple of potentiometers that can be read as analog inputs to set the frequency. AD0 will set a base frequency from 1 to 100. AD1 will set a (Timer 10) factor ranging from 0 to 3. These factors combined will be used to calculate a timer (reload) value. The timer overflow produces an interrupt, which will be used to handle the output bit's state.

Additional processes are added to the original program to include a few interrupts, one for Timer 1 and one for each analog input. First I'll look at Timer 1. Simply drag and drop IR into the design to add a process. The interrupt component enables me to choose from a list of supported interrupts presented by Flowcode and dependent on the selected chip. After selecting Timer 1 and the interrupt properties, I can define the interrupt's timer setup. Here I choose the clock source and prescaler setting. Note that Flowcode shows what the interrupt frequency interval will be based on the selections made. Also, there is no way to change the 16-bit timer count so its interrupt frequency will be fixed at the 16-bit rollover count (for the source and prescaler selections this is 122 Hz). I want to obtain a variable output frequency (not just the maximum count), which requires additional code to handle the TMR1H and TMR1L registers that can't be directly accessed with Flowcode.

Suppose I could set the timer tick to 5 µs (i.e., a half cycle of 100 kHz) enabling the interrupt routine to toggle the output bit. Loading the timer with 0xFFFF would produce an overflow on the next tick (5 µs) and the interrupt routine would toggle the output bit. Assuming I could do this every 5 µs, I'd have a 100-kHz output. If I then chose to load the timer with 0xFFFE (the next lower value), it would take two ticks 10 µs before the interrupt again toggled the output bit. At this rate, the output would be 50 kHz. What about all those frequencies between 50 and 100 kHz? To achieve more resolution, I need to use a faster tick. There is another choice for the Timer 1 clock source (F_{OSC}, not divided by 4) that can be set in the T1CON register by altering the configuration values set up by Flowcode. In Figure 1, I've added a second Timer 1 interrupt to show the difference between Flowcode's stock translation and what I altered to achieve my goal. I've edited a line to set TMR1CS0, this selects F_{OSC} (not divided by 4) as the source. Notice another line has been eliminated. This sets the prescaler at 1:1 (no prescale). These changes reduce the clock tick to 125 ns. To count to the 5 µs necessary for a 100-kHz output, the timer register must be loaded with a number that will count 40 ticks to get to that same 5 µs. This means 41 ticks will produce a 5.125 µs (i.e., 41 × 125 ns) or 97,561-Hz half cycle time, which is a big improvement in high-frequency resolution.

ANALOG FREQUENCY SELECTION

Rotary pots are used to enable a user to request a particular frequency (and then figure out what value to load into Timer 1 to produce interrupts at the correct rate). By placing 5 V across a pot, the voltage at its wiper is proportional to its position. This voltage (position) will be read by the microcontroller through an analog input. The PIC device has one ADC; however, any input can be connected to it via an internal multiplexer. For each analog input you want to use, an ADC component must be dragged into the design (i.e., flowchart panel). The ADC's properties pane is easy to use. Choose the channel and select how you want the conversion presented, along with the variable, which will receive the completed conversion. I am using the stock settings as I don't have any special requirements for conversion or acquisition times.

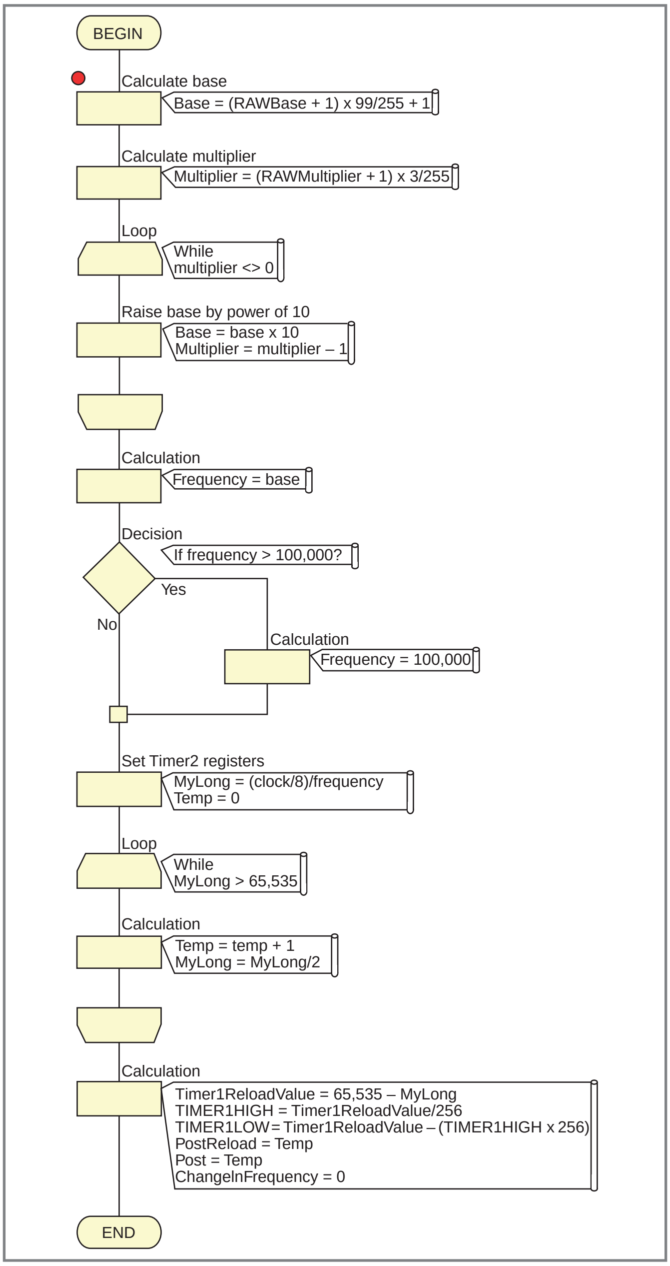


Figure 2—A separate macro (i.e., flowchart) is called whenever a change is made to any input (i.e, pots). All calculations are performed here to transform the ADC values (i.e, pot positions) to values that are meaningful to other process operations.

Also, even though this ADC is capable of 10-bit resolution, I am using only the most-significant 8 bits (byte result).

Each input uses a special calculation to put the result values into the specific range I need. The pot inputs are converted to results in a 0-to-255 range. This value is compared to the last conversion value. If a change has occurred, the new value is saved along with a Boolean variable flag, `ChangeInFrequency`. After both pots have been converted, I use this flag to determine if the Timer 1 reload value needs to be recalculated. If nothing has changed, execution jumps back to resample the pots again, thanks to the `while=1` loop.

This simple loop is only expanded if any pot has changed position. If so, the `Calculate_Frequency` macro is called. The macro has its own flowchart and consists of some calculations to change a pot's conversion result into a useable value (see Figure 2). Integer arithmetic is used to convert pot 1's result (0–255) is to the design value (1–100). Here is the formula:

$$\text{Base\_Frequency} = (\text{result}+1) \times \frac{99}{256} + 1$$

$$\text{for result} = 0\text{, that's } \left(1 \times \frac{99}{256}\right) + 1 \text{ or } 1$$

$$\text{for result} = 255\text{, that's } \left(256 \times \frac{99}{256}\right) + 1 \text{ or } 100$$

In a similar way, integer arithmetic is used to convert pot 2's result (0–255) to the design value (0–3). Here is the formula:

$$\text{Multiplier} = (\text{result}+1) \times \frac{3}{255}$$

$$\text{for result} = 0\text{, that's } \left(1 \times \frac{3}{256}\right) \text{ or } 0$$

$$\text{for result} = 255\text{, that's } \left(256 \times \frac{3}{256}\right) \text{ or } 3$$

When an a variable is added to the design, a "type" must be selected for this variable. This could range from a single-bit to a floating-point type variable. The choice impacts the data space required for the variable (and also the support code that may be needed to handle calculations involving this variable). Don't use floating-point variables if you can avoid them, especially when memory is precious. Receiving the "out of memory" message when building your application is a bummer!

A `while` loop is used to scale up the `Base_Frequency` by a factor of 10 if the multiplier is greater than 0. The multiplier is decremented after each multiplication, which enables the `while` loop to eventually exit when the test (multiplier is greater than 0) is false.

Now that I have frequency, I can determine how many clock ticks (125 ns) this equals. Remember, one cycle is two toggles of the output. This count's complement is used because Timer 1 counts up to overflow on an increment from 0xFFFF to 0x0000. This calculated complement continuously reloads Timer 1 every time it overflows (once each half cycle), so it is saved in byte form as an upper byte (`TIMER1HIGH`) and a lower byte (`TIMER1LOW`). Once recalculated, this macro is done and you can return to the main execution loop.

TIMER 1 INTERRUPT

The `interrupt` routine is this project's last flowchart. As discussed earlier, Flowcode does not give direct access to the Timer 1 counter register. Some C code needs to be added to reload this register every time the interrupt occurs, so the next interrupt will occur right on time. Previously, I calculated the required Timer 1 count and saved the values in variables `TIMER1HIGH` and `TIMER1LOW`. These values are available in the C code, which makes this easy. Figure 3 shows the C code simply stops Timer 1, reloads the 16-bit counter a byte at a time, then restarts the timer. The remaining interrupt code is responsible for toggling the output bit.

By adding a couple of pots to the panel pane and assigning them to the analog inputs for AN0 (PORTA.0) and AN1

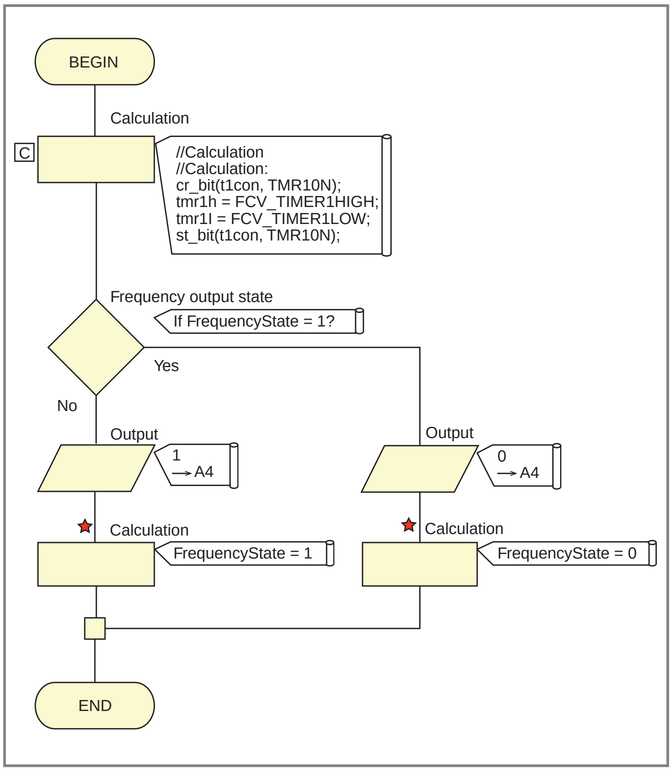


Figure 3—When an interrupt occurs from the Timer 1 counter rolling over, the timer's 16-bit counter register needs to be reloaded with the proper reload value (calculated in the `Calculate_Frequency` macro) so it will interrupt again after the proper delay. The output's state also needs to be determined.

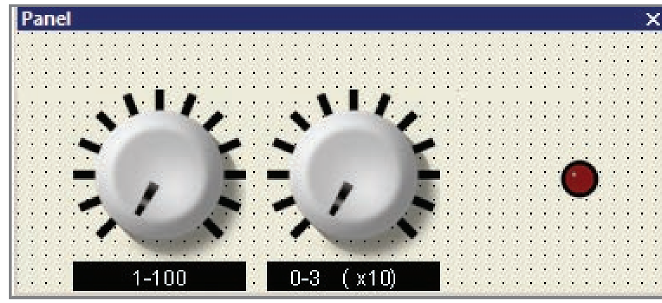


Photo 3—Simulated components can be added to a panel pane to help the design's simulation. Inputs are pots and outputs are LEDs. During simulation, the mouse can be used to rotate the pots to see how the position relates to the conversion values as the main loop continually converts these inputs. Simulation enables you to step through your code so you can verify whether or not the calculation macro's conversion values actually produce the desired values. When Timer 1 produces an interrupt, you can also verify that the timer is reloaded correctly and the output state (LED) is set/cleared as necessary.

(PORTA.1), these become part of the simulation (see Photo 3). And, by stepping through the code, I can twist the pots and ensure the conversion results are properly translated into the correct design values. Simulating individual routines helps ensure that all is in order. As the frequency is cranked down, the Timer 1 counter value continues to grow. Unfortunately, it will grow larger than its 16-bit maximum before it reaches 1 Hz. If Timer 1 used a 24-bit count, I'd be all set. However, it can't count as high as required (even if all the prescaling is set to maximum). I could limit the lowest frequency to 122 Hz and avoid this altogether, but what's the fun in that?

I can trick the toggling output from doing its job by introducing

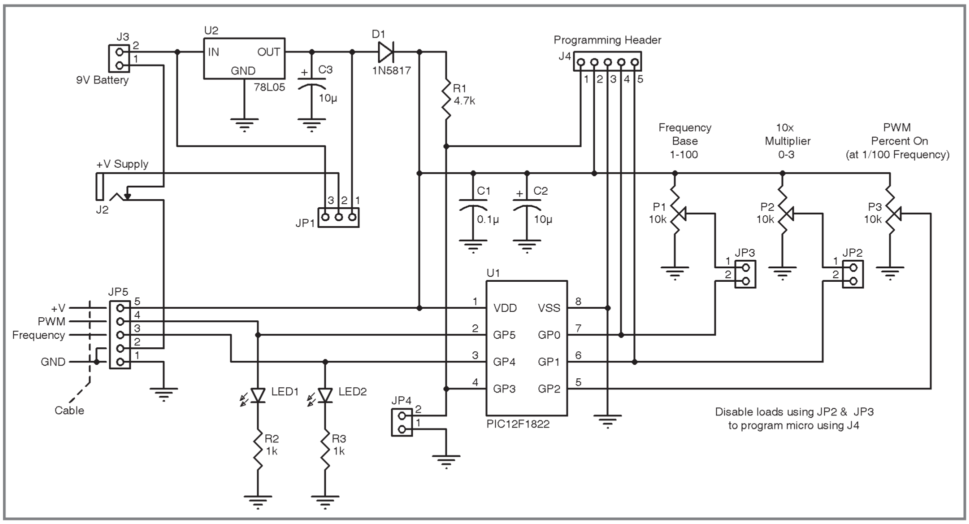


Figure 4—A schematic is a great place to start because it defines the functions of and connections to all the microcontroller's pins. This must be done so you can define them correctly when adding components to your flowchart.

an additional counter in the `interrupt` routine. I can use the `Post` variable. If it isn't 0, I just exit the interrupt without doing any output toggling (just reducing `Post` by 1). If it is 0, I reload `Post` with `PostReload`, which might already be 0. This provides an extra layer of delay before any toggling can happen.

Back in the `Calculate_Frequency` macro, another `while` loop needs to be added that divides the Timer 1 reload value in half and increments `PostReload` if the reload value exceeds 65535. The increase in `PostReload` offsets the reduction in the reload value such that it can never be too large (greater than 65535).

BREAKING UP THE OUTPUT

The frequency output I've created can be thought of as a 50% duty cycle PWM output. Let's create a more configurable PWM output, one that can be controlled from 0% to 100%. In reality, 0% and 100% are kind of useless since both of these will never change state! If I was happy with PWM in 10% increments, I would only need to divide the frequency by 10 to achieve this. I'll be dividing by 100 to get 1% resolution.

To start, I must add another pot to the circuit. The third pot enables the PWM to be set from 0–100 (percent ON:OFF). AN2 is handled the same way as AN0 and AN1. Integer arithmetic is used to convert the byte result from the conversion (0–255) to the design value (0–100). Here is the formula:

$$\text{NewONTime} = (\text{result} + 1) \times \frac{100}{256}$$

$$\text{for result} = 0\text{, that's } \left(1 \times \frac{100}{256}\right) \text{ or } 0$$

$$\text{for result} = 255\text{, that's } \left(256 \times \frac{100}{256}\right) \text{ or } 100$$

Photo 4—The PCB fabricated for this project was designed to fit a Pactec enclosure that comes with a 9-V battery compartment. The pots and LEDs are mounted so they will be available from the PCB's solder side. An external cable includes frequency and PWM outputs as well as ground and 5 V.

The byte `NewONTime` is used to reload two new variables (`ONTime` and `OFFTime`) within the `Timer1_Overflow` interrupt routine (see Figure 4). Previously, this interrupt routine used a Boolean variable `FrequencyState` to determine the frequency output's state. Likewise, another Boolean variable is added, `PWMState`, which determines the PWM output's state. Each time through the routine (assuming `Post` hasn't demanded an early exit), the `PWMState` is tested and the program flow divides into one of two paths, `PWMState=1` (the output is high-ON) or `PWMState=0` (the output is low-OFF).

If `PWMState=1`, I need to determine if it should be turned off. If `ONTime=100`, it should never be turned off, so I just need to exit. However, the ON and OFF times are reloaded for next time, in case the PWM pot was changed. If `ONTime=0` then I am already at the minimum count and don't need to decrement it. Otherwise, I decrement the `ONTime` count and test it for less than 1. If the count has reached 0, either because it was set to 0 (`NewONTime=0`) or it has gone through this path, `ONTime` counts and has finally reached 0 (remained on for the required duty cycle), set `PWMState=0`, the PWM output to "0" and reload the `ONtime` and `OFFTime` variables. Otherwise, I must still decrement the `ONTime`, so I just exit.

The other path (`PWMState=0`) is similar, but this path controls the amount of time the PWM output remains at 0 before I must set `PWMState=1`, set the PWM output to 1, and reload the `ONtime` and `OFFTime` variables. Note the PWM frequency is 50 times (not 100) slower than the frequency. This is because I am dividing each half cycle of the frequency output by 50 to get 1% resolution over the full cycle.

SCHEMATIC & PCB

The frequency and PWM outputs are brought out along with 5 V and ground through connector J1, as shown in Figure 4. While the Pactec plastic case has a 9-V battery compartment that can power the oscillator for hours, a DC power jack can also be enabled. The 5-V wire can power external circuitry if the oscillator is powered by battery or power supply. If the oscillator is left unpowered, the external circuit can supply 5 V to the oscillator.

The fabricated PCB uses board-mount pots accessible from the solder side of the board along with the LEDs connected to the frequency and PWM outputs. The cable connector ties the supply ground to the circuit ground so a power switch is not needed. I placed microhooks on the ends of each cable wire to make it easy to apply signals to external circuitry. Photo 4 shows the simple circuitry that fits nicely into a small plastic enclosure. Now it's ready to provide external stimulation as sometimes required by other projects.

FUTURE POSSIBILITIES

I like this PIC part. It also includes a 5-bit DAC that could be used to simulate other wave shapes. Presently, the digital-to-analog converter (DAC) is not a supported component, but MML makes provisions for a user to add to the component library. Most of the wireless and peripheral components were added this way. Wave-shape tables could be implemented and the DAC could be updated with every Timer 1 interrupt. This would enable sine and triangle output to replace PWM output. I'd add a fourth pot to choose wave shape.

Programming with Flowcode is an interesting experience. In my case, I was a bit disappointed with the amount of finagling I needed to get my design to work. It was unclear what restrictions I would have using various peripherals. I'm sure there are some designs that might not take any tweeking at all.

Flowcode is only one of MML's products. Hardware E-blocks are available for developing Flowcode on PIC, AVR, and ARM microcontrollers. There is a whole line of component E-blocks that enables you to plug onto any of the micro E-blocks. This is handled via multiple DB-9s (e.g., circuit ground and a full 8-bit port) sprinkled around the microcontroller's perimeter. While MML makes it convenient for you to plug and play, you can certainly use Flowcode to produce a .HEX file for your own circuitry without any of the E-blocks hardware.

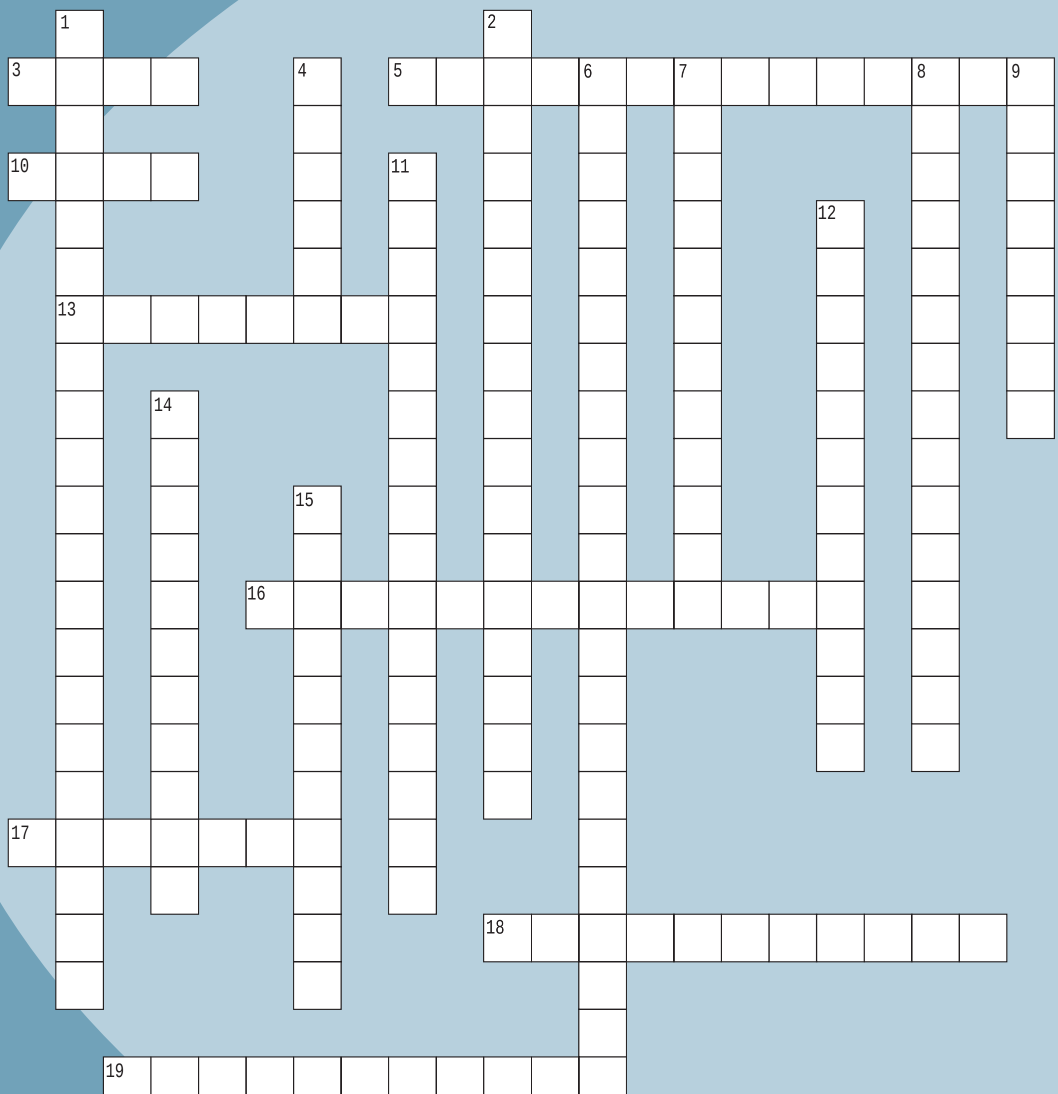
Jeff Bachiochi (pronounced BAH-key-AH-key) has been writing for Circuit Cellar *since 1988. His background includes product design and manufacturing. You can reach him at jeff.bachiochi@imaginethatnow.com or at www.imaginethatnow.com.*

SOURCES

Flowcode IDE and E-blocks multiprogrammer board
Matrix Multimedia, Ltd. | www.matrixmultimedia.com

PIC12F1822 Microcontroller
Microchip Technology, Inc. | www.microchip.com

CROSSWORD



Down

1. Decodes using two pieces of information, one public and one private [three words]
2. A loudspeaker that uses a consolidating technique to achieve high efficiencies [two words]
4. The flexible collar that helps keep a voice coil magnetically centered
6. A pocket-sized radar that runs off AA batteries and is often used as a basic motion sensor for security applications [three words]
7. Check performed by an independent team on a system installed at a place other than the targeted customer's site [two words]
8. An action that is non-interruptible by any other one and never presents partial results to an outside observer [two words]
9. As *Circuit Cellar* prepares to celebrate its 25th anniversary, a key past, present, and future theme of the magazine centers on this type of technology
11. Involves measuring physical quantities with time and spatial variances [two words]
12. Jeff Bachiochi describes this method of code writing in this issue
14. Mark Csele's article, "DSP-Based Color Organ" (*Circuit Cellar*, 249 2012), used this technique to create high-performance filters
15. A device that combines input signals, shares a single transmission channel, and enables data compression

Across

3. German engineer and inventor (1910–1995) who is credited with creating the Z3, a program-controlled Turing-complete computer
5. Alternating voltage/current with the exact same energy content as the same value of direct current; a.k.a., quadric mean [three words]
10. Stores binary data; synonym: drop
13. A specialized linear-beam vacuum tube
16. English physicist and inventor William Sturgeon (1783-1850) who is credited with using electric current to develop the first one of these objects in 1825
17. *Circuit Cellar* columnist who frequently writes about frequency
18. An energy-saving device that was the topic of Fergus Dixon's article about building an MCU-based, 'Net-enabled controller (*Circuit Cellar*, 263 2012) [two words]
19. Measures low current

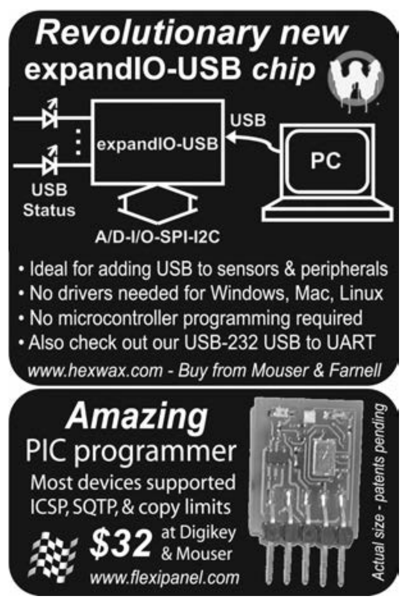
The answers are posted at www.circuitcellar.com/crossword and will be available in the next issue.

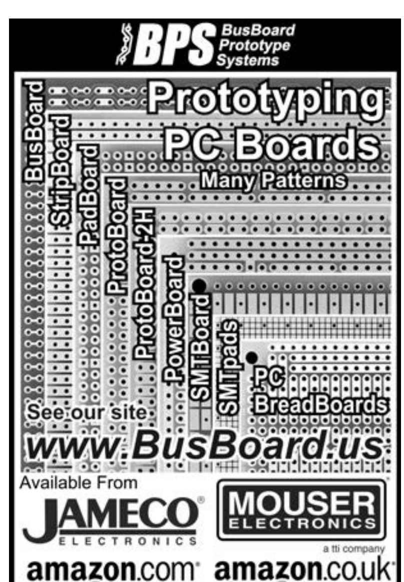
IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

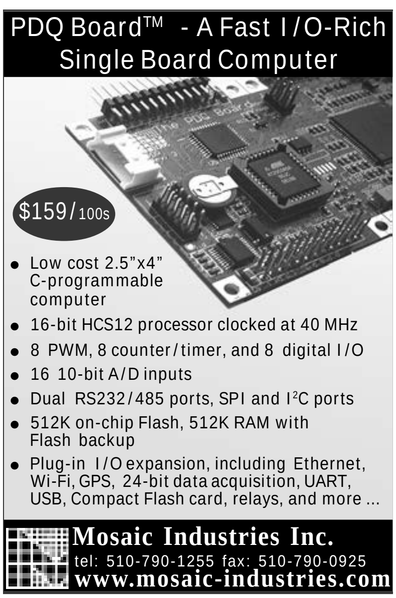
AD FORMAT: Advertisers must furnish digital files that meet our specifications (www.circuitcellar.com/advertise). ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2" x 3" FORMAT. E-mail adcopy@circuitcellar.com with your file or send it to IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066. For current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





Burn-In & Test Sockets
0.4mm to 1.27mm
Industry's
Smallest Footprint
RoHS
Ironwood
ELECTRONICS
1-800-404-0204
www.ironwoodelectronics.com



PDQ Board™ - A Fast I/O-Rich Single Board Computer
$159/100s
Low cost 2.5"x4" C-programmable computer
16-bit HCS12 processor clocked at 40 MHz
8 PWM, 8 counter/timer, and 8 digital I/O
16 10-bit A/D inputs
Dual RS232/485 ports, SPI and I²C ports
512K on-chip Flash, 512K RAM with Flash backup
Plug-in I/O expansion, including Ethernet, Wi-Fi, GPS, 24-bit data acquisition, UART, USB, Compact Flash card, relays, and more ...
Mosaic Industries Inc.
tel: 510-790-1255 fax: 510-790-0925
www.mosaic-industries.com

ALL ELECTRONICS
CORPORATION
Electronic and Electro-mechanical Devices, Parts and Supplies. Wall Transformers, Alarms, Fuses, Relays, Opto Electronics, Knobs, Video Accessories, Sirens, Solder Accessories, Motors, Heat Sinks, Terminal Strips, L.E.D.S., Displays, Fans, Solar Cells, Buzzers, Batteries, Magnets, Cameras, Panel Meters, Switches, Speakers, Peltier Devices, and much more....
www.allelectronics.com
Free 96 page catalog
1-800-826-5432

Circuit Cellar's
25th Anniversary
Sale 25% off
all Circuit Cellar and Elektor books, back issues, archive CDs, and kits!
CIRCUIT CELLAR
CC-WEBSHOP
Shop now at
www.cc-webshop.com
COUPON: 25SEPT12
Offer expires September 30, 2012
CIRCUIT CELLAR
EMBEDDED PROGRAMMING
Build a Capacitive-Touch Personal Amp
An MCU for Low-Power Applications
Mesh Networking with the SNAP OS
CIRCUIT CELLAR
EMBEDDED DEVELOPMENT
Q&A: Programming, Testing, & Success
LOCATION: United States
PAGE: 40
INSIGHT: Environmental Stress Screening
LOCATION: Canada
PAGE: 52
HOW TO: Choose Linux for an Embedded App
LOCATION: United States
PAGE: 56
THE WORLD'S SOURCE
ADuC841
Microcontroller Design Manual
Microprocessor Design using Verilog HDL
Assembly Language Essentials

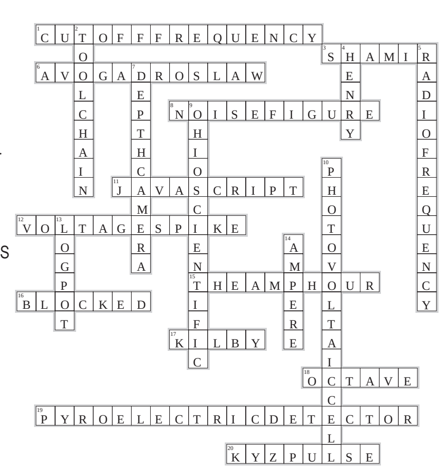
CROSSWORD ANSWERS from Issue 265

Across

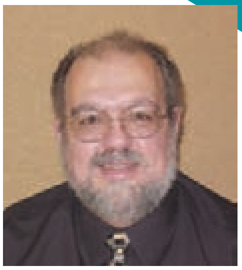
1. **CUTOFFFREQUENCY—**The frequency at which a filter's output has fallen by 3 dB from the maximum level [two words]
3. **SHAMIR—**Israeli cryptographer and one of inventors of the RSA algorithm
6. **AVOGADROSLAW—**V/n = k [two words]
8. **NOISEFIGURE—**Columnist Robert Lacoste's article, "Noise Filters 101" (*Circuit Cellar* 249, 2012), discussed how to determine this and measure it in a radio frequency filter [two words]
11. **JAVASCRIPT—**ECMAScript-approved language that supports a lot of C-structured programming syntax
12. **VOLTAGESPIKE—**Lightening or a tripped circuit wire [two words]
15. **THEAMPHOUR—**Name of the radio show co-hosted by two recent *Circuit Cellar* Q&A interviewees [three words]
16. **BLOCKED—**When a software thread relinquishes control of the processor to the operating system
17. **KILBY—**American physicist (1923–2005) who worked with Robert Noyce to create the first integrated circuit
18. **OCTAVE—**The interval between one musical pitch and another with half or double its frequency
19. **PYROELECTRICDETECTOR—**A capacitive sensor that changes its polarization in response to a change in temperature [two words]
20. **KYZPULSE—**In a mechanical electrical meter, a pulse that changes state every half rotation of the meter's disk and represents a quanta of energy [two words]

Down

2. **TOOLCHAIN—**Software used to create other software, usually including a text editor, a compiler, a linker, and a debugger
4. **HENRY—**A unit of inductance; abbreviation "H"
5. **RADIOFREQUENCY—**An amount of oscillation ranging from 3 kHz to 300 GHz [two words]
7. **DEPTHCAMERA—**Miguel Sánchez's article, "Image Processing Development" (*Circuit Cellar* 263, 2012), used one of these along with an MCU and bipolar stepper motors to accomplish some computer vision-related tasks [two words]
9. **OHIOSCIENTIFIC—**In 1978, this company released one of its first products—a simple, MOS Technology 6502 microprocessor-based, single-board computer [two words]
10. **PHOTOVOLTAICCELL—**According to Jeff Bachiochi's article in this issue, a single one of these is "a translucent sandwich of P-type and N-type material forming a huge diode junction that can be exposed to a light source" [two words]
13. **LOGPOT—**Abbreviation; used as a volume control in audio amplifiers [two words]
14. **AMPERE—**An SI unit of electric current



PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

Managing Expectations

I have a theory. People are a lot more comfortable when they can predict the future, or at least if they think they can. Look at all the resources we put into forecasting the weather or economic conditions, despite the fact that we know these are complex, chaotic systems whose sensitivity to initial conditions makes any long-term predictions less dependable. This applies on a personal level, too. We have developed protocols that help us interact with each other. We say "hello" when we pick up the phone. We shake hands when we meet for the first time. These protocols (i.e., "social customs") help us control the process of learning about each other—what we need and what we can provide in a relationship.

Communication "protocol" is particularly important in the relationship between an engineer and his client. There is a huge amount of diversity in such a relationship. Unstated assumptions can lead to enormous gaps in expectations resulting in disappointment, frustration, anger, or even legal action in extreme cases.

Despite the fact that human resource types tend to treat engineers as interchangeable cogs in a machine, individual engineers may have distinctly different talents. Some have extensive expertise in a particular technology. Others have more general system-level design skills along with an ability to pick up the finer points of new technologies "on the fly." Some are good at communicating with clients and developing system concepts from vague requirements, while others need to dig into the minutiae of functional specifications before defining low-level implementation details.

As an engineer, it is important to recognize where your talents lie in this broad spectrum of possibilities, and to be honest about them when describing yourself to coworkers and potential clients. Be especially careful with people who are going to represent you to others, such as headhunters and engineering services brokers. Resist the urge to "inflate" your capabilities. They'll be doing that on your behalf, and you don't want to compound the problem.

Similarly, engineering services customers come in all shapes and sizes. Some only have a vague product idea they want to develop, while others may have a specific description of what needs to be solved. Some small companies will want you to manage the entire product development process, while larger ones have management systems (i.e., bureaucracies) and will expect you to work within established procedures. Some will want you to work onsite using their equipment, while others will expect you to have your own workspace, support infrastructure, elaborate test equipment, and so forth.

In any case, from the customer's point of view, there are risks to using outside engineering services. How much are they going to have to spend? What are the chances of success at that level of expenditure? Unless there are unusually large, nonrecurring engineering (NRE) charges associated with the project, labor will be the customer's biggest expense. The obvious question is: How much time is it going to take? These are questions that are sometimes difficult to answer at a project's inception, especially if the requirements are poorly defined. It may become necessary to guide the customer through a process of discovery that delineates individual project steps in terms of cost and accomplishment for each step. These early iterations could include things like a feasibility study or a detailed functional specification.

Generally, the customer is going to ask for a fixed-price arrangement, but beware. As the engineer, this means you are assuming all the risk. If the schedule slips or problems crop up, you are the one who will take the loss. Fixed-price contracts are a tough equilibrium. Invariably, they involve padding time estimates to balance the risk-benefit ratio, but not so much that you price yourself out of the job in the first place. A better consulting situation is a time and materials contract that puts more of the risk back on the customer and provides flexibility for unforeseen glitches. Knowledgeable customers should understand and be okay with this.

The point is, you need to be willing to take the lead and let the customer know what is happening now and every step of the way. That way, they don't get surprised, particularly in a negative way. Since we can't assume every consulting customer is reading my editorial, it's up to you to explain these issues. Do it right, and you'll have a positive foundation on which to build your relationship.

And, even though I have been directing my remarks primarily to independent consultants and contractors, as an engineer, you are providing your services to others. Even as a full-time employee in a company where your only "customers" are other departments (i.e., manufacturing or testing), these principles still apply. While your present salary is a given, its future progress and longevity is all about the art of *managing expectations* in the eyes of others.

steve.ciarcia@circuitcellar.com

www.ftdichip.com
FTDI Chip
ENHANCED USB PERFORMANCE
Streamlined USB Bridge Solutions
X-CHIP
EXtensive Interfaces
UART, FIFO, SPI, I2C, FT1248
EXtended Features
Battery charger detection
Low active power (8 mA, typical)
Internal MTP memory
Expandable clocking; clock generation and system clock out
EXceptional Drivers
Windows, MacOS, Android, and Linux
FTDI
1142-B
HNY37.00
FT200XD