PROJECT: Multilevel Audible Probe
LOCATION: Bulgaria
PAGE: 14

EE TIPS: Electronic Product Simulation & Testing
LOCATION: United States
PAGE: 60

INSIGHT: Efficient Low-Power Electronics
LOCATION: United States
PAGE: 64

CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

MAY 2012
ISSUE 262

MEASUREMENT & SENSORS

Implement Cap-Touch Sensors

Sensors & Image Stabilization

DIY Multipurpose Interface for Embedded Systems

Embedded Security: Threat Prevention & System Protection

"SNAP" Projects

$7.50 U.S. ($8.50 Canada)

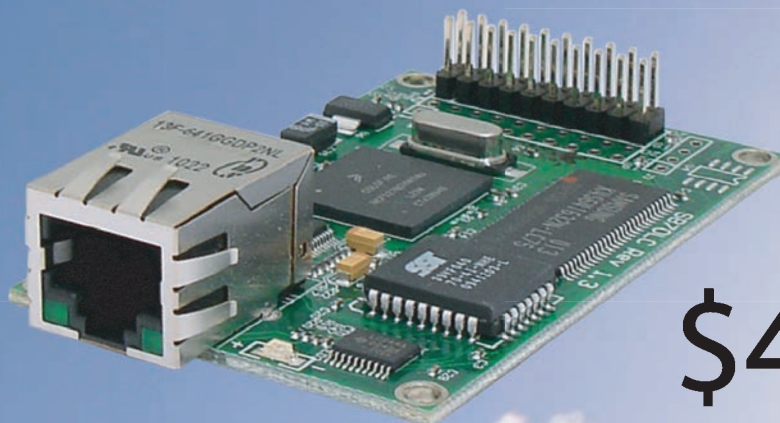
www.circuitcellar.com

PLUS

Life in Transmission

The Exciting Endeavors of a Communications Pro
// Operating Sonar Systems
// Working in Fiber Optics
// Developing IR Technology
// And More

The NetBurner SB70LC



$49.00
Qty. 100
SB70LC Board

The complete hardware and software solution

- Low-cost
- 10/100 Ethernet
- Two Serial Ports
- 32-bit Performance
- Digital I/O
- SSH/SSL Option
- Flash Card Ready
- I^{2}C • SPI Interface
- Example applications

$99.00
SB70LC Development Kit

The SB70LC Development Kit

The NetBurner SB70 LC Development Kit is available to customize any aspect of operation including web pages, data filtering, or custom network applications. The kit includes platform hardware, ANSI C/C++ compiler, TCP/IP stack, web server, e-mail protocols, RTOS, flash file system, Eclipse IDE, debugger, and cables. The kit enables you to communicate with peripherals that use SD/MMC Flash Card (including SDHC), SPI, I^{2}C, or the general purpose digital I/O interface. The NetBurner security suite option includes SSH v1, v2 and SSL support.

NetBurner
Networking in One Day!

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

freescale™
Alliance Member

TASK MANAGER

Advances in Measurement and Sensor Tech

As I walked the convention center floor at the recent Design West conference in San Jose, CA, it quickly became clear that measurement and sensor technologies are at the forefront of embedded innovation. For instance, at the Terasic Technologies booth, I spoke with Allen Houng, Terasic's Strategic Marketing Manager, about the VisualSonic Studio project developed by students from National Taiwan University. The innovative design—which included an Altera DE2-115 FPGA development kit and a Terasic 5-megapixel CMOS sensor (D5M)—used interactive tokens to control computer-generated music. Sensor technology figured prominently in the design. It was just one of many exciting projects on display.

In this issue, we feature articles on a variety of measurement-and sensor-related embedded design projects. I encourage you to try similar projects and share your results with our editors.

Starting on page 14, Petre Tzvetanov Petrov describes a multilevel audible logical probe design. Petrov states that when working with digital systems "it is good to have a logical probe with at least four levels in order to more rapidly find the node in the circuit where things are going wrong." His low-cost audible logical probe indicates four input levels, and there's an audible tone for each input level.

Matt Oppenheim explains how to use touch sensors to trigger audio tags on electronic devices (p. 20). His design is intended to help visually impaired users. But you can use a few capacitive-touch sensors with an Android device to create the application of your choice.

Read the interview with Lawrence Foltzer on page 30 for a little inspiration. Interestingly, one of his first MCU-based projects was a sonar sensor.

The impetus for Kyle Gilpin's "menU" design was a microprocessor-based sensor system he installed in his car to display and control a variety of different sensors (p. 34). The current system enables him to navigate through a hierarchical set of menu items while both observing and modifying the parameters of an embedded design.

The final measurement-and-sensor-related article in this issue is columnist Richard Wotiz's "Camera Image Stabilization" (p. 46). Wotiz details various IS techniques.

Our other columnists cover accelerated testing (George Novacek, p. 60), energy harvesting (George Martin, p. 64), and SNAP engine versatility (Jeff Bachiochi, p. 68).

Lastly, I'm excited to announce that we have a new columnist, Patrick Schaumont, whose article "One-Time Passwords from Your Watch" starts on page 52. Schaumont is an Associate Professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. His interests include embedded security, covering hardware, firmware, and software. Welcome, Patrick!

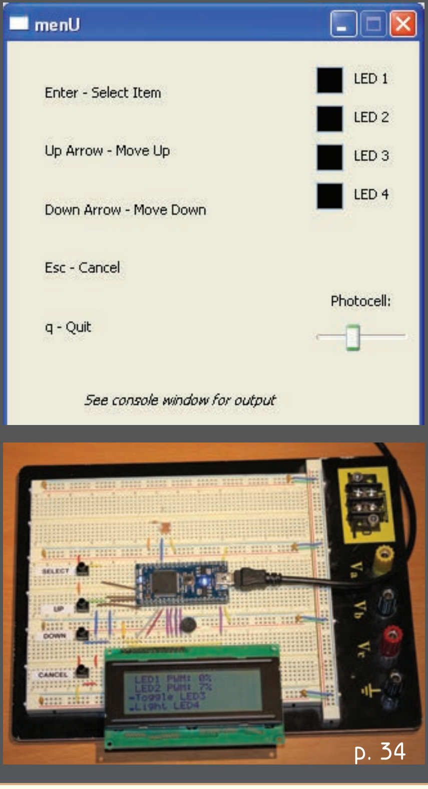
cj@circuitcellar.com

C. Abate

INSIDE ISSUE 262

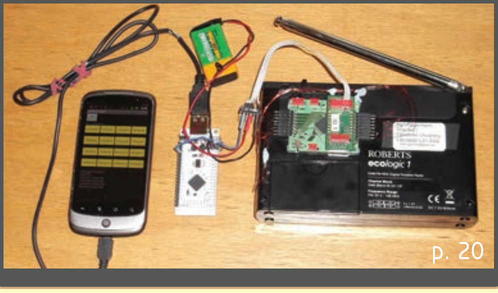
May 2012 • Measurement & Sensors

Menu System Development & Testing



p. 34

Portable Audio Touch Sensor

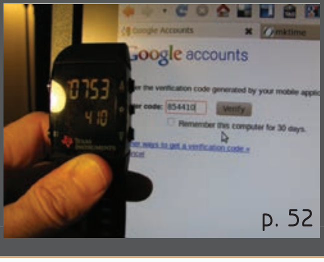


p. 20

Image Stabilization Technology

p. 46

One-Time Passwords



p. 52

THE TEAM

FOUNDER/EDITORIAL DIRECTOR:	Steve Ciarcia	CONTROLLER:	Jeff Yanco
EDITOR-IN-CHIEF:	C. J. Abate	PUBLISHER:	Hugo Van haecke
ASSOCIATE EDITOR:	Nan Price	ASSOCIATE PUBLISHER:	Shannon Barraclough
CONTRIBUTING EDITORS:	Jeff Bachiochi, Bob Japenga, Robert Lacoste, George Martin, Ed Nisley, George Novacek, Patrick Schaumont, Richard Wotiz	ART DIRECTOR:	KC Prescott
		CUSTOMER SERVICE:	Debbie Lavoie
		ADMINISTRATIVE COORDINATOR:	Valerie Luster
		MARKETING REPRESENTATIVE:	Amanda Anderson
		GRAPHIC DESIGNER:	Nordian Davis
PROJECT EDITORS:	Ken Davidson, David Tweed	ADVERTISING COORDINATOR:	Kim Hopkins

THE NETWORK

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

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 262 May 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 276438 members in 83 countries.

Not a member yet?

Sign up at www.circuitcellar.com

Supporting companies

Not a supporting company yet?

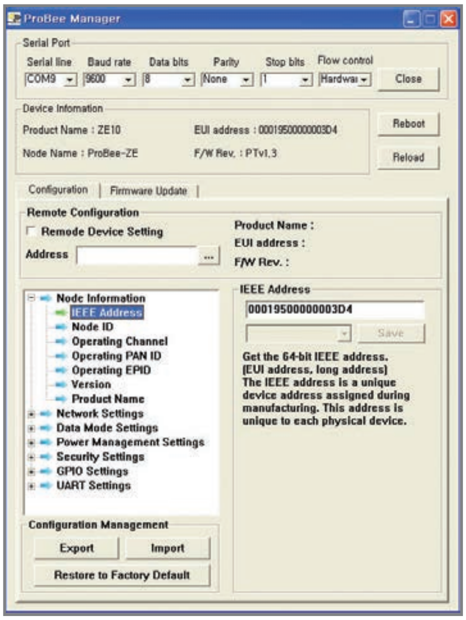
Contact Peter Wostrel (peter@smmarketing.us, Phone 978.281.7708, Fax 978.281.7706) to reserve your own space for the next issue of our member's magazine.

Head Office

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

ALL-IN-ONE ZigBee SERIAL ADAPTER

The **ProBee-ZS10** is an all-in-one ZigBee serial adapter with an integrated ZigBee core and radio/antenna circuits combined with a high-level software library. It is designed to meet specific requirements of low-cost and low-power wireless applications such as home automation, smart energy, healthcare, building automation, and telecom services by adopting the ZigBee technology. The adapter is well suited for several purposes: advanced metering infrastructure applications; home area networks (HAN); neighborhood area networks (NAN); advanced lighting, entertainment, and climate-control systems; service-based monitoring, security, and awareness systems; commercial building automation; climate and lighting-control systems; and industrial and domestic applications.



The ZS10 is capable of transferring data at up to 250 kbps and can reach up to 1 mile of outdoor line-of-sight. It fully supports ZigBee 2007/ZigBee Pro stack. In addition, the serial adapter has an integrated 2.4-GHz, IEEE 802.15.4-compliant coprocessor and supports RS-232 serial communication with a data rate up to 115 kbps. The ProBee-ZS10 wirelessly supports remote management features such as firmware upgrade and system configuration using a modem AT command set. It also features enhanced portability, including a standard and extended battery pack. An easy-to-use Windows configuration tool is also available with the adapter.

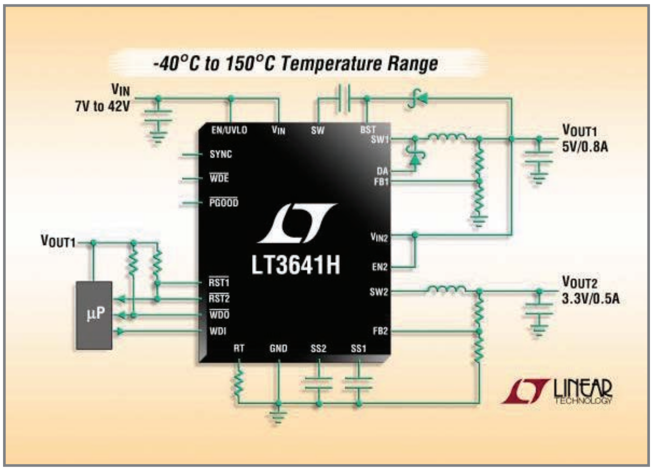
Contact Lemos for pricing.

Lemos International, Inc.
www.lemosint.com

DUAL STEP-DOWN REGULATOR WITH POWER-ON RESET & WDT

The **LT3641** is an H-grade version, dual-channel, current-mode, step-down switching regulator with power-on reset and a watchdog timer (WDT). It features a high-voltage buck regulator with 4-to-42-V input voltage range with 55-V transient capability and 1.3-A output current, making it well suited for load dump and cold crank conditions commonly found in automotive applications that require constant output regulation with input transients as high as 55 V. The LT3641 features a low-voltage synchronous buck regulator with 2.5-to-5.5-V input voltage range and up to 1.1-A continuous output current.

Separate inputs can be used for each channel; however, most applications will use the output of the high-voltage channel to power the lower voltage channel, offering dual outputs while optimizing efficiency and switching frequency.



The LT3641 H-grade version operates at a junction temperature of up to 150°C, compared to the E- and I-grade versions which operate at a maximum junction temperature of 125°C. All electrical specifications are identical for the E-, I-, and H-grade versions.

Additional features include: a synchronizable, adjustable, 350-kHz to 2.5-MHz switching frequency; a programmable power-on reset timer; a programmable Window-mode WDT; a typical quiescent current of IQ less than 290 µA; a low shutdown current of IQ less than 1 µA; a programmable soft start; and thermal shutdown. The LT3641 is available in thermally enhanced 28-lead (4 mm × 5 mm) QFN and 28-lead TSSOP packages.

Pricing starts at **$4.43** each for 1,000-piece quantities.

Linear Technology Corp.
www.linear.com

by Rob Dautel (USA)

The RL78 Green Energy Challenge Has Begun

Are You Ready to Shape the Future We Live In?

Green Energy ideas are rapidly changing our future, so what's your idea? Here's your opportunity to step into the light, present your Green Energy design to the world, and help create a future that's bright, clean, and healthy.

Green Energy. It's a phrase that's become so commonplace in recent that we now hear it every day of our lives. But what is it? Some would say it's improving and using our current energy systems and supplies better and more efficiently. Others might say it's concentrating on building out technologies for alternate energy such as wind, solar, and geothermal. Perhaps it's completely new ways of thinking, such as energy harvesting and extremely low-power means like RF gathering, heat scavenging, and piezo mechanical generation. Others may say it's creating intelligent control systems, monitoring, data gathering, and processing to better utilize energy and resources, whatever the source technology may be.

The encompassing term "green energy" is not limited to electrical energy, but has become a blanket term adopted by a breadth of industries covering a host of topics from natural resources like water, air, minerals, and petroleum, to building structures and best practices, materials reuse, and working smarter and more efficiently both for ourselves and our surroundings.

In only a few short years, the green energy idea has taken the world by storm and whatever the definition, one thing is clear: green energy is about making the world a better place to live, work, and play in. It's about the future and leaving future generations with a world worth having.

So what is green energy to you? That's what the RL78 Green Energy Challenge is all about. We want to see your ideas, your designs, and your future of the green energy revolution. Perhaps it's a remote device monitoring pollution? Maybe it's a box collecting data on home power usage or an energy harvesting biometric design? It could be a low-power controller scavenging heat from an oven or furnace, a meter reading biomass parameters, or a braking system for a wind turbine. Send us your best RL78-based ideas to help make the world a better place, and you'll have a chance to win something green in return: cash.

Renesas Electronics and our contest partners want to see what your version of the green energy future looks like, so we're giving away more than $20,000 in prizes in the RL78 Green Energy Challenge.

We're looking for designs that define the essence of green energy, and inspire others to stand up and do the same. We're not just hosting a design contest; we're beginning a grassroots movement that begins with you. As embedded systems become more connected to each other and to cloud resources, we're enabling new ways to gather, process, and react to data and the environment around us. This new level of processing provides a never-before-seen ability to interact with and manage resources. Are you ready to help shape the future?

The RL78 Green Energy Challenge will focus on designs using the Renesas RL78 family of microcontrollers which combines advanced low-power technology, outstanding performance, and the broadest line-up in its class for the most demanding 8- and 16-bit embedded applications.

These MCUs incorporate key features of the well-established R8C and 78K0R families from Renesas Electronics, giving designers an excellent upgrade path for next-generation designs. The platform concept of the RL78 family provides great flexibility while 41 DMIPS at 32 MHz and 66 µA/MHz provide for high efficiency and ultra-low-power operation.

To make your RL78 Green Energy Challenge design a reality, we're giving away nearly 1,000 RL78G13 Renesas Demonstration Kits (RDKs). The kit includes an onboard debugger, three-axis accelerometer, temperature sensor, liquid crystal display (LCD), isolated triac, a light sensor, FET circuit, IR transmitter and receiver, serial EEPROM, microphone input, audio output, SD card slot, Pmod connector, and much more.

To top off the hardware, we've partnered with IAR Systems to provide their Embedded Workbench Kickstart Edition for RL78 and, for the duration of the challenge, they'll provide full licenses of their Embedded Workbench to contestants. We've also partnered with great companies such as Micriµm, CMX Systems, SEGGER Microcontroller, Total Phase, Exosite, and Okaya Electric to provide a host of software and code examples to jumpstart your design.

So, if you've already signed up for the RL78 Green Energy Challenge, we welcome you and thank you for being part of something we truly believe can help shape the future. If you haven't signed up yet, what are you waiting for? Do it today and show us what green energy means to you!

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

To register and enter the RL78 Green Energy Challenge, go to: www.circuitcellar.com/renesasrl78challenge

PSoC-PRECISION ANALOG DEVELOPMENT KIT

The **CY8CKIT-025** development kit is designed for the Cypress PSoC 3 and PSoC 5 programmable system-on-chip (PSoC) architectures. The kit highlights PSoC's unique 8-to-20-bit Delta-Sigma analog-to-digital converter (ADC) with an internal voltage reference of –0.1%, a sampling rate of up to 384 ksps, and up to 18-bit effective number of bits (ENOB) performance.

The development kit is capable of microvolt measurements at 0.003% precision. In addition, designers can use the ADC to implement both differential and single-ended inputs for the Delta Sigma ADC, which enables them to interface with sensors of varying output types.

The CY8CKIT-025 can be easily configured with a wide range of resolutions, sample rates, and input buffer and gain modes using the graphical user interface (GUI) inside Cypress's PSoC Creator integrated development environment (IDE). In order to simplify device configuration, the PSoC Creator features prewritten software "components" which enable users to graphically select the desired values.

The development kit includes four types of analog temperature sensors: a thermocouple, a diode, a thermistor, and a resistance temperature detector (RTD). It also includes a bonus CY8CKIT-012 PSoC prototyping and development expansion board.

The CY8CKIT-025 development kit costs **$115**.

Cypress Semiconductor Corp.
www.cypress.com

NPN

CIRCUIT CELLAR

Test Your EQ

Problem 1—The classic two-transistor astable multivibrator is shown in Figure 1. Typically, R2 and R3 have at least 10 times the value of R1 and R4. This circuit oscillates, with Q1 and Q2 turning on alternately. From the point in time in a cycle where Q1 first switches on, describe what happens until Q2 switches on.

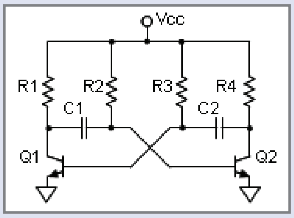


Figure 1

Problem 2—What determines the time of one half-cycle of the oscillation? Does this depend on V_{CC}?

Problem 3—Recently, a different circuit appeared on the web, shown in Figure 2. Again, R2 and R3 are significantly larger than R1 and R4. The initial reaction of one observer was that this circuit can't work, because there's no DC bias path for either transistor. Is this assessment correct?

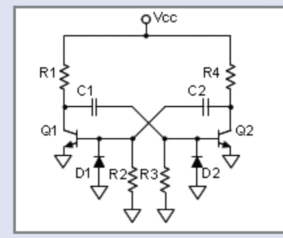


Figure 2

Problem 4—What role do R2 and R3 play in this circuit?

Problem 5—Does the timing of this circuit depend on V_{CC}? If not, what does it depend on?

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

Petre Tzvetanov Petrov (Bulgaria)

Multilevel Audible Logical Probe

Testing, Troubleshooting, and Debugging

You can build a single-input, multilevel audible logical probe in fairly short order. The design detailed in this article is intended for checking equipment with transistor-transistor logic and CMOS integrated circuits or other technologies for installations powered from a 12-V car battery.

Most low-cost logical (digital) probes visually indicate three conditions: low-input level, high-input level, and pulses at the input. These conditions are indicated with LEDs and require the eyes to move and to switch attention from the tested circuit to the logical probe. This can be tiring and may introduce fatigue or irritation. Some probes incorporate 3.5-digit voltmeters, but these are not very useful when you are only interested in several predefined levels.

It is beneficial for the audible signal to relate to the voltage level of the logical signal. The attention should be focused only on the tested circuit without switching between the tested circuit and the logical probe or between the circuit and the oscilloscope's screen.

Consequently, it is preferable when troubleshooting digital systems or in 12-V electrical installations to watch the circuit and listen for information about the voltage level you touched. Talking voltmeters are not sufficient. You don't need to listen to the all digits of the voltage level.

Also, with digital systems, it is good to have a logical probe with at least four levels. It ena180

bles you to find the node in the circuit where things are going wrong. For example, in a 5-V circuit, it is useful to have two level detectors at approximately 0.4 and 0.8 V in order to better evaluate the loading of the tested node.

In this article I'll describe a simple and low-cost audible logical probe that indicates four input levels (see Figure 1). There is an audible tone associated with each input level. When input pulses are present, the audible signal changes to indicate an unstable input level or pulses.

THE CIRCUIT

The logical probe is designed for V_{EE} (i.e., 5-V) systems. But you can apply it to any practical power supply for digital systems or for troubleshooting 12-V installations powered from a car battery. The power supply is limited mainly by the CMOS integrated circuit with Schmitt triggers from National Semiconductors, such as MM74C14 and CD40106 (now discontinued). At 5-V operation, usage of 74LS14 is possible, but the value of the feedback resistor R9 and the capacitors C1–C5 should be appropriately changed.

The logical probe shown in Figure 1 is built around two low-cost and popular integrated circuits: a National Semiconductor LM339 quad comparator (note Freescale Semiconductor's/Motorola's MC3302 and National Semiconductor's

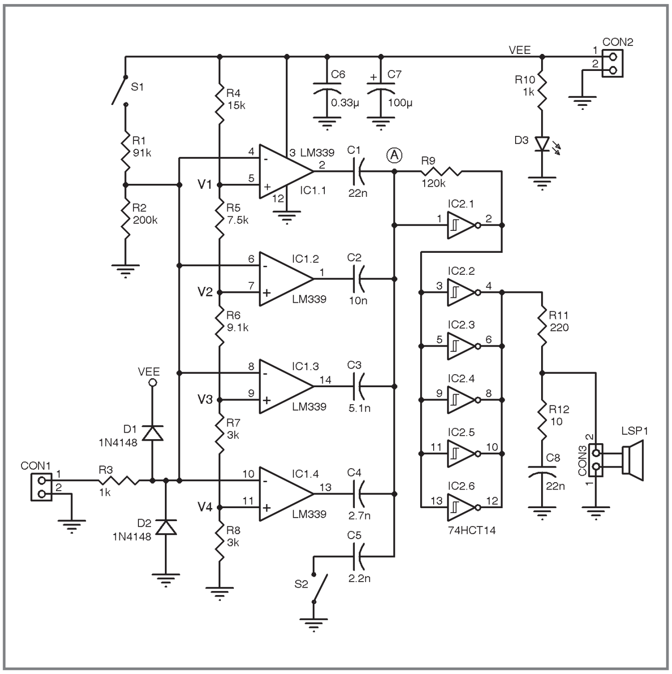


Figure 1—Schematic diagram of the simple audible logical probe

Condition for Input Signal	Maximal Threshold Level	Activated Comparators (Output at Zero)	Connected Capacitor	Approximate Frequency
V_{IN} < V4	V4 = 0.4 V	None	C5	5 kHz
V4 < V_{IN} < V3	V3 = 0.8 V	IC 1.4	C4 + C5	2 kHz
V3 < V_{IN} < V2	V2 = 2.0 V	IC 1.4, IC 1.3	C3 + C4 + C5	1 kHz
V2 < V_{IN} < V1	V1 = 3.0 V	IC 1.4, IC 1.3, IC 1.2	C2 + C3 + C4 + C5	500 Hz
V1 <V_{IN}	V1 = 3.0 V	IC 1.4, IC 1.3, IC 1.2, IC 1.1	C1 + C2 + C3 + C4 + C5	200 Hz
Pulses	It depends on the amplitude of the pulses	Changing combination of activated comparators	C5 + changing combination of connected capacitors	Changing frequency in the range of 200 Hz to 5 kHz
Notes: V4 is the lowest reference level referred to IC 1.4. V1 is the highest reference level referred to IC1.1. The rule for the table is to lower the output audible tone when the input voltage is increasing. V5 > V1 > V2 > V3 > V4, where V5 is the voltage level between R1 and R2 without load (the input of the probe is disconnected).				

Table 1—Threshold levels, conditions, and produced frequencies. The S1 switch is in position 1 (open) and the S2 switch is in position 2 (closed).

LM2901 could also be used) and the National Semiconductor hex inverting Schmitt triggers 74HCT14 or 74HC14 or CD40106.

The quadruple comparator LM339 (or two LM393 comparators) can work as a level detector. The threshold levels of the comparators in the LM339 are determined by the resistors R4, R5, R6, R7, and R8. The levels could have any appropriate value from ground up to the level limited by the comparator LM339, which does not have rail-to-rail input. Using a comparator with rail-to-rail input is possible and could be better, but it is not required.

For example, if the input voltage V_{IN} is higher than 0.4 V and lower than 0.8 V, only the output pin 13 of the comparator integrated circuit (IC) 1.4 will be at a low level and the capacitor C4 will be connected in parallel with C5 (if the switch S2 is closed). Both capacitors will determine the frequency of the square-wave generator built with IC 2.1.

The threshold levels for the comparators, the conditions, and generated output frequencies are listed in Table 1. Table 2 shows tone levels and switch (S1 and S2) positions when the input isn't connected.

For example, if S1 and S2 are open and the input is open, there is no tone produced. At V_{EE} = 5 V, and if the S1 is closed and the input is left open, the input level is set by a resistor divider (R1, R2) to around 3.2 V, or higher than any predefined appropriate voltage level (V1, V2, V2, and V4).

If S1 and S2 are closed, no input signal is applied, and all capacitors C1, C2, C3, C4, and C5 are connected in parallel, the generated frequency will be the lowest possible—around 200 Hz, in that case. If the input is shorted to ground, capacitors C1–C4 are disconnected and the output frequency is determinate by C5. In that condition, the highest frequency is generated and it will be around 5 kHz.

You can change and adapt the frequencies in Table 1 to a particular application and for easier recognition of the input levels.

In practical terms, resistors R1 and R2 do not significantly load the test point due to their relatively high impedance. These resistors could have any appropriate value. But, preferably they should not be omitted because when the input is left open the generated frequency could unpredictably change. Also, the values of R1 and R2 depend on the input currents of the comparators.

It should be noted that if the inverting and noninverting inputs of each comparator are swapped, the logical probe will still work but new tables similar to Table 1 and Table 2 should be produced. For example, all capacitors will be connected in parallel when the input signal is below the lowest threshold level (V4 = 0.4 V in that case).

THRESHOLD-LEVEL MODIFICATION

You don't need to recalculate or redesign the resistor network R4–R8 if the probe is only used with 5-V systems. When several applications with different power supplies and threshold levels are targeted for each application, a proper probe could be designed and used.

If the supply level and the input level are within the range of 4.5 to 16 V, and if it is necessary to change the threshold level, there are several approaches you can use. One of them is to replace R5–R8 with a trimmer potentiometer (e.g., 10 kΩ) and to adjust the voltage levels V1–V4 before each application. The other is to use a switch to select between several predefined threshold levels (see Figure 2). In this figure, the quadruple switch S1ABCD is used to select a set of predefined logic levels. The jumpers J1, J2, and J3 are selecting the voltage source for the resistors giving the threshold levels. In Position 1 of the jumpers, the reference source with V_{REF} and TL431 is selected. In Position 2 of the jumpers, the voltage is

Status of the S1 and S2 Switches		Status of the Output (1), (2)
S1	S2	
open (R1 disconnected)	open (C5 disconnected)	No tone
closed (R1 connected)	open (C5 disconnected)	Lowest tone
open (R1 disconnected)	closed (C5 connected)	Highest tone
closed (R1 connected)	closed (C5 connected)	Lowest tone
Notes: R2 should not be too high to rise the input level above the lowest threshold V4. The exact tone depends on the relation between R2, R1, V5, V1, V2, V3, and V4. Also V5 > V1 > V2 > V3 > V4, where V5 is the voltage level between R1 and R2 with disconnected input of the probe.		

Table 2—Output tone and status of the S1 and S2 switches when the input is not connected

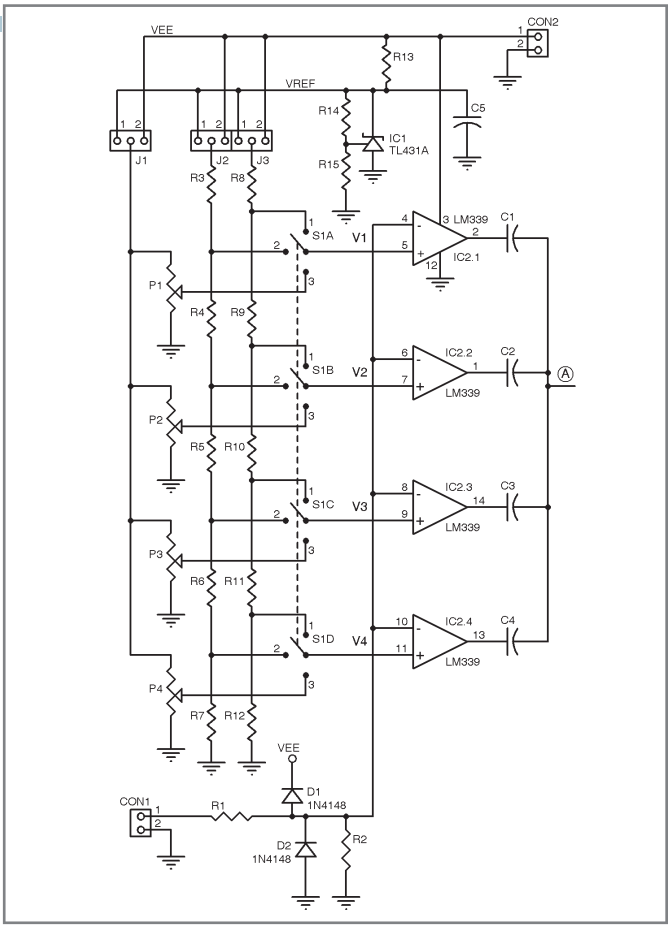


Figure 2—Switching between three sets of threshold levels

taken from the power supply. The values of the components in Figure 2 are not given because it is easy to calculate according to the targeted application.

If the threshold levels are relatively stable and independent on V_{EE} but still lower than the power supply voltage, some sort of voltage reference source should be used. There are a wide range of adjustable low-power voltage references such as National Semiconductor's LM385 series of adjustable micropower voltage references. Other options include Texas Instruments's TL430 adjustable shunt regulator, TL431/2 series of adjustable precision shunt regulators, TL1431 precision adjustable (programmable) shunt reference, and so forth. But if you have a TL317 (100 mA) it could also do the job because here high precision is not needed. Consequently, the input of the probe could be changed accordingly (see Figure 3). Resistors R1–R9 may be any appropriate value or replaced with trimmer potentiometers so the required set of voltage levels could be adjusted. Switch S1 in Figure 3 selects the voltage source for resistors R6–R9. Switch S2 selects the reference voltage V_{REF}. Switch S3 connects or disconnects capacitor C8.

THE SQUARE-WAVE GENERATOR

The audible signal in Figure 1 is produced by a square-wave generator built with a CMOS Schmitt trigger. The frequency of the produced signal on output pin 2 of IC 2.1 is given with the approximate formula below:

$$F = \frac{1}{(0.8 \times R9 \times C)}, \ [\text{Hz}]$$

C is the capacitor connected between input pin 1 of IC 2.1 and ground. The capacitor C5 is always connected and it determines the highest possible frequency, which is selected to be between 5 and 8 kHz.

Depending on the input level of the signal, the comparators in LM339 are connecting capacitors C1, C2, C3, and C4 in parallel to C5 and the combination of the connected capacitors is determining the produced frequency. The signal from the output of IC 2.1 is buffered by five parallel inverters IC 2.2–IC 2.6. The output current is limited by R11 and filtered to be more pleasant by R12 and C8.

The sound is produced with the high-impedance loudspeaker LSP1. You can use low-cost headphones from a Walkman, Discman, or MP3 player. The advantage of the headphones is that the produced sounds don't disturb others.

If higher output current is needed, 74C/HC/HCT14 could be replaced with a NE55 timer (i.e., those manufactured by STMicroelectronics, Fairchild Semiconductor, NPX Semiconductors, Texas Instruments, etc.) or with its CMOS equivalent, but the latter will give lower output current than the old NE555. A solution with the output stage with the timer is shown in Figure 4. The calculations of the components and generated frequencies could be done with the formulas available in the datasheets of National Semiconductor's LM555 timer and LMC555C CMOS timer, NXP Semiconductors's ICM755 CMOS timer, and similar timers.

The circuit's power supply is the full-power supply range of the IC (see Table 3). The obvious limitation came from the IC with the shortest supply range.

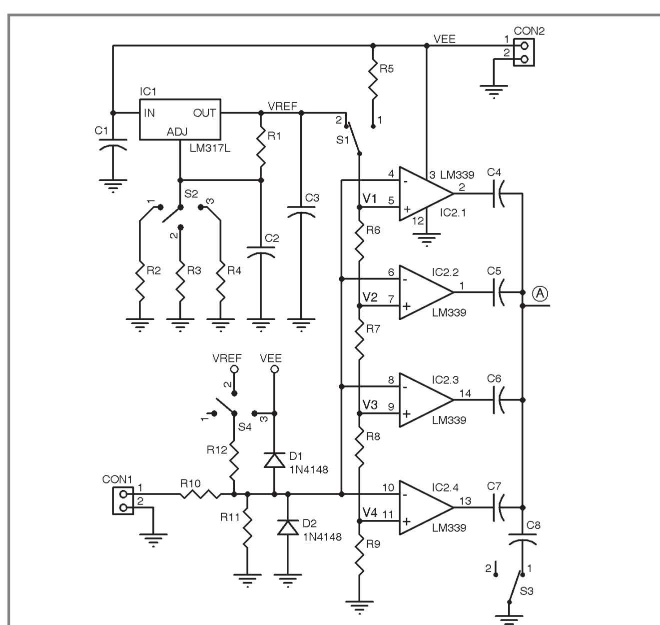


Figure 3—Switching between several reference levels with LM317L and single-resistive network


ROUTE FASTER !

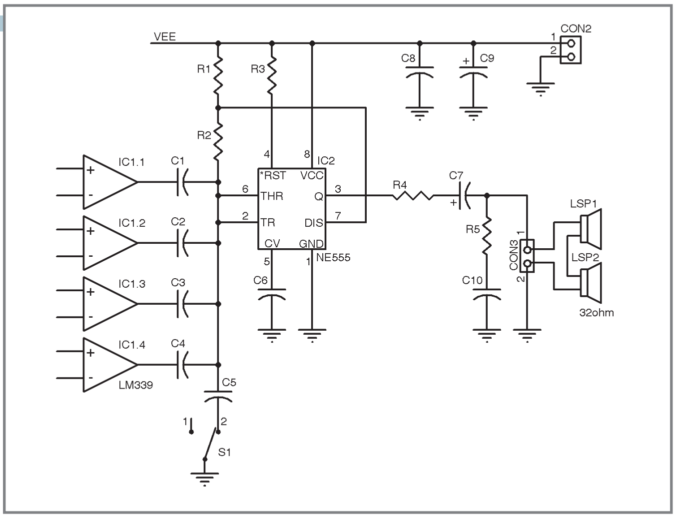


Figure 4—The square-wave generator and the output stage with the timer NE555

EXPANSION & CUSTOMIZATION

Some may find that four levels of comparison are not enough. The circuit could be easily expanded to eight levels by adding a second comparator LM339 or to six levels by adding one LM393. The frequency could be calculated according personal preference.

Also, there is a choice between several working principles: When the probe is switched on, there is an audible signal that will change when a higher voltage level is applied. When the probe is switched on there is no audible signal when the input signal is lower than the lowest threshold voltage level. The input level of the probe should be shifted to ground by connecting the input to either a voltage divisor or a voltage source. The threshold levels could be switched to several fixed values or made adjustable with a trimmer according to the particular application. You can invert the logic of the comparators in order to change the correspondence between frequencies and voltage levels or connect and disconnect an inverter at the output of each comparator. You may have a higher output tone at a higher input voltage or a lower output tone at a higher input voltage. You may add hysteresis to the comparators. (Usually 20 to 50 mV is enough.)

All of these working principles could be implemented in a single probe or in several audible probes, depending on the user's preference.

AUDIBLE PROBE MCU SOLUTION

Today microcontrollers (MCUs) are found in almost any electrical and electronic equipment. Why not put one in an audible probe? Texas Instruments, STMicroelectronics, Microchip Technology, and many more manufacturers offer low-cost MCUs in easy-to-use, eight-to-20-pin DIP/DIL packages appropriate for that application. For the audio probe, you will need an analog-to-digital converter (ADC) that is at least 8 bits to measure the input signal, several digital inputs to select the desired mode of operation, and output capable of driving a high-impedance loudspeaker or headphones. Frequently, several outputs of the MCU could be easily paralleled in order to produce higher output current.

MONITORING WITH AN AUDIBLE PROBE

Sometimes it is necessary to have an audible signal when monitoring input or output signals to see when they are changing their levels. The audible probe described here could help. The input is connected to the monitored signal and the threshold levels are adjusted accordingly. The oscilloscope's screen does not have to be constantly watched, and the embedded system software does not have to be modified to provide a signal when a signal changes. Multi-input audible probes are developed for that purpose but they will not be described here.

CHECKING 12-V INSTALLATIONS

The audible probe is useful for checking 12-V installations driven by car batteries. The threshold levels are adjusted to the required threshold points. Most of the 12-V installations do not experience voltage higher than 16 V, so the application of the probe does not create problems. But, in an installation with running motors and with large inductors or long wires, it is possible to have pulses much higher than 16 V, so appropriate precautions should be taken.

BILL OF MATERIALS

Table 4 shows the parts of the circuit in Figure 1. The values are not critical and could be adapted to the particular application. All resistors are 0.25 W, ±5% and exceptions are noted. All nonpolarized capacitors are for 63 V. Usage of 50-V capacitors is also possible. Exceptions are also noted. The polarities and the voltage ratings of the polarized capacitors are also shown on Figure 1. LM339 could be replaced with two dual comparators LM393.

PROBE LOGIC

I presented a simple single-input multilevel audible logical probe that is appropriate for checking equipment with transistor-transistor logic (TTL) and CMOS ICs or other technologies for installations powered from 12-V car battery.

The probe is not handheld and the size of the implementation is not critical. In most of the applications it is important to keep the input capacitance as low as possible (e.g., less than 100 pF) and the input resistance as high as possible (e.g., more than 50 kΩ). Also, the probe could be used for checking digital equipment including systems with microcontrollers and microprocessors and interface circuits for these systems.

The logical probe from Figure 1 does not require any

IC	Approximate Power Supply Range
LM339, MC3302, LM2901	2 V to 36 V
74HC14	2 V to 6 V (max 7 V)
74C14, CD40106B	3 V to 15 V (max 18 V)
NE555	4.5 V to 16 V
SE555	4.5 V to 18 V
LM317L, TL317 (100 mA)	to 35 V ($V_{INPUT} – V_{OUTPUT}$)

Table 3—Power supply range of the IC in the audible probe

Component	Value	Component	Value
R1	91 kΩ	C1	22 nF/63 V
R2	200 kΩ	C2	10 nF/63 V
R3	1 kΩ	C3	5.1 nF/63 V
R4	15 kΩ	C4	2.7 nF/63 V
R5	7.5 kΩ	C5*	2.2 nF/63 V
R6	9.1 kΩ	C6	330 nF/63 V
R7	3 kΩ	C7	100 µF/16 V (100–1,000 µF/25 V)
R8	3 kΩ	C8*	22 nF/63 V (22–100 nF/63 V)
R9	120 kΩ	IC1	Quadruple comparator LM339, MC3302, LM2901. Possible replacement with two dual comparators LM393 or similar.
R10	1 kΩ	IC2	74HCT14, 74HCT14, 74C14, CD40106 or similar depending on the power supply range
R11	220 Ω (100–620 Ω)	D1*, D2*	1N4148, 1N914
R12*	10 Ω	D3	Any LED with visible light for 0.5–2 mA
CON1, CON2, CON3	Any appropriate connector	LSP1	High-impedance loudspeaker (preferably > 8 Ω), or 32–2,000-Ω headphones. Possibly two 32-Ω low-cost headphones in series.
S1*, S2*	Any appropriate switch or jumper	–	–
*These components could be omitted or adapted to the particular application			

Table 4—Components in the simple audible logical probe

adjustment. During maintenance, the input levels should be tested. The proposed solution is flexible. You can change the threshold levels V1, V2, V3, V4, and V5 to adapt the circuit to particular application. If the probe is damaged, you can easily repair it without any readjustments. ■

Petre Tzvetanov Petrov (ptzvp1@yahoo.fr) has an MS in Electrical Engineering and an MS in Applied Mathematics. He has worked as a researcher and assistant professor at the Technical University of Sofia (Bulgaria) and as a lecturer in Morocco. Petre currently works as an engineer in Bulgaria. His interests include hardware and software engineering, testing, signal sampling, conversion and reconstruction systems, embedded systems, and audio applications. Petre authored/coauthored six books and more than 150 published technical and scientific papers.

SOURCES

NE555 Single timer
Fairchild Semiconductor Corp. | www.fairchildsemi.com

MC3302 Op-amp
Freescale Semiconductor, Inc. | www.freescale.com

LM339 and LM2901 Quad comparator, LM385 voltage reference, MM74C14 Hex Schmitt trigger, LM555 timer, and LMC555 CMOS timer
National Semiconductor Corp. | www.national.com

ICM7555 CMOS Timer
NXP Semiconductors | www.nxp.com

TL430 Adjustable shunt regulator, TL431/2 series of shunt regulators, and TL1431 shunt reference
Texas Instruments, Inc. | www.ti.com

by Matt Oppenheim (United Kingdom)

Audio-Enhanced Touch Sensors

Using a touch sensor, you can trigger audio tags on electronic devices. Here you learn how to couple capacitive-touch sensors with an Android device.

Adding touch sensors that trigger audio tags to the controls of everyday objects can help visually impaired users operate an unfamiliar device. One solution is to couple capacitive-touch sensors with an Android device.

I built a scalable, 16-channel touch sensor that interfaces with a PC or an Android smartphone. I installed these in some audio players to help visually impaired users. When the user touches a control, an audio tag is triggered that tells the user what the button does before it performs the function.

I used Microchip Technology's mTouch capacitance touch-sensor technology to instigate the touch sensors. I built a clip-on light-emitting diode (LED) board to show which channels are active and a battery board that powers the assembly from a single AAA battery. Photo 1 shows the completed boards and how they are mounted on a CD player.

Two implementations of the touch sensor are presented. The first uses a PC or laptop to produce the audio tags. The second is a fully portable system using an Android smartphone to produce and optionally record the audio tags.

This article explains how this technology works and how I built the touch-sensor boards. Firmware using MikroElekronica's mikroC PRO complier and software development using Java are also described.

CAPACITIVE-TOUCH SENSING

During my annual run around campus I got to thinking about how capacitive-touch sensors work. Bear with me in this analogy. On the road surface I get a firm rebound and trundle along with the low but constant rate of energy expenditure needed to maintain my pace. Then I head onto the soggy playing fields and my feet begin to trace a much more ragged path. To maintain the same pace, I would have to work harder. But I can't, as I am already maxed out. However, my stopwatch continues its merciless count unaffected by how much I have slowed down. The analogy with how capacitive-touch sensors work is that the user adds capacitance to a line directly connected to an oscillator—the capacitance-sensing oscillator (CSO). This capacitance acts in the same way that a soggy surface acts on me as I run—it slows down the oscillator. If the oscillator could add more current, it could maintain its pace, just as if I could add more energy to my running, I could maintain my pace. But the capacitive-touch sensor line is designed to be fixed current, so it will slow down. The user has access to a register to set this current to one of three preset values to enable a range of different size touchpads. By monitoring the frequency of the CSO using an isolated internal counter (which corresponds to my wrist watch when I am running), you can measure that the CSO slows down as it is loaded. Similarly, when the finger is removed from the touch sensor, the decrease in

Photo 1a—Two daisy-chained Microchip Technology mTouch boards with a battery board providing the power and LED boards showing the channel status. b—A single mTouch board attached to a CD player with an LED board attached. c—This is a close up of the buttons on the CD player. The thin laminated wire connects to the mTouch board and triggers the detector when touched.

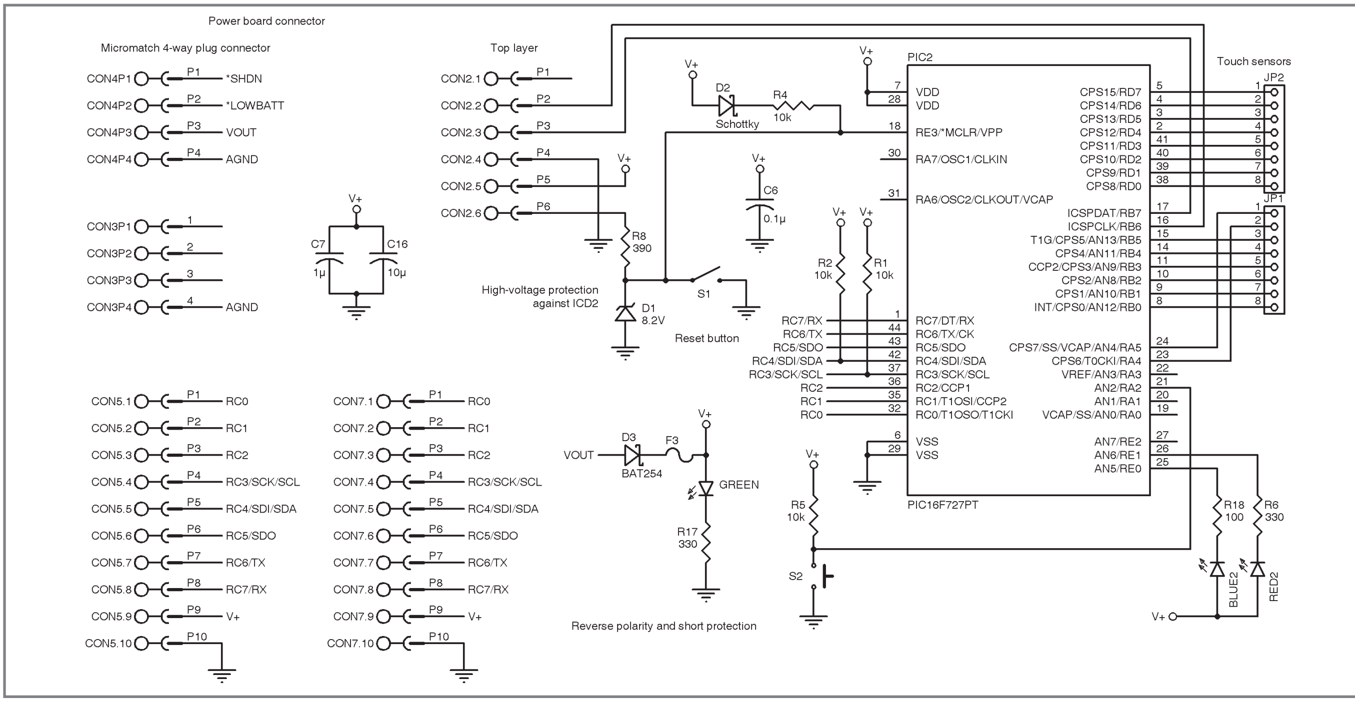


Figure 1—The main touch sensor board features a Microchip Technology PIC16F727 microcontroller.

capacitance will cause the CSO to speed back up again.

mTOUCH BOARD

In my article, "Multichannel Touch Sensors: Implement Scalable Capacitive Touch Sensing" (*Circuit Cellar* 234, 2010), I described a scalable touch-sensor system I built based on Atmel's QProx QT1103 capacitive-touch sensor chip. Microchip brought out a range of microcontrollers after I finished that project, which makes adding touch sensors a bit more straightforward for the homebrew engineer. Microchip calls the technology mTouch. You have to set up the touch sensors in firmware, which is a bit less convenient than using the dedicated QProx solution. But, having a microcontroller wrapped around the touch sensor makes adding communications, flashing LEDs, and button controls easy to implement. Plus, the chips come with legs which makes home assembly a lot less painful than the legless QFN32 package available in the QT1103!

My main mTouch board is shown in Figure 1 and Figure 2. I used Microchip's PIC16F727 microcontroller, which has 16 channels available to configure as touch sensors. A single-capacitance CSO is multiplexed between these channels.

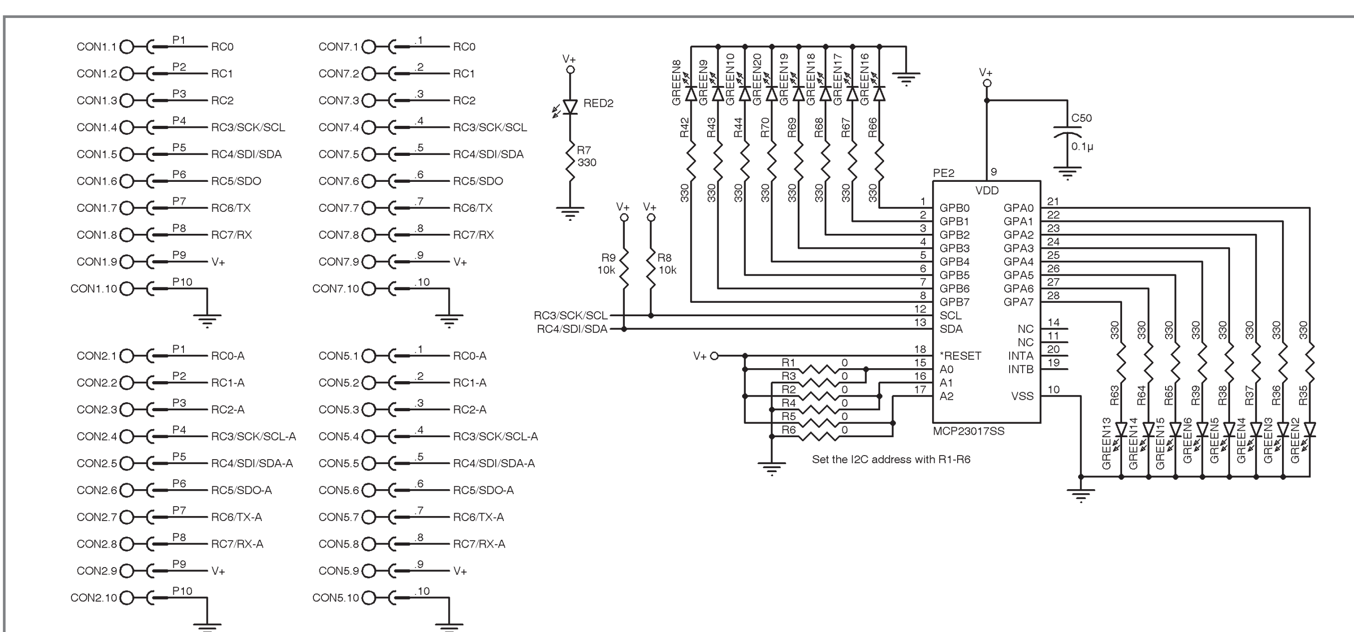


Figure 2—The LED board indicates which touch sensor channels are active.

The 44-pin TQFP version of the chip is not too tricky to hand-solder. Lightly flex a scalpel blade against each pin after tacking it down to verify that the solder joint is solid. The chip's datasheet recommends protecting the MCLR line against overvoltage protection as the Microchip MPLAB ICD 2 in-circuit debugger/programmer can take it dangerously high during programming. This is provided by the Zener diode D1 and resistor R8.

Listing 1—In the while loop, each capacitive-touch sensor channel is repeatedly configured until TMR0 overflows. The time taken to do this is compared each time the loop repeats. A change indicates that a touch event has occurred.

```
while(1) {
   channelNumber = startChannel;
   reset_system();
   while (channelNumber <= lastChannel){  // scan through each touch channel.
      configurePort(channelNumber);// repeated until timer overflows
      if (interrupt_alarm) { // s/w interrupt alarm bit set by ISR
         alarmInterrupt(channelNumber); // handle interrupt flag
         reset_system(); // Start again with a clean slate
         channelNumber++ ;
       } // end if
   } // end while
   update_channel_status(); //  updates channel status register
   //noChange is a flag for when channel status is changed.
   if (noChange == false) { // changed i/p, so change LEDs
      mcp_activechannels(); // work out which LEDs to light on board
      mcp_update(LEDboard, MCPportA, MCPportB);// activate LEDs on board with status
      noChange = true; // reset flag
   } // end if
}   // end while(1)
```

Two LEDs connected to the PIC enable simple debugging flashes and an "I'm alive" signal during start-up. I placed a Reset button on the *MCLR line. The firmware recalibrates itself each time the board is reset. This means if you twist up the touch sensor wires and they start to randomly trigger all you have to do is reset the board and you are good to go again. Of course, you can just toggle the power switch to do this as well. I guess I'm trying to justify habit—I've always put a Reset button on my microcontroller boards.

I designed and laid out my two-layer printed circuit board (PCB) using Eagle v5.60. I'm too stingy to pay for silkscreen, so I put my text onto the copper layers. I have learned the hard way. Always print the version of board onto the PCB to avoid later confusion.

CONNECTORS

I use a six-way Tyco Electronics Micro-MaTch header to connect the ICD2–CON2 shown in Figure 1. I like Tyco's Micro-MaTch connectors as they are keyed and give a positive click when connected. They are available in a range of styles: through-pin, surface-mount, and crimp-on connectors for ribbon cable. The ubiquitous 0.1" unkeyed header pins commonly used for connecting programmers just ask you to connect your programmer the wrong way around and potentially connect the board's ground to your power rail! A pair of four-way Micro-MaTch connectors (CON3 and CON4) enables me to clip on a DC-to-DC converter board with an AAA battery to power the assembly. The pair of 10-way connectors enables access to the I^{2}C, SPI, UART, power, and three signal lines. As both of these connectors share the same signals, I can use them to daisy chain several boards together using the corresponding crimp-on headers and a length of ribbon cable (see Photo 1a). An LED board clips across these connectors to show which of the channels are active.

I crimped a 10-way Micro-MaTch connector to a Future Technology Devices International (FTDI) UART-to-USB cable to enable communication between the PIC's UART and the PC. For this application, I only sent data one way, from the PIC, but I have tested two-way communications using this hardware in another project. Using the FTDI cable enabled detailed debugging messages to be displayed on the serial port terminal of the mikroC PRO compiler that I use to develop firmware. In the finished device, the cable enables the touch-sensor status to be sent to the PC to trigger an audio tag for each channel. The cable also enables the boards to be powered from the 5-V rail on the USB port.

Each of the 16 touch-sensor channels are routed directly to 0.1" header pins, eight on each side of the board. A corresponding strip of header sockets slides onto these pins, which enables the boards to be removed from the devices they are being used with. Thin laminated wire is soldered to the back of these sockets. The connectors and the laminated wire are shown in Photo 1b and Photo 1c. Thin laminated wire is soldered to the back of these sockets. These wires run to where the touch sensors are required. In my case, they run to the top of the radio and CD player controls. On the CD player, I made a little coil of wire on the area where I needed the touch sensor and stuck it down with a thin layer of clear adhesive. On the radio, I was able to use a needle to stitch the laminate wire through the rubber buttons.

LED BOARD

I made a clip-on LED board that indicates the active channels. The touch sensors run fine without the board attached, but the lights look great and are useful for checking that the touch pads are correctly connected.

All LEDs are not created equal. Check the brightness rating, expressed in millicandela (mcd), the candela being a unit of luminosity. You can get more bang for your buck or more mcd for your milliamp by paying a little more for your LEDs.

I used Microchip's MCP23017 I/O expander to drive the LEDs. This interfaces with the PIC16F727 using the I^{2}C communication protocol. Prior to this project, I used the SPI version of the port expander as I thought the extra data line this protocol uses is somehow more robust. But Jeff Bachiochi's article, "Extend and Isolate the I^{2}C Bus" (*Circuit Cellar* 233, 2009), made me reconsider. I^{2}C only ties up two lines and is

Listing 2—Bit handling definitions and configuration and interrupt handling subroutines. The ISR deals with TMR0 overflowing. The complete listing on *Circuit Cellar*'s FTP site deals with other interrupts.

```
#define bit(num) (1 << num) // creates a bit mask
#define bit_set(v, m) ((v) |=(m)) // Sets the bit
// e.g. bit_set (PORTD, bit(0) | bit(1));
#define bit_clear(v, m) ((v) &=  ~(m)) // Clears the bit
#define bit_toggle(v, m) ((v) ^= (m)) // toggle the bit
#define bit_read(v, m) ((v) & (m))  // read a bit and see if it is set
#define bit_test(v,m)      ((v) && (m))
#define startChannel 0
#define lastChannel  15

// Configure tris & CPSCON1 settings for channel
void configurePort(char channel) {
   CPSCON1 = channel; // Set CSO to active channel
   //uart_write_short(CPSCON1); // debug info
   if (channel < 6) {
     TRISB = TRISBCapOscOn[channel]; // Channels 0-5 on portB
   } else if (channel < 8){
     TRISA = TRISACapOscOn[channel]; // Channels 6-7 on portA
   }else {
     TRISD = TRISDCapOscOn[channel]; // Channels 8-15 on portD
   }
}

void scan_channels(){ // get CSO count for each channel
   char channel;
   while (channel <= lastChannel){
     configurePort(channel);// repeated until timer overflows
     if (interrupt_alarm) { // s/w interrupt alarm bit set by ISR
       alarmInterrupt(channel); // handle interrupt flag
       reset_system(); // Start again with a clean slate
       channel++ ;
     }// end if
   } // end while
}// end scan_channels

void interrupt() { // Interrupt Service Routines
   INTCON.GIE = 0; // Global disable interrupts
   if (INTCON.T0IF) { //tmr0 overflow
     CPSCON0.CPSON = 0;    // Turn cap sense off - disable tmr0
     T1CON.TMR1ON = 0; //Turn tmr1 off
     INTCON.T0IF = 0; // Reset tmr0 interrupt flag
   } // end if
}// end interrupt()

void alarmInterrupt(short intchannel) { // deal with interrupts outside of ISR
   if bit_read(interrupt_alarm,t0Interrupt) { //tmr0 interrupt flag
     disable_interrupts(); // The LCD write will cause timer IF
     capCount[intchannel] = TMR1H; // CSO count from MSB of tmr1
     bit_clear(interrupt_alarm, t0Interrupt);  // clear alarm flag for tmr0 interrupt
   }
} // end alarmInterrupt
```

plenty fast enough to cope with this application. The chip can be given a unique I²C address by pulling the address lines A0–A2 high or low using 0-Ω resistors on pads R1–R6. Photo 1a shows two boards daisy chained together. Each has its own LED board with different I²C addresses. Naturally, you have to change the firmware for the mTouch board to match the LED board's I²C address! I defined the I²C addresses of the board's statements such as `#define board2 0x44` at the start of my code, then refer to `board2` throughout the rest of the code.

I placed the same 10-way port expander sockets the LED board clips onto on the top side of the LED board. This enables you to connect the stack to a PC using the FTDI cable or to daisy chain several boards together with the LED boards attached.

FIRMWARE

I developed the firmware for the PIC16F727 using the mikroC PRO compiler v4.60 programming environment which is a free download. I have yet to exceed the demo version's size limit. The final code is available on the *Circuit Cellar* FTP site. I based my code on the example C code supplied by Microchip for the mTouch technology.

Once the various registers are initialized, the program enters an endless while loop—which is a standard way of writing microcontroller code (see Listing 1). In this loop, each of the touch-sensor channels is connected to the CSO in turn. The CSO clocks the 8-bit timer register, TMR0, until it overflows. This triggers an interrupt that is handled by the interrupt service routine (ISR) called `interrupt()`. This resets TMR0 and sets the flag `interrupt_alarm` to be true (see Listing 2). The program then branches to the subroutine `alarmInterrupt (channelNumber)`. I could have put the code from this subroutine into the ISR, but good practice is to keep the ISR as short as practical—in case another interrupt comes along while you are already in the ISR. This could put you into a situation where you never manage to get out of the ISR! There are a couple of other possible interrupt situations, such as the UART triggering an interrupt if you set up two-way communication. These are dealt with in the complete listing on the FTP site.

So, touching a channel will cause the CSO to slow down while it is connected to that channel. As the CSO clocks the 8-bit counter TMR0, this will take longer to overflow. Meanwhile the 16-bit counter TMR1 is being clocked by the internal oscillator in the PIC. The rate of increase of TMR1 is unaffected by touching the channel, so it will have a higher value than usual when TMR0 overflows. If TMR1 starts to overflow instead of the smaller TMR0, this would be a reason to change the current setting bits, which are called CPSRNG0 and CPSRNG1, in the CPSCON0 register. This controls what fixed current is available to the CSO, which in turn controls how much the CSO will slow when it is loaded with the extra capacitance of a finger touch.

In `alarmInterrupt`, the most-significant byte of TMR1, is stored in an array as `capCount[intchannel]`, which represents the capacitance of that channel. If there is an increase in this value, the channel is flagged as having been touched. Similarly, once touched, if there is a decrease, the channel is flagged as no longer being touched.

Note the definitions of `bit_set`, `bit_clear`, `bit_read`, and `bit_test` at the start of Listing 2. I got this idea from "Bit Flipping Tutorial: An Uncomplicated Guide to Controlling MCU Functionality" by Eric Weddington (*Circuit Cellar* 180, 2005).

I use the Microchip PICkit2 programmer/debugger for my programming. The standalone interface it uses can be configured to automatically load a hex file whenever you recompile your code. This makes it easy to use your choice of C compiler. I made a programming cable by crimping a six-way Micro-MaTch plug onto some ribbon cable, then soldering the other end of the cable to standard 0.1" header pins, which slots into the PICkit2. You can power the board via the programmer, which can be a useful option while debugging your code—in the unlikely event your code ever needs debugging.

My code works. The Microchip example has a time-averaging function to compensate for changing atmospherics starting to trigger the detectors. This may be something to incorporate at a later date if I start having problems.

JAVA GUI

As mentioned, I have two setups for the boards. For some of my research on human-computer interaction, I need to connect the boards to a PC to accurately log timing information.

I developed a graphical user interface (GUI) on my PC that displays the status of each channel and plays an audio tag whenever a touch sensor triggers. Luckily, I work in a computer science research lab, so I trade soldering for Java tuition. I use the freely available Eclipse programming environment for Java development. Oracle, which now develops Java, provides some excellent Java tutorials online.

A screenshot of the GUI is shown in Photo 2. There is a button control for each channel that lights up as a channel is triggered. You can also click on the button to test the audio clip that is associated with that channel. A record of when each channel is triggered and released is displayed and can be logged to a text file. The sound can be toggled on and off. As you move your finger from one touch sensor to another, the audio from the previous channel is killed to prevent a confusing cacophony of overlapping sound tags.

I used the gnu.io library to service the virtual com port set up by the FTDI driver. This enables me to treat the interface as a serial port. The audio clips are all .wav snippets that are held in a hashtable structure. I used Swing components to build the interface.

To make the .wav files, I recorded my own voice using High Criteria's Total Recorder audio and video recording software program. I used the same program to edit the files to remove any dead space at the start of the recording, so there is immediate audio as soon one of the sensors is touched. The full Java listings are on *Circuit Cellar*'s FTP site.

PORTABLE ANDROID VERSION

The touch-sensor circuitry can be discretely stuck onto the back of the devices I am enhancing. Having a

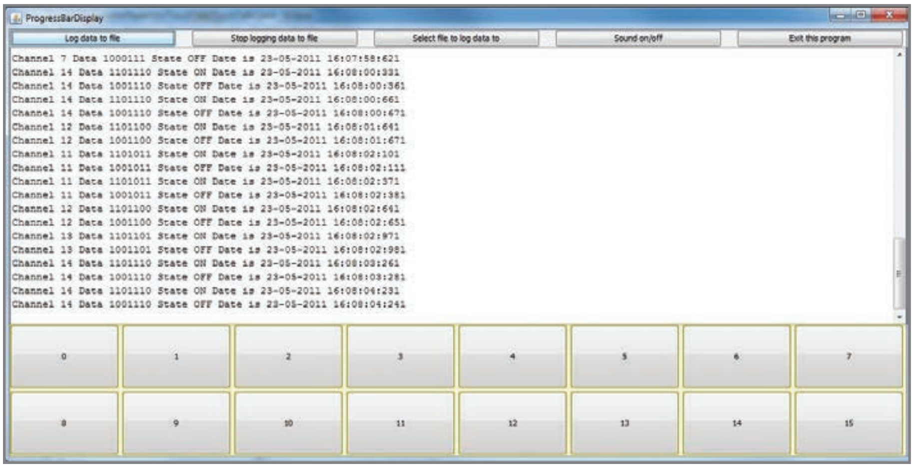


Photo 2—The Java GUI for the mTouch board. Each button represents a channel. The channel statuses are updated in the text area.

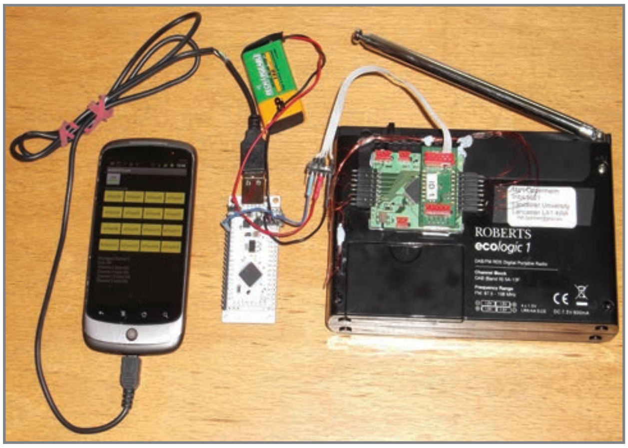


Photo 3—The portable touch-sensor assembly. The touch-sensor boards are mounted on the back of a digital radio, connected to a IOIO board and a Nexus One smartphone. The Android interface is displayed on the phone.

portable system for generating and recording the audio tags means I can leave an enhanced device with a visually impaired user until he or she has learned the layout of the controls. Then I can disconnect the touch-sensor boards and take away the extra hardware. The sensor wires that have been added to the device can stay in place with no effect on the usability of the player.

I started by prototyping a board based on a Nuvoton Technology ISD ChipCorder speech recorder IC. But the quality of sound was never going to be great due to the low bit-rate sampling. The recording capacity on the chips is adequate for the implementation shown in this article, but I have plans to daisy chain a number of boards together to add tags to a keyboard. I settled on using an Android phone as it already has a speaker, a microphone, and a lot of memory to store as many audio tags as I could ever need. To add audio to the handful of devices I am looking at enhancing, buying a few second-hand phones is not going to be more expensive than paying for a few custom-made boards.

Detailing how to program an Android phone is an article in itself. I used Java running on the Eclipse development platform with a plug-in produced by Google. Luckily, the Java skills I acquired through programming the interface on the PC could be reused for programming the Android phone. The Android interface is shown in Photo 3. Activating a touch sensor causes the corresponding button to flash and the audio tag to be triggered. If you need more volume than the phone can deliver, it is easy to add a portable speaker.

Audio tags can be directly recorded using the phone's built-in microphone or preloaded into memory. I found that preloading the audio files provided the best quality. I used a separate program to generate audio tags using Google's text-to-speech library.

But how could I get the data from the touch-sensor boards into the phone? This is where the IOIO (pronounced "yo-yo") board developed by

SparkFun Electronics and Ytai Ben-Tsvi, a Google engineer, came to the rescue. The board uses a Microchip PIC24FJ256DA206 microcontroller from the PIC24F family, which has USB on-the-go built in, so it acts as a USB host. The phone is hosted by the board. The IOIO comes preloaded with firmware that enables you to easily control the interfaces on the PIC24 (UART, SPI, I²C, etc.) directly from the phone. Adding the IOIO library to your Eclipse project enables you to access these interfaces.

I looked at some other solutions for interfacing my phone with the board, which gives greater control of the interface board by directly programming the board's firmware. But I couldn't get them to be as reliable as the IOIO board. I'm probably doing something wrong and I hope to return to the more advanced boards for a future project. The IOIO board proved to be a simple and reliable solution.

One of the stumbling blocks in using Android phones for data collection is that the phone will try to recharge itself whenever you connect it to a USB port. This means being able to supply up to 500 mA, which is a tall order for a battery-powered portable device. The way around this is to solder in a 1-kΩ resistor to the voltage line (red wire) in the USB cable that you use to connect the phone to your interface board. The phone will connect, but it won't recharge itself through the connection. Thanks to *Dr. Monk's DIY Electronics Blog* for putting this information onto the blog.

The full assembly, which shows the touch-sensor board mounted on the back of a digital radio connected to the IOIO board and a Nexus One Android phone is shown in Photo 3.

POWER

"Power is nothing without control." But it is also hard to control anything without power! The FTDI cable provides power to the mTouch board when it is directly connected to a USB port. The portable version runs from a 9-V battery that is connected to the IOIO board.

I built a small DC-to-DC converter board that clips directly onto the touch-sensor board. This is useful for demos involving only the touch sensor and LED board (see Photo 1a). Running your finger along the pins of

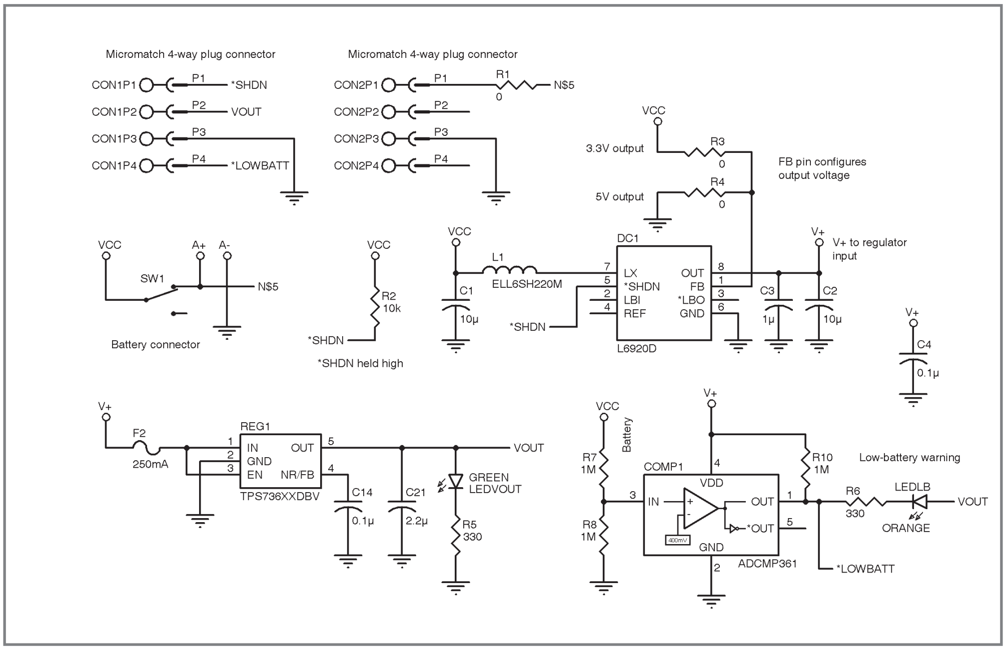


Figure 3—DC-to-DC converter board with a selectable output of 3.3 V or 5 V

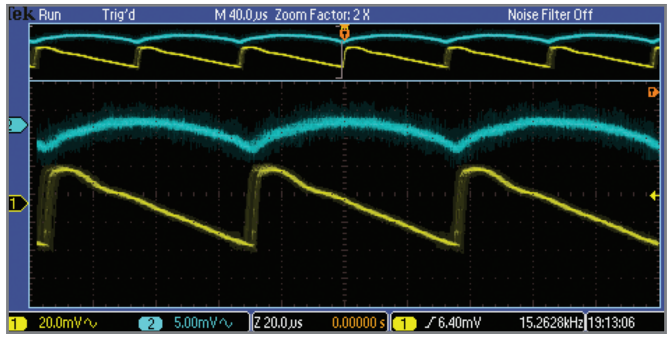


Photo 4—A 3-V DC-to-DC converter: regulated output on top, DC-DC output on bottom. Note the different traces on the scales.

the touch-sensor boards makes for a nice little light show to demonstrate your technology. The battery board runs from a single AA or AAA battery and uses an STMicroelectronics L6920D step-up converter (DC1 in Figure 3) to boost the voltage. The output of the converter can be set to either 3.3 or 5 V by connecting the feedback (FB) pin to the output (OUT) pin or to ground. This is set by populating either R3 or R4 in Figure 3 with a 0-Ω resistor. As with all DC-to-DC converters, there is an appreciable ripple on the output. Touch-sensor circuits can be sensitive to a ripple in the voltage rail, so I use the 5-V output option and an additional regulator stage to reduce this ripple (REG1 on Figure 3). Photo 4 shows the ripple from the converter and the smoother output from the 3.3-V regulator. Note the different scales on the traces. Clearly, the regulator significantly reduces the ripple. This ability is listed as power supply rejection ratio (PSRR) on the component's datasheet. During my testing, I used the smooth 5-V USB power rail available from the FTDI cable to power the device. But having the clip-on battery board enables me to show off the touch sensor and LED boards at meetings. The smoothed output from the regulator will be useful for future analog designs I have in the pipeline.

The touch sensor and LED boards are happy to run at either 3.3 V from the regulated DC-to-DC converter or 5 V from the USB port. If you are using the IOIO board, use the 3.3-V pin on the IOIO board to connect to the touch-sensor board's +V rail.

TESTING & FUTURE WORK

The initial application of these boards was to add audio tags to the controls of a CD player, a cassette player, and a digital radio to help the visually impaired learn the layout of an unfamiliar device. When the controls are touched, an audio tag states the function (e.g., play or stop) before the button is pushed.

The devices have been tested by visually impaired volunteers who like the idea. As expected, the technology is most useful with the more complex digital radio as it has the most controls. Wiring up a complete computer or music keyboard is a practical idea as the boards can be daisy chained together and the Android device has the hardware to cope. The Android code can easily be extended to cope with the extra channels.

My hope is that manufacturers will adopt the concept of adding touch sensors and audio tags to the controls of devices geared toward the visually impaired community. The Roberts Symphony CD player I used for testing is one of the few CD players with high visibility and tactile buttons manufactured specifically for the visually impaired. But, for people who can't see the buttons, the added audio tags make it even easier to use.

Author's note: The Lancaster University Faculty of Science and Technology supported this work through a faculty grant. Andrew Greaves and Robert Hardy of InfoLab21 supplied invaluable help by coaching my Java.

Matt Oppenheim (matt.oppenheim@gmail.com) is a geophysicist for Polarcus Ltd. who works onboard seismic survey ships. He spends his time onshore at InfoLab21, Lancaster University, working in the Embedded Interactive Systems group. By nature a hardware specialist, Matt realizes that software is a necessary evil.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/262.

RESOURCES

Android Developers, http://developer.android.com/index.html.

J. Bachiochi, "Extend and Isolate the I²C BUS," *Circuit Cellar* 233, 2009.

Eclipse Foundation, Inc., www.eclipse.org.

Dr. Monk's DIY Electronics Blog, "Open-Source Hardware, Arduino, IOIO, General DIY Electronic Construction, Reviews, Projects, How-Tos, and Recipes," 2011, http://srmonk.blogspot.com/2011/06/android-open-accessory-without charging.html.

M. Oppenheim, "Multichannel Touch Sensors: Implement Scalable Capacitive Touch Sensing," *Circuit Cellar* 234, 2010.

———, InfoLab21, Lancaster University, "Touch and Speech Enhanced Cassette Player to Help the Visually Impaired," www.youtube.com/watch?v=Z98IRhTjz30.

Oracle, "The Java Tutorials," 2011, http://download.oracle.com/javase/tutorial.

Vogella Blog, "Android Tutorials," www.vogella.de/android.html.

E. Weddington, "Bit Flipping Tutorial: An Uncomplicated Guide to Controlling MCU Functionality," *Circuit Cellar* 180, 2005.

SOURCES

Eagle PCB Software
CadSoft, Inc. | www.cadsoftusa.com

USB TTL Serial cables
Future Technology Devices International (FTDI) Ltd. | www.ftdichip.com

Total Recorder software
High Criteria | www.highcriteria.com

mTouch Capacitive touch evaluation kit, PIC16F727 microcontroller, MPLAB ICD 2 in-circuit debugger/programmer, MCP23017 I/O expander, PICkit2 programmer/debugger, and PIC24FJ256DA206 microcontroller
Microchip Technology, Inc. | www.microchip.com

mikroC PRO compiler
MikroElektronika | www.mikroe.com

ISD ChipCorder speech recorder
Nuvoton Technology Corp. | www.nuvoton.com

IOIO board
SparkFun Electronics | www.sparkfun.com

L6920D Step-up converter
STMicroelectronics | www.st.com

Micro-MaTch Connectors
Tyco Electronics Corp. | www.te.com

QUESTIONS & ANSWERS

On Being a Communications Specialist

An Interview with Lawrence Foltzer

From sonar systems to secure communications, Lawrence Foltzer has had a full career. In March, he described his background in fiber optic communication, telecommunications, and access control methods. We also discussed his interests in infrared communication, direct digital synthesis, and robot navigation.—*Nan Price, Associate Editor*

NAN: Give us a little background information. Where do you live?

LARRY: I have lived in Mountainair, NM, for the last five years. Twenty years in Occidental, CA. Before that, nine years in Scottsdale, AZ. Before that, three years in Roanoke, VA. Before that, eight years in Mount Airy, NC. And four years in Bedford Hills, NY, before that. Prior to that, I lived on a mobile platform known as the USS Moale (DD-693), a Navy destroyer sailing with the 6th fleet in the Mediterranean Sea.

NAN: You spent 30 years working in the fiber optics communication industry. How did that come about? Have you always had an interest specifically in fiber optic technology?

LARRY: My career has taken me many interesting places, working with an amazing group of people, on the cusp of many technologies. I got my first electronics training in the Navy, both operating and maintaining the various anti-submarine warfare systems including the active sonar system; Gertrude, the underwater telephone; and two fire-control electromechanical computers for hedgehog and torpedo targeting. I spent two of my four years in the Navy in schools.

When I got out of the Navy in 1964, I managed to land a job with IBM. I'd applied for a job maintaining computers, but IBM sent me to the Thomas J. Watson Research Center in Yorktown Heights, NY. They gave me several tests on two different visits before hiring me. I was one of four out of forty who got a job. Mine was working in John B. Gunn's group, preparing Gunn-oscillator samples and assisting the physicists in the group in performing both microwave and high-speed pulsed measurements.

One of my sample preparation duties was the application of AuGeNi ohmic contacts on GaAs samples. Ohmic contacts were essential to the proper operation of the Gunn effect, which is a bulk semiconductor phenomenon. Other labs at the research center were also working with GaAs for other devices: the LED, injection laser diode, and Hall-effect sensors to name a few. It turned out that the evaporated AuGeNi contact used on the Gunn devices was superior to the plated AuSnIn contact, so I soon found myself making 40,000 A per square centimeter pulsed-diode lasers. A year later I transferred to Gaithersburg, MD, to IBM-FSD where I was responsible for transferring laser diode technology to the group that made battlefield laser illuminators and optical radars. We used flexible light guides to bring the output from many lasers together to increase beam brightness.

> "One of my sample preparation duties was the application of AuGeNi ohmic contacts on GaAs samples. Ohmic contacts were essential to the proper operation of the Gunn effect, which is a bulk semiconductor phenomenon. It turned out that the evaporated AuGeNi contact used on the Gunn devices was superior to the plated AuSnIn contact, so I soon found myself making 40,000 A per square centimeter pulsed-diode lasers."

As the Vietnam war came to an end, IBM closed down the Laser and Quantum Electronics (LQE) group I was in, but at the same time I received a job offer to join Comsat Labs, Clarksburg, MD, from an engineer for whom I had built Gunn devices for phased array studies. So back to the world of microwaves for a few years where I worked on the satellite qualification of tunnel (Asaki) diodes, Impatt diodes, step-recovery diodes, and GaAs FETs.

About a year after joining Comsat Labs, the former head of the now defunct IBM-LQE group, Bill Culver, called on me to help him prove to the army that a "single-fiber," over-the-hill guided missile could replace the TOW missile and save soldier lives from the target tanks counterfire.

NAN: Tell us about some of your early projects and the types of technologies you used and worked on during that time.

LARRY: So, in 1973-ish, Bill Culver, Gordon Gould (Laser Inventor), and I formed Optelecom, Inc. In those days, when one spoke of fiber optics, one meant fiber bundles. Single fibers were seen as too unreliable, so hundreds of fibers were bundled together so that a loss of tens of fibers only caused a loss of a few percent of the injected light. Furthermore, bundles presented a large cross section to the primitive light sources of the day, which helped increase transmission distances.

Bill remembered seeing one of C. L. Stong's Amateur Scientist columns in *Scientific American* about a beam balance based on a silica fiber suspension. In that column, Stong had shown that silica fibers could be made with tensile strengths 20 times that of steel. So a week later, Bill and I had constructed a fiber drawing apparatus in my basement and we drew the first few meters of fiber of the approximately 350 km of fiber we made in my home until we captured our first army contract and opened an office in Gaithersburg, MD.

Our first fibers were for mechanical-strength development. Optical losses measured hundreds of dBs/km in those days. But our plastic clad silica (PCS) fiber losses pretty much tracked those of Corning, Bell Labs, and ITT-EOPD (Electro-Optics Products Division). Pretty soon we were making 8 dB/km fibers up to 6 km in length. I left Optelecom when follow-on contracts with the army slowed; but by that time we had demonstrated missile payout of 4 km of signal carrying fiber at speeds of 600 ft/s, and slower speed runs from fixed-wing and Helo RPVs. The first video games were born!

At Optelecom I also worked with Gordon Gould on a CO_2 laser-based secure communications system. A ground-based laser interrogated a Stark-effect based modulator and retro-reflector that returned a video signal to the ground station. I designed and developed all of that system's electronics.

Government funding for our fiber payout work diminished, so I joined ITT-EOPD in 1976. In those days, if you needed a connector or a splice, or a pigtailed LED, laser or detector, you made it yourself; and I was good with my hands. So, in addition to running programs to develop fused fiber couplers, etc., I was also in charge of the group that built the emitters and detectors needed to support the transmission systems group.

NAN: You participated in Motorola's IEEE-802 MAC subcommittee on token-passing access control methods. Tell us about that experience.

LARRY: My work on fiber couplers, directional couplers, and stars was in support of optical fiber networks, so I applied to Motorola to guide their network product development. I represented Motorola in the IEEE-802 MAC subcommittee, where my expertise was at the physical layer, while the battle to be fought was on the protocol level between Ethernet and token passing. I guess you all know who won that battle, and I was on the losing side of the argument. Ethernet required custom couplers to maximize system node count and transmission distance, while token networks used repeating nodes. It was just too early for optical networks, primarily because things like erbium doped fiber amplifiers (EDFAs), single-frequency lasers, and wavelength division multiplexing did not yet exist, so optical solutions were messy, to say the least.

Motorola de-emphasized fiber optics, so I joined Maxlight, as small fiber optic company that was later acquired by Raychem. Raychem supposedly had a unique coupler technology called a noninvasive tap. Snake-oil salesmen, all! But they sold an uninquisitive customer on the Raynet coupler story, and took a bath later. Meanwhile, the passive optical network (PON) myth persists while component costs are high and volumes low; and upgrades for increased capacity means dropping "Passive" from the lexicon of PON, or replacing expensive infrastructure.

The last 10 years of my career have been with Optilink, DSC, Alcatel, and Turin Networks. All were based in Petaluma, CA. SONET was the reason for being for these and other companies, but Ethernet has once again captured the mindset of the network providers.

"I was in Motorola's strategic marketing department (SMD) when the Apple 2 first came on the scene. I quickly became a fan of the MC6809 CPU, and wrote some pretty fancy code for the day that rotated 3-D objects, and a more animated version of Space Invaders."

NAN: How long have you been designing MCU-based systems? Tell us about your first MCU-based design.

LARRY: I was in Motorola's strategic marketing department (SMD) when the Apple 2 first came on the scene. Some of the folks in the SMD were the developers of the RadioShack color computer. Long story short, I quickly became a fan of the MC6809 CPU, and wrote some pretty fancy code for the day that rotated 3-D objects, and a more animated version of Space Invaders. I developed a menu-driven EPROM programmer that could program all of the EPROMs then available and then some. My company, Computer Accessories of AZ, advertised in *Rainbow* magazine until the PC savaged the market. I sold about 1,200 programmers and a few other products before closing up shop.

NAN: How long have you been reading *Circuit Cellar*?

LARRY: Way back in the beginning. But not on a continuous basis.

NAN: *Circuit Cellar* has published four of your articles about design projects. Your first article, "Long-Range Infrared Communications" was published in 1993 (*Circuit Cellar* 35). Which advances in IR technology have most impressed and excited you since then?

LARRY: Vertical cavity surface-emitting lasers (VCSEL). The Japanese were the first to realize their potential, but did not participate in their early development. Honeywell Optoelectronics was the first to offer 850-nm VCSELs commercially. I think I bought my first VCSELs from Hamilton Avnet in the late 1980s for $6 a pop. But 850 nm is excluded from Telecom (Bellcore), so companies like Cielo and Picolight went to work on long wavelength parts. I worked with Cielo on 1310-nm VCSEL array technology while at Turin Networks, and actually succeeded in adding VCSEL transmitter and array receiver optics to several optical line cards. It was my hope that VCSELs would find their way into the fiber to the home (FTTH) systems of the future, delivering 1 Gbps or more for 33% of what it costs today.

NAN: Your most recent *Circuit Cellar* article, "Robot Navigation Position Determination from Acoustic Delay Triangulation" (260, 2012), focuses on using the mathematics of navigation to target a destination. What was the impetus for your investigation of this topic?

LARRY: I was just curious about how it might be done, so I sat down with paper and pencil and figured it out for myself. This is how I like to do things. One can always, or mostly, go to a book and find a solution. I prefer to try and derive my own solution. I find this approach gives me better understanding of the problem.

As I stated at the beginning of this interview, I was a sonar man in the Navy. One of my first PIC projects was to design and build my own sonar sensor. As time permits, I will eventually marry my sonar with the robot, given enough time.

NAN: Do you have any advice for *Circuit Cellar* readers who are considering building their own autonomous robots?

LARRY: Save the broken pieces from those beat up RC cars. Do your part to recycle!

NAN: What projects are you currently working on?

LARRY: I'm interested in software-defined radio. Right now, I'm looking at front-end design with analog switch-type mixers, driven by DDS local oscillator (LO). I'm using the frequency reference generator I wrote about in "Sweep-Frequency Generator Design: DDS-Based Test Equipment from Sweat Equity" (*Circuit Cellar* 254, 2011) for the LO.

NAN: Do you plan to write more articles for *Circuit Cellar*?

LARRY: I'd like to. It is my hope that I help to inspire young folks to love to learn and to be persistent in the pursuit of answers. That is where the reward lies.


THE NEXT
GENERATION OF
MAXSONAR®
The HRLV-MaxSonar Sensors
• Amazing One-Millimeter Resolution
• Simultaneous Multiple Sensor Operation
• Superior Rejection of Outside Noise Sources
• Target Size Compensation for Accuracy
• Temperature Compensation ($4.95)
• Outputs now include TTL Serial
$34.95 (MSRP)
MaxBotix
High Performance Ultrasonic Rangefinders
www.MaxBotix.com
facebook
THEY SAID
IT COULDN'T BE DONE
SO WE DID IT

by Kyle Gilpin (USA)

The menU Universal Menu System

The menU system is a universal menu library designed as a multipurpose user interface for embedded systems. It enables you to navigate through a hierarchical set of menu items while both observing and modifying the parameters of an embedded design.

The "menU" system—named because it is a *universal*, highly reconfigurable, dynamic menu system—is a multipurpose user interface for embedded systems. It enables the user to navigate through a hierarchical set of menu items while both observing and modifying parameters of the larger embedded design that the menU system belongs to. The system excels in several types of designs: space-constrained systems in which a standard keyboard and/or terminal emulator are not available, consumer products where one wishes to both restrict and guide the user, and minimalist hardware configurations in which the number of microprocessor pins available for a user interface is limited.

The impetus for the menU system was an embedded computer I developed for a 2003 Jetta. As I considered the number of engine parameters I wanted to monitor with the system, I realized there were many more of interest than would fit on the small liquid-crystal display (LCD) that I had integrated into the car's instrument cluster (see Photo 1). The menU system was developed to sort all of these parameters into categories I could easily navigate using a three-button interface and a four-line LCD. Since its initial application, I have rewritten the software twice: once as the "menbed" system, pictured in Photo 2—which won honorable mention in the *Circuit Cellar* NXP mbed design challenge—and again for this article. My initial thoughts on the design of the menU system were inspired by Aubrey Kagan's article, "Hierarchical Menus in Embedded Systems" (*Circuit Cellar* 160, 2003). While Kagan developed a solid, lightweight menu system, it was difficult to configure and maintain. It lacked the modularity of the menU system, did not support dynamic menu creation and deletion, and was not designed to be integrated as a component into other projects. The menU system aims to avoid these shortcomings.

Photo 1—The original inspiration for the menU system was a microprocessor-based sensor system that I designed and installed in my Jetta TDI. In particular, I needed a way to display and control a variety of different sensors using a small screen mounted between the gauges and the three buttons at the end of the windshield wiper stalk.

FEATURES & MENU NAVIGATION

The menU system is written in C, complies with the C99 standard, and can be targeted to any number of embedded or desktop processors. I have compiled the menU system for both Windows and Linux machines using the Qt development framework as a substitute for actual hardware. This enables a user to test the code and develop the menu hierarchy in an environment conducive to source-level debugging.

I have also compiled menU for both mbed and Atmel processors without any modification to the core code. When compiled for an Atmel ATmega1284P (128-KB flash memory, 16-KB RAM) using the AVR-GCC toolchain (with size optimization enabled), the core of the menU system consumes 10.4 kB (8%) of flash and 171 bytes (1%) of the chip's RAM. After adding a demo menu, flash consumption increases to 13%, and RAM usage also increases to 13%. On the mbed processor, the core menU code and the demo menu system together consume only 2% of flash

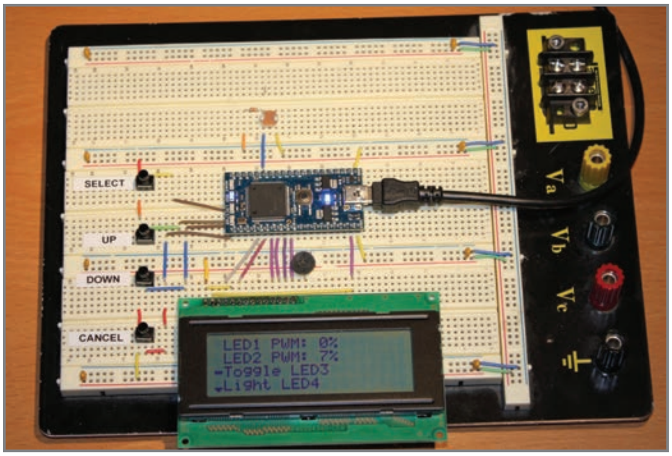


Photo 2—The menU system was tested on an NXP Semiconductors mbed module using the online mbed C++ compiler. Here four buttons are used to drive the system which uses a simple 4 × 20 parallel interface alphanumeric LCD. The four LEDs on the mbed board are used as outputs, and a single photocell provides an analog input.

and 2% of RAM. While the menU system may require some optimization for low-end microcontrollers, its memory footprint is small enough to fit in many common processors.

The menU system is highly configurable. Both the input method used to navigate the menU system and the interface used to display the menus are configurable at runtime. As a result, the user may choose to run the menU system using push-button switches and a simple LCD, a bidirectional serial connection to a terminal emulator program running on his computer, or any other combination of input and visualization interfaces. In particular, the menU system may be driven by as little as a single multi-modal button that reacts differently when held for varying amounts of time. To make navigation easier, the user may optionally employ up to three other buttons: Down, Up, and Cancel.

If the designer has configured the menU system to only use a single multipurpose push button, short momentary pushes of the button scroll through the menu items. If the button is depressed for a longer period, the currently selected menu item will be activated. When an item is activated, any associated function is executed, and either the item's child menu is displayed or its parameter is selected for editing. Parameters are the interface used by the menU system to display and modify variables in the embedded system at large. In Parameter Edit mode, the user can use the single button to modify the parameter's value within system-defined bounds. The user confirms a parameter's new value and exits Parameter Edit mode with another long hold of the button. Not all parameters may be edited; some parameters may be used solely to monitor system state (e.g., sensor values).

Up and Down buttons may be added to simplify the process of both scrolling through the menu items and incrementing and decrementing parameter values. With Up or Down buttons attached, the select button, if held for a long period of time, now cancels Parameter Edit mode discarding any changes the user has made. The designer may also opt to enable a Cancel button that serves to both exit Parameter Edit mode and navigate to the current menu's parent menu. Even without a Cancel button, the user may still navigate to the current menu's parent by ensuring that no menu item is selected and then pushing and holding the Select button.

SOFTWARE INFRASTRUCTURE REQUIREMENTS

To use the menU system, the system designer needs to supply three software components: a function indicating whether a given button is depressed, a display driver interface, and a regular clock tick. First, the function that indicates whether a particular button is currently depressed expects a button name and returns a Boolean. The buttons may be tangible push-button switches, or they may be emulated by any number of other interfaces commonly found on a microprocessor. For example, a UART may receive and translate key codes into virtual button presses. The designer informs the menU system of the button depressed button's function name at runtime by calling `menuButtons_setButtonDepressedFcn`.

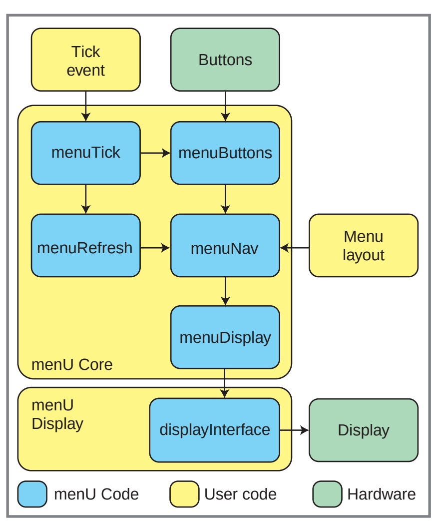


Figure 1—The core of the menU system has five modules. The menuTick module stimulates both the menuButtons and menuRefresh modules on a regular basis. These in turn drive the menuNav module, which manages the current state of the system. The menuDisplay module translates this state into commands that the displayInterface can understand and use to illustrate the state of the system on the attached display.

Second, the user must supply a display driver—an interface enabling the menU system to write characters to the attached display. The display driver must include a function that writes a character string to a line of the display and two functions that each expect a Boolean and display (or hide) an up or down arrow on the display. To inform the menU system of which display driver to use, the designer calls the `menuDisplay_setInterface`, which expects a pointer to a struct containing pointers to the aforementioned functions.

Finally, the user must supply a regular "tick" to keep the menu updated. The designer provides the tick to the menU system by arranging for the menuTick function to be called at regular intervals. Typically, this would be accomplished through the use of a generic timer/counter on the host microprocessor. The rate of ticks needs to be no faster than 10 ms, and may be slower if the user is concerned about

Listing 1—The process of creating a menu hierarchy is driven by three functions that create parameters, items, and menus. Parameters are encapsulated in items, and multiple items are grouped together in a menu. This example shows how to create a system that monitors the average, minimum, and maximum values of a generic analog sensor.

```
1.  menuParamHandle_t minParam = menuStructure_createParam(getMin, false, 0, 0, 0);
2.  menuItemHandle_t minItem = menuStructure_createItem("Min: %.2fV", minParam, resetStats, NULL);
3.  menuStructure_setSelectionIndex(minItem, 0);
4.  menuParamHandle_t maxParam = menuStructure_createParam(getMax, false, 0, 0, 0);
5.  menuItemHandle_t maxItem = menuStructure_createItem("Max: %.2fV", maxParam, resetStats, NULL);
6.  menuStructure_setSelectionIndex(maxItem, 1);
7.  menuItemHandle_t parentItem = menuStructure_createItem("Back", NULL, NULL, parentMenu);
8.  menuHandle_t minMaxMenu = menuStructure_createMenu(3, minItem, maxItem, parentItem);

9.  menuParamHandle_t avgParam = menuStructure_createParam(getAvg, false, 0, 0, 0);
10. menuItemHandle_t avgItem = menuStructure_createItem("Average: %.2fV", avgParam, NULL, minMaxMenu);
11. menuParamHandle_t windowParam = menuStructure_createParam(getSetWindow, true, 1, 100, 1);
12. menuItemHandle_t windowItem = menuStructure_createItem("Avg. Window: %.0f samples", windowParam,
    NULL, NULL);
13. menuHandle_t pcellMenu = menuStructure_createMenu(3, avgItem, windowItem, parentItem);
```

processing overhead. With each tick, the menU system scans for new input from the buttons and updates the display as necessary.

SOFTWARE DESIGN & HIERARCHY

The block diagram in Figure 1 illustrates the major software components within the menU system. The blue blocks are major software components with names that correspond to source file names. The system designer is responsible for supplying the yellow Tick event and Menu layout software blocks. The arrows in the diagram represent a generalized data flow between modules. The designer-supplied Tick Event drives the menuTick module which in turn activates the menuButtons and menuRefresh modules at regular intervals.

The menuButtons module watches the state of the physical buttons and records when they are pushed and released. From this data, the module generates messages indicating if a button has been pushed, released, or held for a significant amount of time. These messages are passed along to the menuNav module. Additionally, the menuButtons block implements a "typematic" function that automatically sends repeated button events at an increasing rate, the longer the user keeps a button continuously depressed.

The menuNav module is the heart of the menU system. This module uses the incoming button events from the menuButtons module to navigate the hierarchy of the user-defined menu layout. The module is responsible for calling any function associated with a menu item and updating any parameter attached to a menu item. Each time the menuNav module receives a button event it updates its state and sends a message to the menuDisplay module. The menuRefresh module reminds the menuNav module to continue sending messages to the menuDisplay at regular intervals even if the user is not pushing buttons. This regular display refresh ensures that the user can monitor changing parameters in real time.

The menuDisplay module parses messages from the menuNav module to determine what should be shown on the display. The menuDisplay module communicates with the user-specified displayInterface by specifying the exact text that should be displayed on each line of the physical display. The menuNav and menuDisplay modules employ a clever method to indicate when a menu item or item parameter should be highlighted: all text is limited to the standard ASCII characters (0–127), and the high-order bit of each character is used to indicate that the character should be highlighted. In general, the displayInterface translates the strings it receives from the menuDisplay module into display-specific signals that drive the display hardware. While the displayInterface is blue, the system designer will likely create his own interface to drive the particular display that he connects to the system.

MENU HIERARCHY SPECIFICATION

The menU system uses four different structs to describe the entire hierarchy of a menu system. The struct that forms the content of any menu is a menuItem. Each menuItem references a string containing the item's text and may optionally reference a `menuParam` struct (the second of the four key structs) that specifies a parameter that enables either the monitoring of internal system state (e.g., battery voltage) or the control of some system input (e.g., brightness). Each menu struct groups several menuItems into a coherent menu. To enable the reordering of menu items and to enable a single menuItem to be referenced by multiple menu structs, each menuItem is encapsulated by a menuLine struct.

The process of creating a menu hierarchy is shown in Listing 1. This process creates the hierarchy shown in Figure 2. In particular, Listing 1 creates a menu that can be used to monitor the voltage produced by an analog sensor, in this case, a photocell. The menu enables the user to observe an averaged value, control the number of samples used to compute the average, and monitor—as well as reset—the minimum and maximum measured values.

The designer starts by constructing the submenu that enables the user to monitor the minimum and maximum values. For a visual guide to this process, reference the right-most column of Figure 2. Line 1 of Listing 1 creates

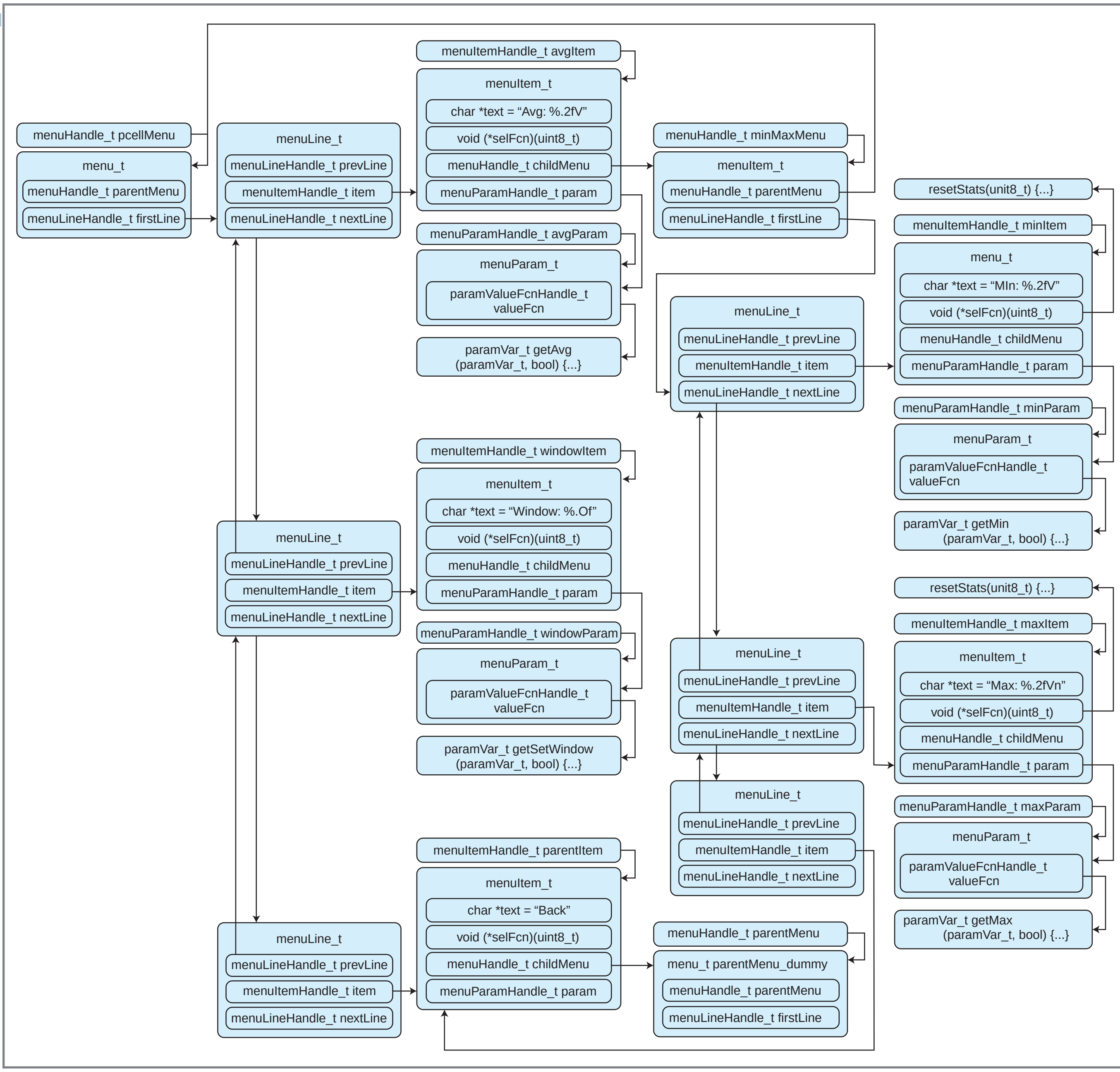


Figure 2—A menu hierarchy in the menU system is described by `menu_t` structs that each point to the first `menuLine_t` in the given menu. The `menuLine_t` structs form a doubly linked list, and each points to a `menuItem_t`. A `menuItem_t` struct may reference a `menuParam_t` if the item is used to display the state of some internal system variable. Additionally, each `menuItem_t` may point to another `menu_t` establishing a parent-child relationship between menus in the system.

the parameter that is used to hold the minimum measured value of the photocell. The first argument to `menuStructure_createParam` is a pointer to a designer-defined function that returns the minimum measured voltage. The function should be of the type `paramVar_t (*getSetFcn)(paramVar_t, bool)`. There is no need to set the minimum value with the menu system, so the designer only cares that the getMin function returns a `paramVar_t`. (A `paramVar_t` is a designer-configurable typedef for a fundamental type, by default, a float.) The getMin function is called every time the display is updated to get the most current minimum value. The second parameter, a Boolean called `liveUpdate`, specifies whether the `getSetFcn` is called continuously as the user modifies a parameter or only once after the user has confirmed that he wishes to change the parameter's value. Since the parameter is only being used to display the minimum photocell value, it is irrelevant. The next two arguments specify, in the case that the designer is creating a parameter that can be modified, the minimum and maximum allowable values, respectively. The last argument specifies the step size by which to increment or decrement the parameter. Setting this argument to 0 implicitly specifies that the parameter cannot be modified by the menU system.

Having created the minParam parameter, Line 2 of Listing 1 proceeds to incorporate it into a menuItem. The

Listing 2—menU parameters, which are used to display and modify the internal system state from within the menU system, operate by calling a Get/Set function like the one shown here that controls the number of samples in an averaging window. These Get/Set functions always return the most current parameter value, and they can also be used to set the parameter to a new value if the `update` argument is set to true.

```
1.   #include <stdbool.h> /* provides bool */
2.   #include <core/include/menuConfig.h> /* provides paramVar_t */

3.   /* window is the underlying variable I want to access with the
4.      menU system. I assign it an arbitrary initial value. */
     static paramVar_t window = 10;

5.   /* Each get/set function is just an abstraction to allow better
6.      data encapsulation by shielding the underlying variables
7.         that a menu parameter is accessing. */
8.   paramVar_t getSetWindow(paramVar_t newWindow, bool update) {
9.     if (update) {
10.     /* Only if the update parameter is true do I assign the
11.        newWindow value to the underlying static variable. This
12.           allows the caller to retrieve the current value of a
13.           parameter variable without modifying it. */
14.     window = newWindow;
15.    }
16.    /* Always return the newest value of the parameter variable
17.        to the caller. */
18.    return window;
19.  }
```

first argument passed to the `menuStructure_createItem` function is a string specifying the text of the item. It may contain a single printf-style format specifier. The value substituted for the format specified will be the value returned by the `getSetFcn` function associated with the parameter. The third argument is a pointer to a selection function that will be called when the menu item is selected. In this case, selecting the minItem will call that `resetStats` function that resets the minimum measured photocell value. The final argument is a `menuHandle_t` that references a child menu—in this case there is not one—that should be displayed when the item is selected.

Lines 4 and 5 are similar to Lines 1 and 2 in that they create a parameter and an item that together display the maximum measured photocell value. Note that both the minimum and maximum items call the same reset Stats function. The designer differentiates whether the minimum or maximum value should be reset by using a unique selection index (`selIndex`), set in Lines 3 and 6, for each item. When an item is selected, its designer-designated selIndex is passed to the item's `selFcn` as an argument. When the resetStats function is called with a 0, it knows to reset the minimum measured value, and when it is called with a 1, it knows to reset the maximum. Selection indexes are useful when forcing the user to choose one item from a menu containing many mutually exclusive options.

Line 7 is worth noting because it creates an item that has no associated parameter and no selection function, but it navigates to a submenu called `parentMenu`. The `parentMenu` is an always-present, system-defined menu handle that is always automatically translated to the current menu's parent menu. This approach avoids having to explicitly specify a menu's parent, and it enables the user to always return to the correct parent if a child menu is accessible from more than one parent.

Line 8 demonstrates the `menuStructure_createMenu` function that combines the minItem, maxItem, and parentItem into a cohesive menu. The first argument passed to the function is the number of menu items to be combined. It is followed by a variable length argument list of item handles specifying the items to be combined into the menu. When called, the `menuStructure_createMenu` function allocates a `menuLine_t` for each

Listing 3—The core/include/menuConfig.h file controls all aspects of the menu system behavior including the size of the display, the response of the buttons, and how memory is allocated for new menu items.

```
enum {
  menuConfigNUM_LINES = 4,
  menuConfigLINE_LENGTH = 20,
  menuConfigDOWN_BUTTON = true,
  menuConfigUP_BUTTON = true,
  menuConfigCANCEL_BUTTON = true,
  menuConfigNUM_BUTTONS = 1 + (menuConfigDOWN_BUTTON ? 1 : 0) +
    (menuConfigUP_BUTTON ? 1 : 0) + (menuConfigCANCEL_BUTTON ? 1 : 0),

  menuConfigDEBOUNCE_DELAY_MS = 45,
  menuConfigTYPEMATIC_PERIOD_MS = 333,
  menuConfigTYPEMATIC_X10_DELAY_MS = 2997,
  menuConfigLONG_RELEASE_DELAY_MS = 2500,

  menuConfigREFRESH_PERIOD_MS = 100,
  menuConfigSELECTION_WRAP = true,
  menuConfigSWAP_PARAMETER_INC_DEC = false,
  menuConfigDEFAULT_ESCAPABLE = true,
};

#define menuConfigNO_MALLOC false
enum {

  menuConfigTOTAL_NMENUS = 20,
  menuConfigTOTAL_NLINES = 60,
  menuConfigTOTAL_NITEMS = 50,
  menuConfigTOTAL_NITEMTEXTS = 20,
  menuConfigTOTAL_NPARAMS = 20
};

typedef float paramVar_t;
typedef int8_t lineIndex_t;
```

item that is being added to the menu. These lines, shown in the fourth column of Figure 2, form a double-linked list. This approach has two advantages: items within the menu can be reordered by swapping the order of the lines, and a single item can be included in multiple menus because it can be referenced by multiple lines.

Having created the `minMaxMenu` submenu, the Listing creates its parent menu, the members of which are shown in the third column of Figure 2. The first item in this parent menu is created by Lines 9 and 10 and displays the average photocell value. When selected, the `minMaxMenu` is displayed. The second item in the parent menu is created by Lines 11 and 12 and is used to control how many samples are used when calculating the average photocell value. The `menuStructure_createParam` call on Line 11 is different from those used to create the previous parameters because the resulting windowParam is used to both view and modify the averaging window size. The `getSetWindow` function not only returns a `paramVal_t`, but it also expects a `paramVal_t` that may be used to set the new size of the averaging window. To determine whether this underlying window size is actually updated, the function expects a second argument: a Boolean indicating whether the provided `paramVal_t` argument should actually be used to update the underlying variable or simply ignored. Listing 2 shows what the `getSetWindow` function looks like. When the user is not actively modifying the window size, but the menU system needs to display the current window size as part of the text associated with the windowItem, it passes false as the second argument to the `getSetWindow` function so that it will ignore the first parameter while still returning the current value of the window size. When the user has selected the windowItem and the menU system is in Parameter Edit mode, each time the user increments or decrements the parameter, the `getSetWindow` function will be called with true on account of the fact that the liveUpdate argument to the `menu-Structure_createParam` call on Line 11 of Listing 1 was set to true. Line 11 also shows that the minimum window size is 1, the maximum is 100, and the user can choose any integer window size between (and including) these two extremes.

Finally, Line 13 combines the `avgItem`, `windowItem`, and previously mentioned `parentItem` into the `pcellMenu` shown in the first column of Figure 2. While this example is relatively small, it demonstrates most major features of the menU system. One useful feature that is conspicuously absent is the ability to create and destroy menu items and menus at run time. For an example of this, refer to the demo code on *Circuit Cellar*'s FTP site.

MENU SYSTEM OPTIONS

The menU system has numerous compile-time options that are all defined in the "core/menuConfig.h" file. Listing 3 shows a typical configuration file with comments removed due to space constraints. Many of the configuration options are defined as enums instead of macros in order to better utilize the compiler's error checking abilities. In explaining the configuration options, I will omit the menuConfig prefix that precedes the more descriptive option name in order to save space.

The first two enum values, `NUM_LINES` and `LINE_LENGTH`, are intuitive—they specify the number of lines in the chosen display and the maximum number of characters that can be printed per line, respectively. The next three options, `DOWN_BUTTON`, `UP_BUTTON`, and `CANCEL_BUTTON`, are Booleans that specify whether each of the given buttons is connected to the system. They are followed by `NUM_BUTTONS` which the designer should not edit manually.

The next set of four parameters defines the response of the buttons. The DEBOUNCE_DELAY_MS specifies the amount of time, in milliseconds, that a button must be continuously depressed before the system registers it. The TYPEMATIC_DELAY_MS specifies the number of milliseconds a button must be depressed before the button handler automatically generates another keypress. The TYPEMATIC_X10_DELAY_MS specifies the length of time a button must be held down before the button handler starts automatically generating key presses at a rate that is 10 times faster than the initial rate specified by the TYPEMATIC_PERIOD_MS. Finally, the LONG_RELEASE_DELAY_MS specifies the amount of time a button must be held down before it is considered a "long" push that may result in the menuNav module performing a different action than it would have had it received a "short" push.

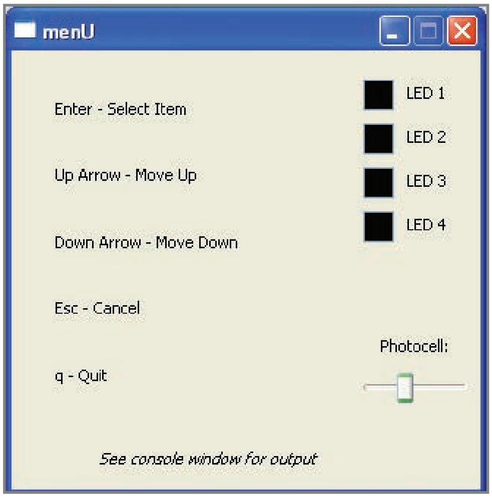


Photo 3—The menU system is generic enough to be compiled for most desktop PCs running Windows, OSX, or Linux using the Qt development framework. This screenshot demonstrates the GUI for the menU system. The menu itself is displayed in a separate terminal window. The GUI has four simulated LEDs and one simulated photocell all of which correspond to the hardware available on the mbed processor development platform shown in Photo 2 and Figure 3.

The final three enum values specify some minor menU options and are not closely related. REFRESH_PERIOD_MS specifies to the menuRefresh module how long to wait between refreshes of the display if the user has not pushed or released a button. The SELECTION_WRAP option is a Boolean that controls whether, upon reaching the last item in a menu, the selected line is able to wrap around to the top, and vice versa. For configurations lacking both up and down buttons, wrapping is always enabled. The SWAP_PARAMETER_INC_DEC option is another Boolean and specifies whether the functionality of the up and down buttons should be swapped when editing a parameter. Finally, the DEFAULT_ESCAPABLE option determines whether the user can navigate to the active menu's parent menu by default. One can change whether an individual menu is "escapable" by calling the menuStructure_setEscapable function. This is useful in cases where the system designer wishes to force the user to choose one of several options in a menu.

The next set of configuration options controls the menU system's memory usage. In particular, the NO_MALLOC option, when set to false, enables the menU system to use C's malloc function to allocate space for each menU system struct on the heap. Alternatively, if NO_MALLOC is set to true, the system preallocates space for a set number of menU-related structs in the static data section. This can be valuable in lightweight systems with limited RAM. If memory is statically allocated, new items are assigned to one of the unused items from the preallocated cache. The TOTAL_NMENUS, TOTAL_NLINES, TOTAL_NITEMS, TOTAL_NITEMTEXTS, and TOTAL_NPARAMS enum values specify the number of menU-related structs to preallocate.

The final two lines of the configuration file specify typedefs that enable added flexibility in the menU system. The paramVar_t typedef specifies the type that is used to convey parameter values within the menU system. By default, it is a float, but it may be changed to an int if the system designer does not want the additional overhead associated with floats. The printf-style text field of each menu item, if it is to display a parameter value, must contain a format specifier compatible with the paramVar_t, (e.g., %f for a float or %d for an int). Finally, the lineIndex_t typedef specifies the integer type that should be used when referring to lines of the display. It should be a signed type as –1 is used to reference the last line of the display. There is only a need to change it to something other than int8_t if one needs more than 127 lines in a single menu.

TRY IT YOURSELF

It is easy to try the menU system on your desktop computer. To facilitate this process, I created the Qt-based

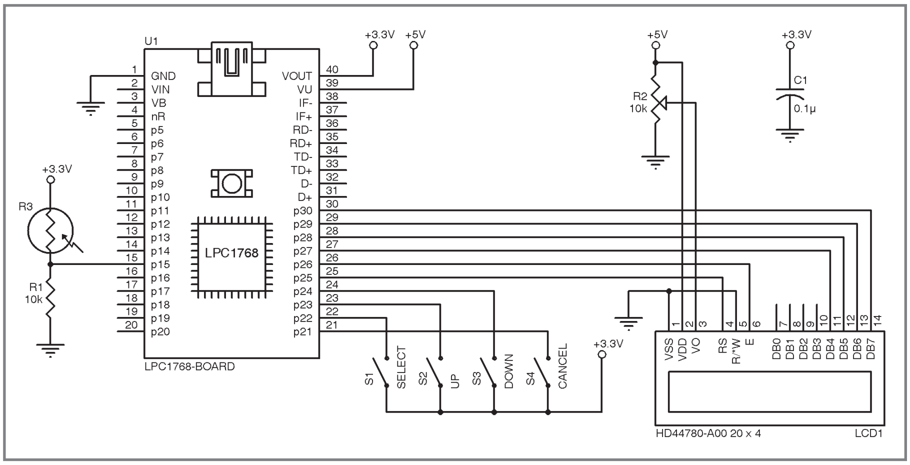


Figure 3—The design used to test the menU system on the mbed processor was meant to be as simple as possible. Four buttons drive the menu system and an alphanumeric LCD is used to display the menu. Alternatively, one can use the mbed's USB-to-serial port to connect with a terminal emulator running on a PC to both display and control the menu system.

graphical user interface (GUI) shown in Photo 3. Qt is a cross-platform application development framework written in C++ that provides both standard GUI elements and high-level language features. The Qt GUI for the menU system simulates hardware button presses, four LEDs, and an analog input. The display for the Qt GUI uses the terminal from which the GUI program was executed. In particular, the demo program uses one of the "curses" libraries to display menu items in the terminal. This approach, in contrast to using the GUI to display the menu, is preferable because it better mimics how the items are likely to be displayed in an embedded context. It is worth noting that the keyboard input must come from the GUI because only the GUI can detect both key press and release events.

To experiment with the menU system on your Windows-, Linux-, or OS-X-based PC, download the Qt SDK (tested with version 1.1.2) and install the default configuration. Once the installation is complete, run Qt Creator and choose Open File or Project from the File menu. Navigate to the Qt subdirectory of the archive containing the source code for this article. Select the menU_QT.pro file and click the Open button. In the Project Setup window that appears, ensure that only the Desktop checkbox is selected, and click the Finish button. Now, on the left-hand side of the main Qt Creator window, select the Projects icon directly above the Help icon (or press Ctrl-5). In the Targets tab, click on Build under the Desktop configuration to display the Build Settings pane. Ensure that the Shadow Build checkbox is unchecked. (As a result, the application will be built in the Qt/debug directory, which is prepopulated with the pdcurses.dll for Windows targets.) Now, click Run under the Desktop configuration to display the Run Settings pane. Check the Run in Terminal checkbox.

To execute the menU demo application, select Run from the Build menu and wait for the project to compile. When complete, two windows should appear: a terminal displaying the menu system and the GUI pictured in Photo 3 showing four virtual LEDs and a slider to control the simulated photocell voltage. Ensure that the GUI window, not the terminal, is active when navigating the menu.

Of course you can also try the demo menU application on the mbed platform. Figure 3 shows the schematic diagram detailing how to connect four push-button switches, a photocell, and a 4 × 20 line alphanumeric LCD to the mbed processor. The mbed subdirectory of the archive of the source code for this article contains the menU_mbed.zip file, which can be imported as a program to the online mbed compiler. Instead of using the LCD to display the menu system, you can select the vt100Display interface. This will redirect all output over the mbed's serial-to-USB interface to a terminal emulator (115200,8,N,1), such as SecureCRT running on your desktop. You can also experiment with using the terminal emulator for input (in place of the push buttons). To do so, consult the hwInterface.c file in the menU_mbed.zip archive that defines which keys correspond to which buttons.

FUTURE WORK

In the process of developing the menU software, I discovered a few features that I will continue working to implement. I would like the ability to include multiple parameters in a single menu item so that I can more easily display variables, such as time, that are represented by multiple fields. I am also pursuing the idea of temporarily diverting control of the display—especially if it is a graphical one—to some other process running on the user's embedded system so that it could show plots or other graphical information that do not fit in the current menU framework.

I encourage you to integrate the menU system in your next project. If you create bug fixes or your own improvements, I will make an effort to include them in a future release of the menU source code. I hope that the menU system will simplify the process of designing user interfaces so that embedded system designers can shorten their development times while focusing on the challenges unique to particular projects.

Kyle Gilpin has been reading Circuit Cellar *since he was 13 years old. He is currently finishing his PhD in EE/CS at MIT's Distributed Robotics Lab. Kyle is also the founder of HighZ Design, a custom hardware, software, and mechanical prototype design firm. You can contact him at kwgilpin@highzdesign.com.*

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/262.

RESOURCES

A. Kagan, "Hierarchical Menus in Embedded Systems," *Circuit Cellar* 160, 2003.

mbed, http://mbed.org.

Nokia Corp., "Download Qt, the Cross-Platform Application Framework," http://qt.nokia.com/downloads.

SourceForge, Public Domain Curses, 2008, http://pdcurses.sourceforge.net.

VanDyke Software, Inc., SecureCRT, www.vandyke.com/products/securecrt.

WebRing, Inc., AVR, http://winavr.sourceforge.net/index.html.

SOURCES

ATmega1284P Microcontroller
Atmel Corp. | www.atmel.com

mbed
NXP Semiconductors | www.nxp.com

Elektor Shop

The world of electronics at your fingertips!

Signal Generator + Universal Receiver + Active Antenna

AVR Software Defined Radio

Atmel AVR microcontrollers are very popular, not least because of the free development tools that are available. In Elektor magazine we show how these processors can be pressed into service for digital signal processing tasks. 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. The software for all these projects is written using the WinAVR GCC compiler in AVR Studio and can be downloaded as C source code (plus fuse settings) or as hex files.

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

Art.# 100182-72 • $133.00

Kits & Modules

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

Improved Radiation Meter

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

Kit of parts incl. display and programmed controller

Art.# 110538-71 • $57.30

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

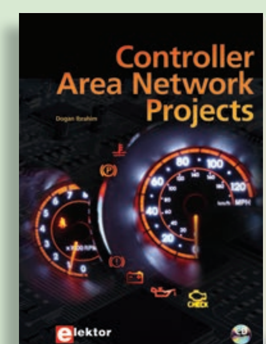
Books

LabWorX: Straight from the Lab to your Brain

Mastering the I²C Bus

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

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

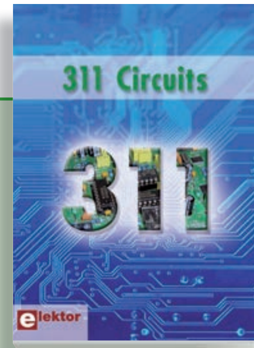


Free mikroC compiler CD-ROM included

Controller Area Network Projects

The aim of the book is to teach you the basic principles of CAN networks and in addition the development of microcontroller based projects using the CAN bus. You will learn how to design microcontroller based CAN bus nodes, build a CAN bus, develop high-level programs, and then exchange data in real-time over the bus. You will also learn how to build microcontroller hardware and interface it to LEDs, LCDs, and A/D converters. This book comes complete with a free CD-ROM.

260 pages • ISBN 978-1-907920-04-2 • $47.60



Creative solutions for all areas of electronics

311 Circuits

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

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

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

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

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

CD/DVD-ROMs

by Richard Wotiz (USA)

Camera Image Stabilization

Image stabilization (IS) helps reduce blurriness that results from slight camera movements when shooting without a tripod. This article details several IS techniques as well as some cameras featuring different types of IS systems.

Digital cameras are all around us, from the first-generation models that have been around since the mid-1990s to the cameras built into modern smartphones. I like to photograph nature scenes, sometimes in the forest under a heavy tree canopy that can be quite dark even on a bright sunny day. When shooting at the slow shutter speeds required by a dimly lit scene, image stabilization (IS) helps to reduce blurriness that comes from slight camera movements when shooting without a tripod. IS has been around since the mid-1990s in high-end lenses for film cameras. It first appeared in compact point-and-shoot digital cameras around 2003, and is now available in many recent models.[1]

This month I'll take you through a tour of various IS techniques, then we'll peek inside a couple of cameras that include different types of IS systems.

A common rule of thumb is that a 35-mm film camera that's held by hand can, on average, take a blur-free image at a shutter speed no slower than 1/FL, where FL is the focal length of the lens. In a digital camera with a smaller image sensor, the focal length is first scaled up to its equivalent 35-mm value. A camera's specification sheet will often list its 35-mm equivalent FL. The ones I'll be looking at have charge-coupled device (CCD) sensors that are approximately one sixth the size of a 35-mm film frame. This means the FL multiplier, or crop factor, would be six. So, a zoom lens that's currently set to a 20-mm FL would need to be scaled up to 120 mm, giving a minimum safe exposure time of 1/120 s. You can typically gain two to four extra exposure steps with IS. That would enable you to shoot as slow as 1/30 s, or possibly even 1/8 s, and still have a reasonably blur-free image.

STEADYING THE IMAGE

The methods used for steadying the image from a handheld camera fall into two broad categories. First, you can steady the entire camera. There are mounting systems that do this with purely mechanical means. Some operate on a similar principle to the circus tightrope performer who holds a long pole for balance. The camera and its battery pack are placed on opposite ends of a pole. This increases the moment of inertia, making the camera more stable. A complex arrangement of pulleys and springs isolates the pole from the photographer's movements. The whole system basically acts like a mechanical low-pass filter. It typically requires careful setup and specialized training, and is frequently used for action shots on movie sets. There are also gyro-based mounting systems. They aren't as difficult to set up or use, though they require a source of power and can be quite heavy.

The other IS method works inside the camera by making small mechanical adjustments to the optical path. It uses an accelerometer or gyro sensor to detect slight vibrations and drives an actuator to compensate for the movement. When taking pictures of a distant subject, even the slightest rotation of the camera is greatly amplified, which a gyro sensor can detect. For close-ups and

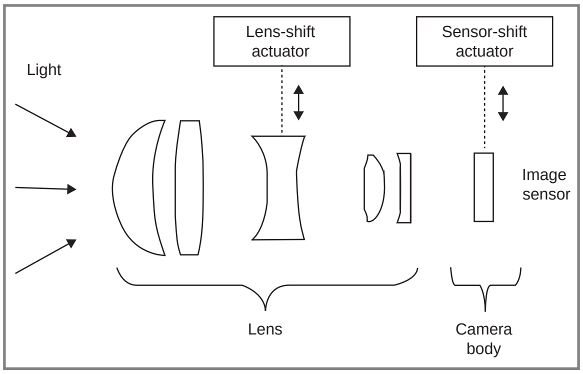


Figure 1—The basics of an image-stabilized camera. Only one of the two actuators shown will be present, depending on the mechanism type. The actuator moves either the sensor or part of the lens in both the horizontal and vertical direction, perpendicular to the direction of the incoming light.

macro photography, linear camera movement becomes more significant, so an accelerometer would be best. Most compact cameras available today use only a gyro sensor, though a few higher-end models have both sensor types. The mechanism can either be located inside the camera body or it can be part of the lens. Figure 1 shows both of these internal IS systems, which I'll be taking a closer look at. (See the Resources section at the end of this article for more information on various types of IS.)

In addition to these methods, some cameras use various software techniques to reduce blur. This can be as simple as increasing the sensor's sensitivity, which also increases the image's background noise, or it could be a more sophisticated image-processing algorithm. These are not true IS, though they are often referred to as "digital image stabilization," or some related term that is unfortunately often designed to confuse rather than inform.

When I first came up with the idea for this article, the only digital camera I had ever owned was one I bought about 10 years ago. It weighs in at just under 1 lb, including the modified camcorder battery pack I mounted to the outside so I could get a reasonable number of shots on a recharge. Now seemed like a good time to find something more recent. I picked up a pair of inexpensive used cameras with IS, knowing they might be sacrificed for a good cause. I had never taken a camera apart before, so I first armed myself with a new jeweler's screwdriver set and a large magnifying glass, just in case.

INSIDE THE LENS

The first camera I opened up uses lens-based IS, often referred to as optical IS. It uses a pair of tiny electromagnetic positioners to shift one of the lens elements in either direction perpendicular to the optical axis. Photo 1a shows the mechanism in one compact package. The entire assembly fits inside the center portion of the lens, with the flex cables enabling it to move as the lens zooms or retracts. The round metal objects near the bottom are the shutter and aperture actuators. This camera model doesn't have a true variable aperture. Instead, it has a neutral-density filter that can swing in front of the shutter to provide a second aperture setting. The underside of the IS mechanism is hidden behind the orange flex circuit layers. You can see it disassembled in Photo 1b. The two magnets attached to the floating-lens carrier (bottom of photo) get sandwiched between the drive coils and the plastic support frame. The camera's microcontroller supplies a pulse-width modulation (PWM) signal to the coils to shift the lens in a horizontal plane to stabilize the image. Hall-effect sensors under each magnet provide position feedback. At the top of the photo are two of the smallest ball bearings I have ever seen. When they fell out of the lens carrier (along with some equally tiny springs), I realized there was little chance that I would ever reassemble these pieces into a working camera. I think that was the mechanical equivalent of letting out the magic smoke!

The camera was packed so tightly that I couldn't get to any of the IS wiring without disassembling the lens, but I still wanted to see the mechanism in action. I found that the drive coils needed about 50 mA to reach their limit of travel. I tried feeding one of them with a sine wave, and I could get reliable end-to-end movement up to 65 Hz. Most people's hands will shake with small vibrations in the range of 10 Hz to 30 Hz, which this mechanism can easily accommodate.

SENSOR SHIFT

The second camera I looked at has a sensor-shift mechanism, which is sometimes called mechanical IS. Since this is located entirely outside of the lens, I found it much easier to examine. Photo 2 shows the camera with its

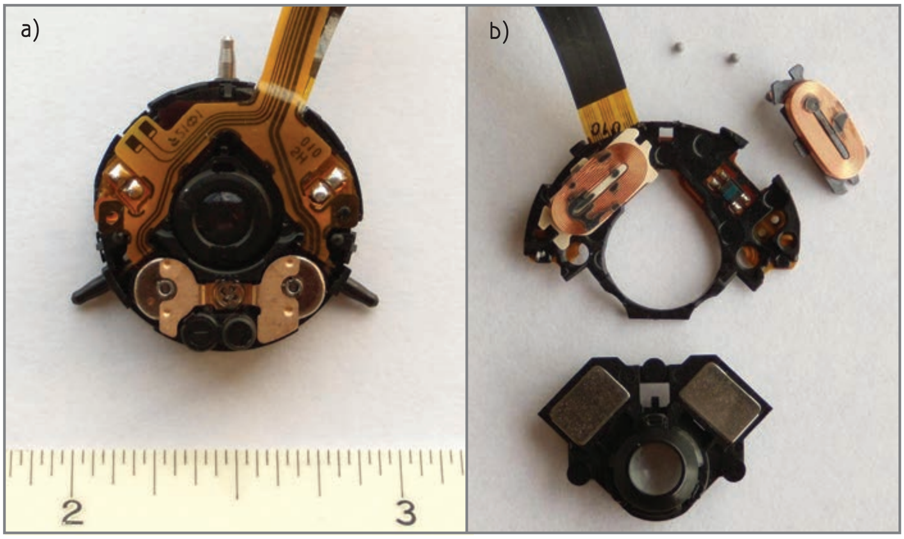


Photo 1a—A lens-based IS mechanism that sits inside the lens barrel. This is the back side of the actuator assembly. b—This is a front view of the actuator components. The lens and magnet assembly (bottom) float on ball bearings between the drive coils and the plastic frame. A Hall-effect position sensor is visible below the former location of one of the coils.

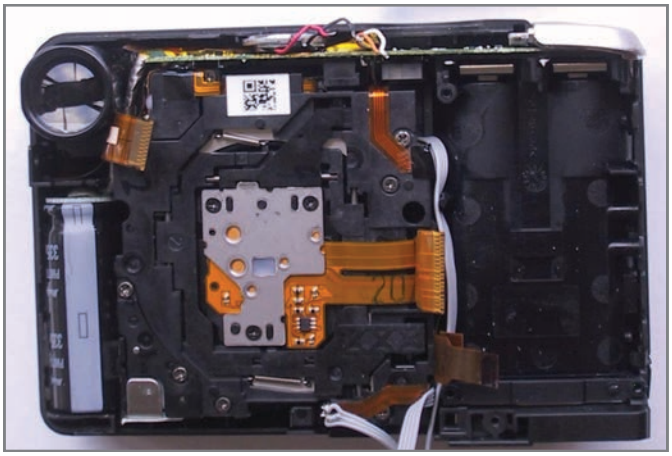


Photo 2—Rear view of the sensor-shift IS mechanism. The image sensor (under the metal plate) slides horizontally and vertically on tiny metal shafts. You can see the gray wires I connected to the IS components to monitor their operation.

Photo 3—Top view of the horizontal sensor-shift actuator. At the top left corner is one of the smallest stepper motors I have ever come across, only 0.3" long. It has an internal leadscrew to move its shaft horizontally, which presses against a carriage to move the image sensor. The larger motor to its right controls zooming and lens retraction.

back, liquid crystal display (LCD), and circuit board removed. The metal plate visible at the center covers the back of the sensor. If you look closely just above the plate, you'll see the two small exposed ends of a shiny horizontal metal shaft. The sensor assembly slides back and forth on this shaft. Vertical movement occurs on a pair of shafts on either side of the sensor, but they're darker colored and harder to see. I removed the lens assembly so I could get a closer look at the actuators and their wiring. Photo 3 is a front view of the horizontal actuator tucked in the corner next to the larger zoom motor. I connected wires to the actuators and the nearby position sensors and ran them down to the bottom of the camera. Then I put everything back together and was pleasantly surprised to find that the camera still worked.

I turned on the camera while looking at the various signals I had just gained access to and discovered that the actuators were actually stepper motors and the sensors were simply limit switches. On power-up, the camera drives each stepper to the end of its travel (until the limit switch trips) then back to center, one at a time (see Photo 4). This process takes about 1 s, and gave me a convenient pattern that I used to sort out the different signals. I found that the steppers are bipolar, with each one driven by a pair of H-bridges. Photo 5 shows the drive signals in more detail. I recognized the waveform as a half-step drive sequence, with a rate of one full step every 2.2 ms. (You can read more about stepper motor waveforms in Miguel Sánchez's article "Three-Axis Stepper Controller," *Circuit Cellar* 234, 2010.)

It's difficult to make much sense of these waveforms as they relate to sensor position. I transferred the raw data to my PC and formatted the two sets of four phase bits at each time interval into a pair of 4-bit numeric values. Then I loaded the numbers into a spreadsheet and converted these phase signals into actual position data. Figure 2a shows this data in a more readable format. You can see the horizontal stepper go to its limit of 180 half-steps then back to center. The vertical stepper follows a slightly different pattern that I found to be dependent on the camera's orientation. Here, it goes up to 204 before settling at 154. It goes back down to 0 when the camera is powered off. You can download the spreadsheet, as well as others showing various position waveforms, from *Circuit Cellar*'s FTP site.

Figure 2b shows a 1-s exposure while I'm holding the camera in front of me. There are slight horizontal and vertical

Photo 4—The horizontal (X) and vertical (Y) steppers go through their initialization sequence at power-up. The A and B signal name suffixes are the positive and negative terminals of each phase winding, respectively.

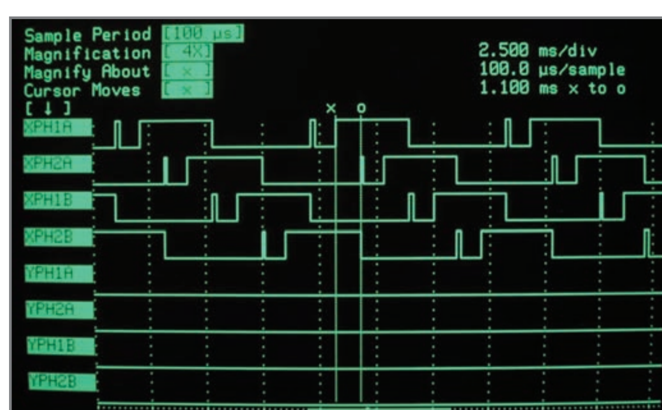


Photo 5—A closer look at the waveform of Photo 4. You can clearly see the half-step pattern. The narrow pulses are noise from the inductive spikes when the opposite ends of each winding are driven low.

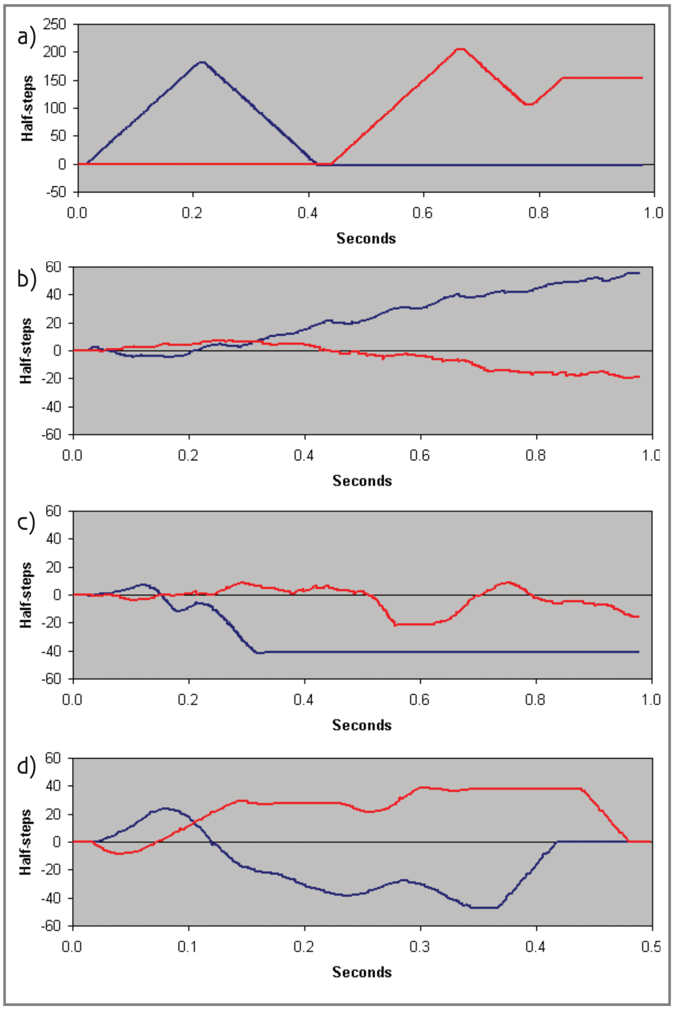


Figure 2—Each of these graphs shows the stepper phase waveform data converted into a set of position values. The horizontal position is shown in blue, and the vertical position in red. a—The initialization sequence from Photo 4. b—A 1-s exposure while I held the camera with one hand. c—Another 1-s exposure, while I was slowly panning horizontally. d—This 1/3-s exposure shows both steppers returning to their standby positions afterward.

oscillations at about 10 Hz, which is typical of the slight movement of muscles in the hand. The slow drift is sometimes steady, as shown here. At other times it appears more random. I had to manually set a long 1-s exposure time because the IS system is only active while the shutter is open. Other cameras enable you to set different modes where IS can be active either continuously or when the shutter button is pressed halfway, but that would have a significant effect on battery life. I measured the current consumption during the exposure at 280 mA without IS, increasing to 550 mA with IS turned on. Idle current was 380 mA with the LCD backlight at full brightness or 340 mA when dimmed. The steppers also make a slightly audible ticking sound while they're running, so I could imagine it might get annoying if they were enabled all the time. I never noticed that with the lens-based actuator in the other camera.

The IS system automatically detects panning when it sees steady movement which causes it to freeze the appropriate stepper at its current position. In Figure 2c, the horizontal stepper stops moving after about 0.1 s of constant travel. This seems like a convenient feature, compared to some other cameras that require you to manually set their IS function to panning mode to disable horizontal shifting. After each shot, the steppers move back to their standby position, which you can see at the end of the 1/3-s exposure in Figure 2d. I found that the IS performance varied with the camera's zoom setting. All of the previous shots were done at maximum zoom. With a wide exposure, the system was much less sensitive to movement, and the maximum step rate slowed down to 2.8 ms per full step. That makes sense, since the farther you're zoomed in, the greater the effect a given amount of camera rotation will have on the image.

SMILE

I learned a great deal about compact cameras while writing this column, and I hope you've learned some while reading it, too. I put aside my 10-year-old camera for a more modern replacement, though for now it still has a few wires sticking out of it. I expect to get more use out of it since it's much smaller and lighter than the old one, and actually fits in a pocket. I'm also used to the larger LCD. Going back to my old camera's smaller display almost feels like I'm watching an antique TV with its tiny screen in the center of a large console. I think it's finally time to let that image fade into the past. ■

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

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/262.

REFERENCE

[1] Digital Photography Review, www.dpreview.com.

RESOURCES

T. Harris, "How Steadicams Work," HowStuffWorks, Inc., 2012, http://entertainment.howstuffworks.com/steadicam.htm.

Kenyon Laboratories LLC, Gyro Stabilizers, www.ken-lab.com/stabilizers.html.

D. Sachs, S. Nasiri, and D. Goehl, "Image Stabilization Technology Overview," InvenSense, Inc., www.invensense.com/mems/gyro/documents/whitepapers/ImageStabilizationWhitepaper_051606.pdf.

M. Sanchez, "Three-Axis Stepper Controller," *Circuit Cellar* 234, 2010.

by Patrick Schaumont (USA)

"Embedded Security" is a new column on the topics of designing secure embedded systems, cryptographic standards, and hardware/software protection. It will appear every other month.

One-Time Passwords from Your Watch

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

Passwords establish the identity of a user, and they are an essential component of modern information technology. In this article, I describe one-time passwords: passwords that you use once and then never again. Because they're used only once, you don't have to remember them. I describe how to implement one-time passwords with a Texas Instruments (TI) eZ430-Chronos wireless development tool in a watch and how to use them to log in to existing web services such as Google Gmail (see Photo 1).

To help me get around on the Internet, I use a list of about 80 passwords (at the latest count). Almost any online service I use requires a password: reading e-mail, banking, shopping, checking reservations, and so on. Many of these Internet-based services have Draconian password rules. For example, some sites require a password of at least eight characters with at least two capitals or numbers and two punctuation characters. The sheer number of passwords, and their complexity, makes it impossible to remember all of them.

What are the alternatives? There are three different ways of verifying the identity of a remote user. The most prevailing one, the password, tests something that a user knows. A second method tests something that the user has, such as a secure token. Finally, we can make use of biometrics, testing a unique user property, such as a fingerprint or an eye iris pattern.

Each of these three methods comes with advantages and disadvantages. The first method (passwords) is inexpensive, but it relies on the user's memory. The second method (secure token) replaces the password with a small amount of embedded hardware. To help the user to log on, the token provides a unique code. Since it's possible for a secure token to get lost, it

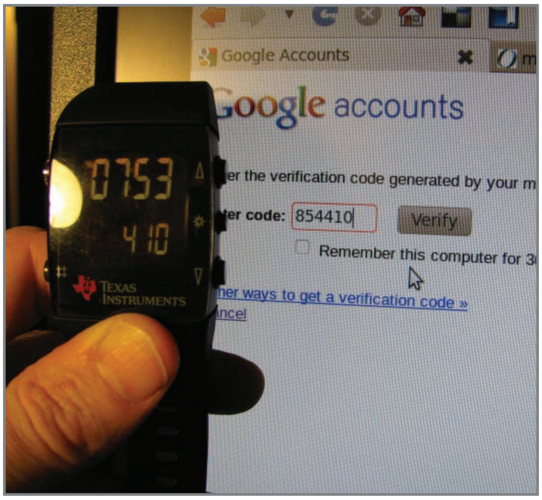


Photo 1—The Texas Instruments eZ430 Chronos watch displays a unique code that enables logging into Google Gmail. The code is derived from the current time and a secret value embedded in the watch.

must be possible to revoke the token. The third method (biometrics) requires the user to enroll a biometric, such as a fingerprint. Upon login, the user's fingerprint is measured again and tested against the enrolled fingerprint. The enrollment has potential privacy issues. And, unlike a secure token, it's not possible to revoke something that is biometric.

The one-time password design in this article belongs to the second category. A compelling motivation for this choice is that a standard, open proposal for one-time passwords is available. The Initiative for Open Authentication (OATH) is an industry consortium that works on a universal authentication mechanism for Internet users. They have developed several proposals for user authentication methods, and they have submitted these to the Internet Engineering Task Force (IETF). I'll be relying on these proposals to demonstrate one-time passwords using a eZ430-Chronos watch. The eZ430-Chronos watch, which I'll be using as a secure token, is a wearable embedded development platform with a 16-bit Texas Instruments MSP-430 microcontroller.

ONE-TIME PASSWORD LOGON

Figure 1 demonstrates how one-time passwords work. Let's assume a user—let's call him Frank—is about to log on to a server. Frank will generate a one-time password using two pieces of information: a secret value unique to Frank and a counter value that increments after each authentication. The secret, as well as the counter, is stored in a secure token. To transform the counter and the secret into a one-time password, a cryptographic hash algorithm is used. Meanwhile, the server will generate the one-time password it is expecting to see from Frank. The server has a user table that keeps track of Frank's secret and his counter value. When both the server and Frank obtain the same output, the server will authenticate Frank. Because Frank will use each password only once, it's not a problem if an attacker intercepts the communication between Frank and the server.

After each logon attempt, Frank will update his copy of the counter in the secure token. The server, however, will only update Frank's counter in the user table when the logon was successful. This will intercept false logon attempts. Of course, it is possible that Frank's counter value in the secure token gets out of sync with Frank's counter value in the server. To adjust for that possibility, the server will use a synchronization algorithm. The server will attempt a window of counter values before rejecting Frank's logon. The window chosen should be small (i.e., five). It should only cover for the occasional failed logon performed by Frank. As an alternate mechanism to counter synchronization, Frank could also send the value of his counter directly to the server. This is safe because of the properties of a cryptographic hash: the secret value cannot be computed from the one-time password, even if one knows the counter value.

You see that, similar to the classic password, the one-time password scheme still relies on a shared secret between Frank and the server. However, the shared secret is not communicated directly from the user to the server, it is only tested indirectly through the use of a cryptographic hash. The security of a one-time password therefore stands or falls with the security of the cryptographic hash, so it's worthwhile to look further into this operation.

CRYPTOGRAPHIC HASH

A cryptographic hash is a one-way function that calculates a fixed-length output, called the digest, from an arbitrary-length input, called the message. The one-way property means that, given the message, it's easy to calculate the digest. But, given the digest, one cannot find back the message.

The one-way property of a good cryptographic hash implies that no information is leaked from the message into the digest. For example, a small change in the input message may cause a large and seemingly random change in the digest. For the one-time password system, this property is important. It ensures that each one-time password will look very different from one authentication to the next.

The one-time password algorithm makes use of the SHA-1 cryptographic hash algorithm. This algorithm produces a digest of 160 bits. By today's Internet standards, SHA-1 is considered old. It was developed by Ronald L. Rivest and published as a standard in 1995.

Is SHA-1 still adequate to create one-time passwords? Let's consider the problem that an attacker must solve to break the one-time password system. Assume an attacker knows the SHA-1 digest of Frank's last logon attempt. The attacker could now try to find a message that matches the observed digest. Indeed, knowing the message implies knowing a value of Frank's secret and the counter. Such an attack is called a pre-image attack.

Fortunately, for SHA-1, there are no known (published) pre-image attacks that are more efficient

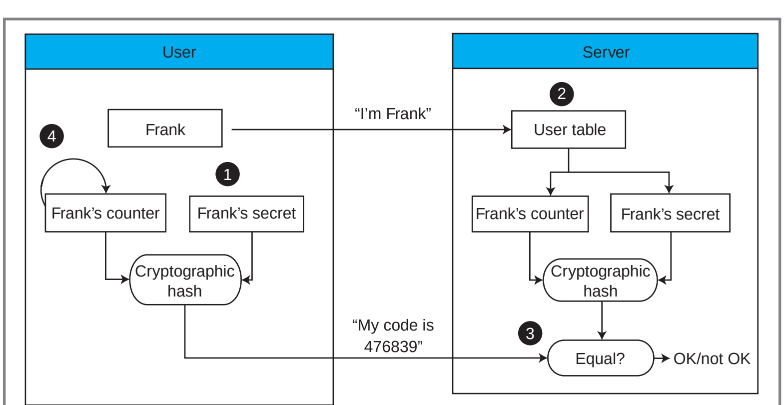


Figure 1—A one-time password is formed by passing the value of a personal secret and a counter through a cryptographic hash (1). The server obtains Frank's secret and counter value from a user table and generates the same one-time password (2). The two passwords must match to authenticate Frank (3). After each authentication, Frank's counter is incremented, ensuring a different password the next time (4).

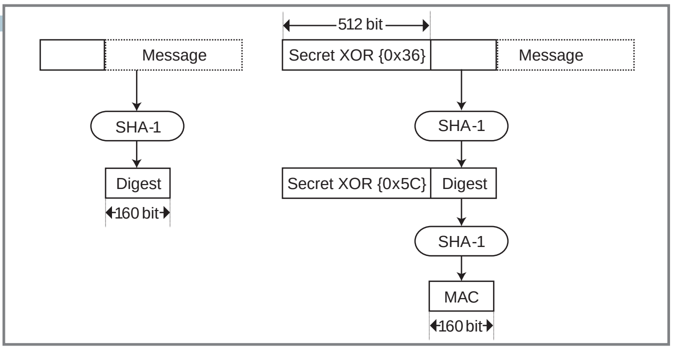


Figure 2—The SHA-1 algorithm on the left is a one-way function that transforms an arbitrary-length message into a 160-bit fixed digest. The Hash-based message authentication code (HMAC) on the right uses SHA-1 to combine a secret value with an arbitrary-length message to produce a 160-bit message authentication code (MAC).

than brute force trying all possible messages. It's easy to see that this requires an astronomical number of messages values. For a 160-bit digest, the attacker can expect to test on the order of 2^{160} messages. Therefore it's reasonable to conclude that SHA-1 is adequate for the one-time password algorithm. Note, however, that this does not imply that SHA-1 is adequate for any application. In another attack model, cryptographers worry about collisions, the possibility of an attacker finding a pair of messages that generate the same digest. For such attacks on SHA-1, significant progress has been made in recent years.

The one-time password scheme in Figure 1 combines two inputs into a single digest: a secret key and a counter value. To combine a static, secret key with a variable message, cryptographers use a keyed hash. The digest of a keyed hash is called a message authentication code (MAC). It can be used to verify the identity of the message sender.

Figure 2 shows how SHA-1 is used in a hash-based message authentication code (HMAC) construction. SHA-1 is applied twice. The first SHA-1 input is a combination of the secret key and the input message. The resulting digest is combined again with the secret key, and SHA-1 is then used to compute the final MAC. Each time, the secret key is mapped into a block of 512 bits. The first time, it is XORed with a constant array of 64 copies of the value 0x36. The second time, it is XORed with a constant array of 64 copies of the value 0x5C.

THE HOTP ALGORITHM

With the HMAC construction, the one-time password algorithm can now be implemented. In fact, the HMAC can almost be used as is. The problem with using the MAC itself as the one-time password is that it contains too many bits. The secure token used by Frank does not directly communicate with the server. Rather, it shows a one-time password Frank needs to type in. A 160-bit number requires 48 decimal digits, which is far too long for a human.

OATH has proposed the Hash-based one-time password (HOTP) algorithm. HOTP uses a key (K) and a counter (C). The output of HOTP is a six-digit, one-time password called the HOTP value. It is obtained as follows. First, compute a 160-bit HMAC value using K and C. Store this result in an array of 20 bytes, `hmac`, such that `hmac[0]` contains the 8 leftmost bits of the 160-bit HMAC string and `hmac[19]` contains the 8 rightmost bits. The HOTP value is then computed with a snippet of C code (see Listing 1).

There is now an algorithm that will compute a six-digit code starting from a K value and a C value. HOTP is described in IETF RFC 4226. A typical HOTP implementation would use a 32-bit C and an 80-bit K.

An interesting variant of HOTP, which I will be using in my implementation, is the time-based one-time password (TOTP) algorithm. The TOTP value is computed in the same way as the HOTP value. However, the C is replaced with a timestamp value. Rather than synchronizing a C between the secure token and the server, TOTP simply relies on the time, which is the same for the server and the token. Of course, this requires the secure token to have access to a stable and synchronized time source, but for a watch, this is a requirement that is easily met.

The timestamp value chosen for TOTP is the current Unix time, divided by a factor d. The current Unix time is the number of seconds that have elapsed since midnight January 1, 1970, Coordinated Universal Time. The factor d compensates for small synchronization differences between the server and the token. For example, a value of 30 will enable a 30-s window for each one-time password. The 30-s window also gives a user sufficient time to type in the one-time password before it expires.

IMPLEMENTATION IN THE eZ430-CHRONOS WATCH

I implemented the TOTP algorithm on the eZ430-Chronos watch. This watch contains a CC430F6137 microcontroller, which has 32 KB of flash memory for programs and 4,096 bytes of RAM for data. The watch comes with a set of software applications to demonstrate its capabilities. Software for the watch can be written in C using TI's Code Composer Studio (CCStudio) or in IAR Systems's IAR Embedded Workbench.

The software for the eZ430-Chronos watch is structured as an event-driven system that ties activities performed by software to events such as alarms and button presses. In addition, the overall operation of the watch is driven through several modes, corresponding to a particular function executed on the watch. These modes are driven through a menu system.

Photo 2 shows the watch with its 96-segment liquid crystal

Listing 1—C code used to compute the HTOP value

```
unsigned char offset = hmac[19] & 0xF;
unsigned b3 = (hmac[offset] & 0x7F);
unsigned b2 = (hmac[offset+1] & 0xFF);
unsigned b1 = (hmac[offset+2] & 0xFF);
unsigned b0 = (hmac[offset+3] & 0xFF);
unsigned hotp_code = (b3 << 24) | (b2 << 16) | (b1 << 8) | b0;
unsigned hotp_value = hotp_code % 1000000;
```

HUMANDATA®

FPGA / CPLD Boards from JAPAN

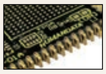
SAVING COST=TIME with readily available FPGA boards

- 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

Arria II GX F572 FPGA board

ACM-025 series NEW

Arria II GX | DDR2

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

RoHS compliant

Cyclone IV E F484 FPGA board

ACM-023 series

Cyclone IV E | MRAM

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

RoHS compliant

CycloneIV GX F484 FPGA board

ACM-024 series

Cyclone IV GX | DDR2 | SIF40

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

RoHS compliant

Cyclone IV E F484 FPGA board

ACM-107 series

Cyclone IV E | MRAM

EP4CE55F23C8N
EP4CE75F23C8N
EP4CE115F23C8N
Compact size (43 x 54 mm)

RoHS compliant

XILINX FPGA Board

Spartan-6 FGG484 FPGA board

XCM-018/018Z series

Spartan-6 | MRAM | DDR2

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

RoHS compliant

Spartan-6 FGG484 FPGA board

XCM-110/110Z series

Spartan-6 | MRAM | DDR2

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

RoHS compliant

Virtex-5 FFG676 FPGA board

XCM-011 series

Virtex-5 | FRAM | SDRAM

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

RoHS compliant

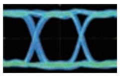
Virtex-5 LXT FFG665 FPGA board

XCM-017 series

Virtex-5 | SDRAM | RocketIO | SIF40

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

RoHS compliant



XILINX PLCC68 Series

XP68-03 Spartan-6 PLCC68 FPGA Module

NEW

Spartan-6 | PLCC68

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

XP68-02 Spartan-3AN PLCC68 FPGA Module

Spartan-3AN | PLCC68

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

ALTERA PLCC68 Series

AP68-04 Cyclone III PLCC68 FPGA Module

NEW

Cyclone III | PLCC68

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

AP68-03 Cyclone III PLCC68 FPGA Module

Cyclone III | PLCC68

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

AP68-02 MAX V PLCC68 CPLD Module

MAX V | PLCC68

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

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

www.hdl.co.jp/CC/

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

display (LCD) and four buttons to control its operation. The left buttons select the mode. The watch has two independent menu systems, one to control the top line of the display and one to control the bottom line. Hence, the overall mode of the watch is determined by a combination of a menu-1 entry and a menu-2 entry.

Listing 2 illustrates the code relevant to the TOTP implementation. When the watch is in TOTP mode, the sx button is tied to the function `set_totp()`. This function initializes the TOTP timestamp value. The function retrieves the current time from the watch and converts it into elapsed seconds using the standard library function `mktime`. Two adjustments are made to the output of `mktime`, on line 11 and line 12. The first factor, `2208988800`, takes into account that the `mktime` in the TI library returns the number of seconds since January 1, 1900, while the TOTP standard sets zero time at January 1, 1970. The second factor, `18000`, takes into account that my watch is set to Eastern Standard Time (EST), while the TOTP standard assumes the UTC time zone—five hours ahead of EST. Finally, on line 14, the number of seconds is divided by 30 to obtain the standard TOTP timestamp. The TOTP timestamp is further updated every 30 s, through the function `tick_totp()`.

The one-time password is calculated by `compute_totp` on line 33. Rather than writing a SHA1-HMAC from scratch, I ported the open-source implementation from Google Authenticator to the TI MSP 430. Lines 39 through 50 show how a six-digit TOTP code is calculated from the 160-bit digest output of the SHA1-HMAC.

The display menu function is `display_totp` on line 52. The function is called when the watch first enters TOTP mode and every second after that. First, the watch will recompute the one-time password code at the start of each 30-s interval. Next, the TOTP code is displayed. The six digits of the TOTP code are more than can be shown on the bottom line of the watch. Therefore, the watch will cycle between showing “totP,” the first three digits of the one-time password, and the next three digits of the one-time password. The transitions each take 1 s, which is sufficient for a user to read all digits.

There is one element missing to display TOTP codes: I did not explain how the unique secret value is loaded into the watch. I use Google Authenticator to generate this secret value and to maintain a copy of it on Google’s servers so that I can use it to log on with TOTP.

LOGGING ONTO GMAIL

Google Authenticator is an implementation of TOTP developed by Google. It provides an

Listing 2—Code relevant to the TOTP implementation

```
void set_totp(u8 line) {
   // this function synchronizes the totp counter
   // to the clock time
   stotp.time2code.tm_sec  = sTime.second;
   stotp.time2code.tm_min  = sTime.minute;
   stotp.time2code.tm_hour = sTime.hour;
   stotp.time2code.tm_mday = sDate.day;
   stotp.time2code.tm_mon  = sDate.month - 1;
   stotp.time2code.tm_year = sDate.year - 1900;
   stotp.code = mktime(&(stotp.time2code))
                 - 2208988800 // adj for unix epoch
                 + 18000;     // adj for EST
   stotp.code = stotp.code / 30;
   stotp.togo = 30;
   stotp.run  = 1;
}
void tick_totp() {
   // this function is called once every second
   // and adjusts the stotp time code every 30 seconds
   if (stotp.run) {
     stotp.togo = stotp.togo - 1;
     if (stotp.togo == 0) {
       stotp.code = stotp.code + 1;
       stotp.togo = 30;
     }
   }
}
u8 val[8];
u8 hash[SHA1_DIGEST_LENGTH];
u8 offset;
void compute_totp() {
   u32 value = stotp.code;
   u8 i;
   for (i = 8; i–; value >>= 8) {
     val[i] = value;
   }
   hmac_sha1(stotp.key,  10,
                 val,  8,
                 hash, SHA1_DIGEST_LENGTH);
   offset = hash[SHA1_DIGEST_LENGTH - 1] & 0xF;
   stotp.totpcode = 0;
   for (i = 0; i < 4; ++i) {
     stotp.totpcode <<= 8;
     stotp.totpcode  |= hash[offset + i];
   }
   stotp.totpcode &= 0x7FFFFFFF;
   stotp.totpcode %= 1000000;
}

void display_totp(u8 line, u8 update) {
   u8 * str;
   u32 n;
   if (stotp.togo == 30) {
     // starting a new interval, recompute totpcode
     compute_totp();
   }
   n = stotp.totpcode;
   if (stotp.run) {
     // Cycle between “totp”, the upper 3 digits of TOTP code,
     // and the lower 3 digits of TOTP code
     switch(stotp.dispseq) {
       case 0: clear_line(LINE2);
          display_chars(LCD_SEG_L2_3_0, (u8 *) “TOTP”, SEG_ON);
          break;
       case 1: clear_line(LINE2);
          str = itoa((n / 1000) % 1000, 3, 0);
          display_chars(LCD_SEG_L2_2_0, str, SEG_ON);
          break;
       case 2: clear_line(LINE2);
          str = itoa((n ) % 1000, 3, 0);
          display_chars(LCD_SEG_L2_2_0, str, SEG_ON);
          break;
     }
     stotp.dispseq = (stotp.dispseq + 1) % 3;
   } else {
     display_chars(LCD_SEG_L2_3_0, (u8 *) “TOTP”, SEG_ON);
   }
}
```

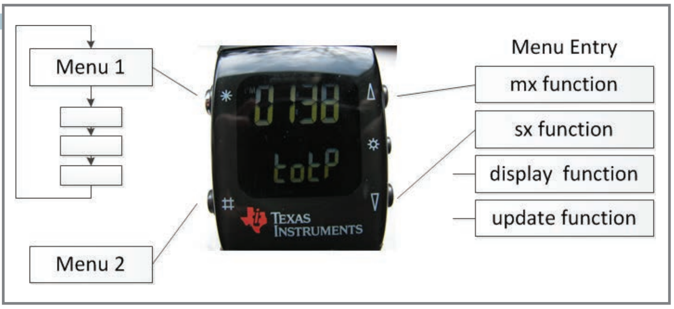


Photo 2—With the watch in TOTP mode, one-time passwords are shown on the second line of the display. In this photo, I am using the one-time password 854410. The watch display cycles through the strings "totP," "854," and "410."

implementation for Android, Blackberry, and IOS so you can use a smartphone as a secure token. In addition, it also enables you to extend your login procedure with a one-time password. You cannot replace your standard password with a one-time password, but you can enable both at the same time. Such a solution is called a two-factor authentication procedure. You need to provide a password and a one-time password to complete the login.

As part of setting up the two-factor authentication with Google (through Account Settings – Using Two-Step Verification), you will receive a secret key. The secret key is presented as a 16-character string made up of a 32-character alphabet. The alphabet consists of the letters A through Z and the digits 2, 3, 4, 5, 6, and 7. This clever choice avoids numbers that can confused with letters (8 and B, for example). The 16-character string thus represents an 80-bit key.

I program this string in the TOTP design for the eZ430-Chronos watch to initialize the secret. In the current implementation, the key is loaded in the function `reset_totp()`.

```
base32_decode((const u8 *)
    "4RGXVQI7YVY4LBPC", stotp.key, 16);
```

Of course, entering the key as a constant string in the firmware is an obvious vulnerability. An attacker who has access to a copy of the firmware also has the secret key used by the TOTP implementation! It's possible to protect or obfuscate the key from the watch firmware, but these techniques are beyond the scope of this article. Once the key is programmed into the watch and the time is correctly set, you can display TOTP codes that help you complete the logon process of Google. Photo 1 shows a demonstration of logging onto Google's two-step verification with a one-time password.

OTHER USES OF TOTP

There are other possibilities for one-time passwords. If you are using Linux as your host PC, you can install the OATH Toolkit, which implements the HOTP and TOTP mechanisms for logon. This toolkit enables you to install authentication modules on your PC that can replace the normal login passwords. This enables you to effectively replace the password you need to remember with a password generated from your watch.

Incidentally, several recent articles—which I have included in the resources section of this article—point to the limits of conventional passwords. New technologies, including one-time passwords and biometrics, provide an interesting alternative. With standards such as those from OATH around the corner, the future may become more secure and user-friendly at the same time.

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

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/262.

RESOURCES

Google Authenticator, http://code.google.com/p/google-authenticator.

Initiative for Open Authentication (OATH), www.openauthentication.org.

Internet Engineering Task Force (IETF), www.ietf.org.

D. M'Raihi, et al, "TOTP: Time-Based One-Time Password Algorithm," IETF RFC 6238, 2011, http://tools.ietf.org/html/rfc6238.

———, "HOTP: An HMAC-Based One-Time Password Algorithm," IETF RFC 4226, 2005, http://tools.ietf.org/html/rfc4226.

OATH Toolkit, www.nongnu.org/oath-toolkit.

K. Schaffer, "Are Password Requirements Too Difficult?" *IEEE Computer Magazine*, 2011, http://dx.doi.org/10.1109/MC.2011.357.

S. Sengupta, "Logging in With a Touch or a Phrase (Anything but a Password)," *New York Times*, 2011, www.nytimes.com/2011/12/24/technology/logging-in-with-a-touch-or-a-phrase-anything-but-a-password.html.

SOURCES

IAR Embedded Workbench
IAR Systems | www.iar.com

eZ430-Chronos Wireless development system and Code Composer Studio (CCStudio) IDE
Texas Instruments, Inc. | www.ti.com

by George Novacek (Canada)

Accelerated Testing

Testing is essential to projecting your product's reliability in the field. This article examines different types of testing, including accelerated life testing, stimulation as accelerated stress testing, and highly accelerated life testing.

Every manufacturer worth his salt is concerned with his product's reliability in the field. This is often expressed as mean time between failures (MTBF) and can also be used to calculate budgetary provisions for warranty, realistic warranty length to be offered to the customer, spare parts allocation, and so forth. Field product reliability should be given to design engineers as a goal and its status should be monitored concurrently with the design process. Unfortunately, with the shortage of resources, reliability prediction is often considered to be something of a useless exercise and, consequently, is calculated after the fact just to satisfy the specification.

STATISTICS & LIES

It frequently becomes an issue when the MTBF calculated after the fact doesn't meet the specification goal. Worse, at times the calculated MTBF is just a few hours less than what is called for in the specification. Panic ensues with engineers tweaking the numbers merely to hit the goal and satisfy some bureaucrat in program management. This only proves the sad truth that, more often than not, no one understands what the reliability prediction is about and that the calculated MTBF is nothing more than a statistical probability based on a component's historical data, its stress levels, and so forth. The former British Prime Minister Disraeli once stated that there are lies, damned lies, and statistics. The calculated MTBF is just a statistic. It indicates the product's reliability ballpark. It is a good start, but not much else.

To arrive at the actual product reliability, the rubber must literally hit the road. As the product enters the service, the manufacturer tracks its performance through field returns. Every failure is analyzed and a corrective action is taken. Discovered "weak" components are replaced with "stronger" ones, or some operating parameters may be adjusted. This process grows the reliability until it settles at a value that reflects balanced design with no weak spots. This is shown in Figure 1 by the blue trace. Unfortunately, this takes a long time, often years, based on the product type and volume.

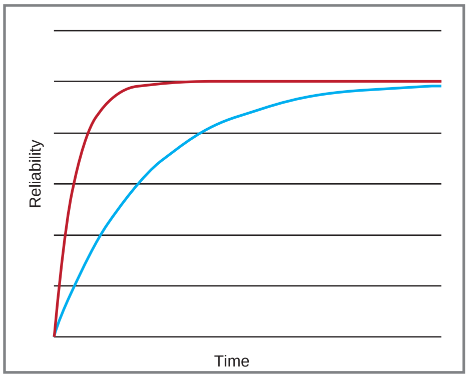


Figure 1—Reliability growth with time. The blue trace indicates the reliability growth based on field data. The red trace indicates the reliability achieved by accelerated testing.

TESTING TO FAIL

Because this process is so slow and costly, as manufacturers must satisfy warranty claims, it is desirable to achieve the final reliability in the shortest period of time so that the product can be released into the field with its reliability already established. This is where accelerated stress testing (AST) comes in. The principle of AST is that applying a high amount of stress to the product for a short period of time should exhibit the same failure mechanisms as it would at lower stress levels for a longer period of time. AST is product-specific and must be tailored for each product.

In short, AST means that the product is tested under increasing stress until it fails. The failure is analyzed, corrected so that future failures of this kind are prevented, and the test continues. In my experience, the most frustrating aspect of AST is ensuring that everyone involved understands that AST is testing to fail. Too often it is viewed as testing to pass, and every failure causes a crisis beginning at the quality assurance (QA) department, moving up the management ladder with the speed of a wildfire. Without failures, AST would be just a waste of time and money. The effect of a well-performed AST is depicted in Figure 1 by the red trace. The optimal reliability is achieved much sooner.

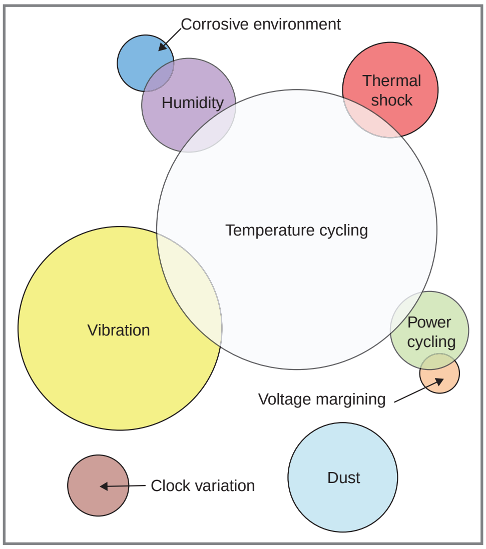


Figure 2—An example of the relationship of precipitated failure population and stresses

SIMULATION & STIMULATION TESTING

Accelerated testing has a long history. Understandably, military and aerospace have always been very interested in receiving products with guaranteed minimum reliability. This gave rise to reliability development growth testing (RDGT) programs. Over the years, reliability growth testing has been refined and implemented by manufacturers of medical, automotive, nuclear, and other durable products.

Fundamentally, there are two types of accelerated tests: Simulation and stimulation. During simulation the product's lifecycle is accelerated, while in the course of stimulation, stresses higher than the product would normally encounter induce fatigue precipitating failure and thus reveal the product's weakness. Simulation testing is known as accelerated life testing (ALT) and stimulation testing is referred to as accelerated stress testing (AST).

A good example of ALT is a computer keyboard. Let's say it is expected to be used for five years. Assuming the operator can type 80 words per minute (350 characters per minute), seven hours a day, 280 days a year, and the highest usage keys are vowels, we can estimate the highest usage keys to see 41.16 million strokes over the five-year period. Subsequently, we can devise a tester with solenoid plungers, striking a key 900 times a minute. Thus the five-year life test can be performed in 31.7 days.

To devise effective ALT, we must understand the product's life expectations, the operating and non-operating environment it will be exposed to over its life, the frequency and duration of the operation in the different environments, the extreme stress levels it will experience, and so forth. For the success of accelerated testing, it is imperative that all failures and conditions under which they occur are documented and analyzed, the failure mechanisms are identified, and proper corrective actions are taken.

The intent of AST is to induce product fatigue by applying higher levels of stress than the product would experience during its life. The concept of AST lies in the assumption that higher stresses applied to a product accelerate the rate at which the failures become evident during normal life. In 1924, A. Palmgren and later in 1945, M. A. Miner independently developed a procedure called the linear cumulative-damage rule. It enables us to calculate the results of the fatigue build up, eventually leading to a failure. Here again, devising AST requires the same amount of understanding of the expected product life as for the development of ALT. It is crucial to determine what kinds of stresses are to be applied to the product for best results. Figure 2 is an example of distribution of failures among different types of stresses. The AST program needs to be designed accordingly. Figure 2 shows us that, for a specific product, vibration and temperature shock plus cycling (shake 'n bake) plus power cycling at the temperature extremes will uncover most of the failures. Do we want to include other stimuli (e.g., humidity, dust, etc.)? It depends on the product. Once again, AST cannot use some generic procedures, but must be specifically designed for each product.

ALT and AST have different purposes. Even when ALT is performed with higher stress levels for time compression, the purpose of ALT is to determine not only whether a failure will occur during the product's expected lifecycle, but also when during the lifecycle it is likely to occur. AST, on the other hand, is based on the assumption that an item will exhibit the same failure in a short time under high stress than it will exhibit in a longer time under lower stress.

STRESS SCREENING

A question may be asked: what is the difference between AST and the environmental stress screening (ESS) I addressed in *Circuit Cellar* 255, 2011? Both methods use stimulation to precipitate latent failures. However, ESS is short and applied to every unit with the purpose of weeding out infant mortality. AST is a part of the development phase to identify design weaknesses and to correct them before releasing the product to manufacturing.

Highly accelerated life testing (HALT) is used to accelerate reliability growth even more during engineering development and thus greatly reduce in-service failures. Here, a progressively more severe stress is applied in discrete steps, building to a level significantly beyond what the product will ever be exposed to in service. Data obtained from HALT can be used for preparation of highly accelerated stress screening (HASS), which is just a different name for ESS.

In this article, I attempted to show another aspect of engineering, often invisible to a budding engineer for whom very little exists beyond the fun of designing a new product. And yet, these seemingly boring peripheral disciplines are just as important as an engineer's ingenuity for bringing good products to market.

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

Automotive Electronics Reliability Handbook, Society of Automotive Engineers, Inc., 1987.

W. Denson, *Handbook of 217Plus Reliability Prediction Models*, Reliability Information Analysis Center (RIAC), 2006.

G. Novacek, "Environmental Stress Screening," *Circuit Cellar* 255, 2011.

Thermotron Industries, "The Fundamentals of Accelerated Stress Testing Handbook," www.thermotron.com/resources/ast-handbook.

by George Martin (USA)

Low-Power and Energy-Harvesting Applications

Low-power and energy-harvesting applications are becoming ubiquitous. This article discusses some emerging technologies and provides tips on how they can be integrated into your designs.

One of the benefits that comes from the high-volume manufacturing of integrated circuits (ICs) is that they result in low-cost solutions. If your requirements are close to any of these high-volume solutions, you can benefit. And the advantages of using the high-volume solution in your embedded system can be great.

Ted Lewis, Professor of Computer Science and National Security Affairs at the Naval Postgraduate School, once wrote an article in *Computer* magazine claiming that there were only four categories of ICs: zero cost, zero delay, zero volume, and zero power. The zero-cost category includes PC, cell phone, car stereo, and other high-volume consumer-type electronic products. The zero-delay category consists of ICs that are used in applications such as supercomputing, where speed is the all-important characteristic. The zero-volume category consists of ICs that are used in applications such as the space shuttle and perhaps fighter planes. These are markets where companies such as IBM must participate. IBM must build the computers on the space shuttle. The zero-power ICs are the battery-operated devices. Lewis argued that the zero-power market is really just part of the zero-cost market. I'm not sure I agree, but let's hold this thought for a moment.

Next, let's think about the ubiquitous devices or solutions. Merriam Webster dictionary defines *ubiquitous* as "existing or being everywhere at the same time: constantly encountered." The mouse is a ubiquitous device. Think about ubiquitous in the embedded world. The USB comes to mind as does the Ethernet. It used to be RS-232, but that's no longer available on new PCs. Carrying this a bit further, as a designer of, say, a file system, you could create your own design or implement an existing file system such as FAT16. In today's fast-moving world, you would need an overwhelming reason to create your own design. Your time-to-market and customer acceptance dictate using the ubiquitous solution if one exists. Keep an eye out for the ubiquitous devices and techniques.

"In today's fast-moving world, you would need an overwhelming reason to create your own design. Your time-to-market and customer acceptance dictate using the ubiquitous solution if one exists. Keep an eye out for the ubiquitous devices and techniques."

Now let's think about low-power applications, extremely low-power applications, such as a wall switch controlling the room lighting that's not connected to any power source or even a transducer in the road that is powered by the vibration of vehicles passing by. There is even a fan for the top of your wood stove that generates electricity from the heat differential coming off the stove (no batteries or external power supply). These are examples of energy harvesting. It's just beginning to evolve and perhaps it seems like a lot of hype. But I think there may be more to it. Energy harvesting extracts energy from the environment. Think of solar cells or a treadmill in which the user generates the energy needed to power the electronics on that treadmill. These are not so far out.

Putting all of those examples together, there is a gathering of forces that is giving rise to zero-powered devices that will be used in our everyday lives. Build a new house and you won't have to wire the light switches on the walls. How much time and money will that save? And if you don't like the switch location you can easily move it. Or sensors in the road measuring traffic and feeding that information to the stop-light control system. This technology is not here yet. But it's going to happen in just a few years.

LOW-POWER ELECTRONICS

To make all these far-out products become real, we will need several components to become real and ubiquitous. First, let's examine the electronics. Take a look at some of the offerings in the low-power CPU area. Texas Instruments has a line of MSP430 ultra-low power 16-bit microcontrollers. These are found in many battery-operated devices. In Sleep mode, these microcontrollers draw about 2 µA of power. Current semiconductor designs have taken this number an order of magnitude lower. You can find standby current for the CPU in the 0.2-µA area. Combine this with super caps or modern batteries. Both storage devices offer long shelf life, low leakage, and good working parameters.

Let's perform some quick calculations. With a 2-µA current draw over a year that's 17,520 mAh, or 17.520 mAh (i.e., 2 × 24 hours a day × 365 days a year). The standard CR3032 coin-cell battery has a 500-mA per hour capacity. So, the sleeping microcontroller would be good for almost 30 years. That's way past the shelf life of the battery. I realize we haven't performed any useful computing or control with the sleeping CPU. My point is to show that low-power electronics are real and currently available. Are the zero-power ICs separating themselves from the zero-cost pack? I think they might be, but we'll see.

SMALL DEVICES

Take a look at Infinite Power Solutions for batteries and ultracapacitors from Maxwell Technologies, just to name a couple. This entire area of energy storage is moving rapidly down the power curve. Infinite Power has a battery that is 0.5″ × 0.5″ × 0.007″ and has a capacity of 0.13 mA per hour at 4 V. That's the size of a postage stamp. And the technology is Lithium-based, so it's reasonably well understood. Devices like these enable energy to be stored in very small volumes (postage-stamp size) and have reasonable overall performance.

ENERGY HARVESTING

Energy-harvesting products, such as solar cells, are not new. I have several calculators that have solar cells as their power source. They harvest the ambient light in the room and work quite well. You might have a flashlight with a magnet and a coil inside. Rotating the flashlight moves the magnet back and forth through the coil and generates enough power to run the light and store the extra power in a capacitor. These examples are real and in the market today.

Imagine a room with Wi-Fi available. What if you had an antenna that received the Wi-Fi signal not for communications purposes but for charging a battery or a capacitor? Not much energy could be harvested in that manner, but as the electronics power consumption is reduced, less is needed to operate. I am currently designing a product that uses this technique. Or imagine a Peletier device that can generate electricity from the inside or outside temperature difference. You can presently buy the devices that consume electricity and either heat or cool. On the horizon are the devices that can generate electricity from the temperature differences. Both of these examples are just coming to the market.

BLUETOOTH & BLE

I'm sure you're familiar with Bluetooth. Most cell phones have it. Bluetooth can be used to implement hands-free cell phone operation. It works very well. I found the initial implementations of Bluetooth complicated and they did not always properly and reliably connect. But today's devices work well. Now they are stable and great for streaming audio. Bluetooth is a wireless standard for communicating using the ISM band (2,400 MHz to 2,480 MHz). The technology defines 79 bands 1-MHz wide and uses frequency hopping to spread the peak energy out over these bands. For this discussion, I need to add that it uses packets as a means of communications.

Now this is where our journey starts to get even more interesting. The industry needs an efficient, ultra-low-power, streamline form of communications. Bluetooth is great for streaming audio, but it consumes too much power and has too much overhead for very-low-power applications. This is where Bluetooth low energy (BLE) comes in.

BLE uses the same ISM band but has 40 bands each 2-MHz wide. Modulation is Gaussian frequency-shift keying with a modulation index of 0.5. This makes for easier detection of devices and ultra-low-power radio designs. Peak data rates are 1 MBps with overall rates of 200 KBps. Packets are in the 8-to-27-byte range. The latency and set up time is in the order of 6 ms down from 100 ms for Bluetooth. Look for radios and chip solutions for Texas Instruments, Panasonic, and Nordic Semiconductor. Great, we all get to buy new development tools.

BLE uses advertising as a means of establishing a connection. Once a device pair is recognized, the data can be sent in small packets. And agreements between devices can be registered for each to go away for up to several seconds and then come back to talk. This approach cuts out the discovery process and keeps the units synchronized. Other communication techniques are in the BLE specification. Remember: BLE is great for short bursts of data.

PUTTING IT ALL TOGETHER

First, let's use some linear thinking. It's clear that light switches, road sensors, and thermostats can be moved into this very low-power, energy-harvesting world. Work needs to be done. But as engineers, that's what we're here for. The applications that use linear thinking are vast and all are ready to bring to market. I would also think that one could follow and perhaps then predict the growth path once computers were connected in a network.

Next, let's try some nonlinear thinking. Could you predict that networking computers would give us Facebook (which recently went public)? That's the problem with the nonlinear process. It's, well, nonlinear. I've tried thinking of examples, but I can't come up with anything worth writing about. It's time to start reading science fiction books looking for good ideas. Actually, you need to spend time with your customer. If you are an in-house engineer, then look to your sales and marketing staff. Inform them of what's coming. If you are an independent engineer, meet with your customer and tell them about what's possible. Get them talking to their customers about the capabilities that are coming.

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.

RESOURCES

Infinite Power Solutions, Inc., www.infinitepowersolutions.com.

Maxwell Technologies, Inc., www.maxwell.com.

Nordic Semiconductor, www.nordicsemi.com.

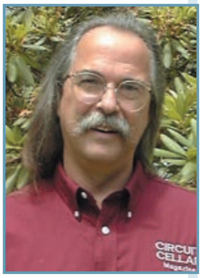
Panasonic Corp., www.panasonic.com.

SOURCES

Bluetooth v4.0
Bluetooth SIG, Inc. | www.bluetooth.com

MSP430 Ultra-low power 16-bit microcontrollers
Texas Instruments, Inc. | ww.ti.com

by Jeff Bachiochi (USA)



SNAP to It (Part 2)

SNAP Engine Versatility

With Synapse SNAP technology, you can easily and effectively run a wireless mesh network. This article details two machine-to-machine projects that highlight the SNAP engine's full potential.

I tend to get excited about things. This seems to have both good and bad side effects. I believe my curious mind helps me to understand our world a little better. Often this requires getting my hands dirty. However, curiosity is what killed the cat, so it's best to use a bit of common sense and stay safe. I have been known to purchase things just to perform autopsies. So I try not to buy on impulse. My personal rule is that I must wait at least 24 hours before I purchase anything. I need to make sure that the burning desire isn't just a momentary rush of adrenalin.

Sometimes the spark doesn't ignite any long-term flame. No harm, no foul. If the feeling passes, I'm ready for the next quake of excitement without needless expense. Sometimes an idea can smolder for a time only to reappear as an inferno, heading in a whole new direction. This trait tends to lead one into becoming a jack-of-all-trades, but a master-of-none. You can't expect much finesse from someone who believes things such as: "There's a hammer for every task." Or, "If it won't move, use WD-40, if it does move, use Duct tape."

Before working with Synapse Wireless's SNAP network operating system (OS), my plan was to use it strictly for its transparent transmission capabilities—that is, use it simply as a wireless serial connection. But I learned it is far more powerful than that. It has sufficient I/O capability and a powerful scripting language that can handle most application tasks without involving another microcontroller in the design. This project is presented as two separate applications because both have merit. Before continuing, I suggest you review my previous article on SNAP ("SNAP to It (Part 1): Mesh Networking Simplified," *Circuit Cellar* 260, 2012). The article provides some groundwork that will prove useful in following these applications.

PROJECT 1: MND

The multinode director (MND) project enables your application to serially connect with a wireless Master node that can communicate with one or more wireless endpoints. Endpoints are serial devices connected to wireless Slave nodes. The host directs the device using ASCII commands. Each node uses a SNAP engine to provide wireless connectivity between your application (master) and any number of serial devices (slaves). The master automatically establishes a connection with every slave device that appears by exchanging SNAP addresses with the device. The master will receive communication from any SNAP device and can send a message to a specifically selected slave device.

Each SNAP engine has a 24-pin footprint that is slightly bigger than 1 square inch. They consist of a microcontroller that is programmed with the SNAP OS and a radio frequency (RF) transceiver. Many of the pins are dedicated to some kind of I/O. For this application, I'll be using four of these pins as TX, RX, CTS, and RTS from onboard UART1. These TTL-level I/Os may be connected directly to the application's TTL serial port if available, which can eliminate the need for Maxim Integrated Products MAX232-type level shifters. Two additional I/Os add an input push button (forcing reconfiguration) and an LED

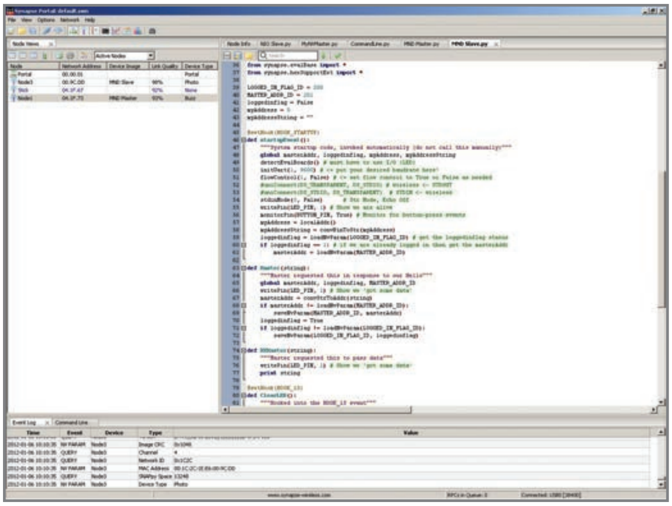


Photo 1—Synapse's Portal application provides a development environment that enables wireless updating of SNAPPy scripts (like this "MND Slave.py" script) to any module within range. From the Portal application, you can also request the execution of any function on any module, which is a great debugging aid.

output to indicate communication activity.

Besides the underlying wireless protocol of the SNAP OS, it also provides the ability to execute user-provided Python-like scripts. Scripts are a combination of "built-in" functions as well as user-defined functions. The user develops the text script and downloads it (wirelessly) into each SNAP engine using the free Portal program. Photo 1 is screenshot of Portal showing the slave script for the MND.

Some "built-in" functions have triggers associated with them (e.g., a timer tick or an I/O changing state). Script execution is event-driven—that is, only the function associated with a triggered event (HOOK) will execute. While other functions may be included in this path, execution ceases at the path end, idly awaiting the next event. Figure 1 depicts how the event functions in the SNAPPy Slave script determine the node's operation.

All scripts execute the HOOK_STARTUP event when power is applied to the module. In this Slave node, you initialize UART1 and connect STDIO (user input and output) to SNAP's UART1 in "String" (CR terminated) mode. Note that if loggedinflag = 1 (the Slave has already made union with a master node), the masterAddr is recalled from NV memory (so you can immediately begin communicating).

All user input coming in the RX input of UART1 is analyzed in the HOOK_STDIN event. If loggedinflag = 1, any data is just sent to the Master node's function "RXSlave," using a "built-in" remote procedure call (RPC) function (wireless transmission requesting execution of the remote function "RXSlave"). Any wireless reception (from the Master) will be of the same RPC form looking to execute a specific function. Here the Slave's function MASTER(string) enables a Slave node to determine the address of the master. The reception of an RPC requesting to execute the function MASTER(string) will only be generated (by the master) when the slave has executed the Hello() function (see the HOOK_1S, described later). The MASTER(string) function creates a union between the slave and its master, converts the string to a address value stored in NV by the slave, and sets the loggedinflag = 1. The RXMASTER(string) function just moves the string of data received to the UART1 RX output.

Two other functions are defined in the Slave script. When any monitored input pin changes state, HOOK_GPIN is executed. I/O0 (BUTTON_PIN) is the only input monitored by this script and the function executes code upon any change to this pin's state. You can differentiate between rising and falling changes by checking the pin's current value. This button is used to cause a slave—which has already formed a union between itself and a master—to break this union, and to look for a master once again, by clearing loggedinflag and resetting the node with the reboot() function.

A number of timer tic HOOKs are available. This script uses the 1-s, HOOK_1S, and performs two functions based on the state of loggedinflag. If loggedinflag = 0, then it sends out a multicast RPC transmission asking all nodes to perform their Hello() function. On the Master node, this function sends out a unicast RPC transmission to the sender (the Slave) asking to perform the MASTER(string) function to establish a union. Once the union has been established and loggedinflag = 1, the HOOK_1S function merely turns the LED OFF. The LED indicates node data activity.

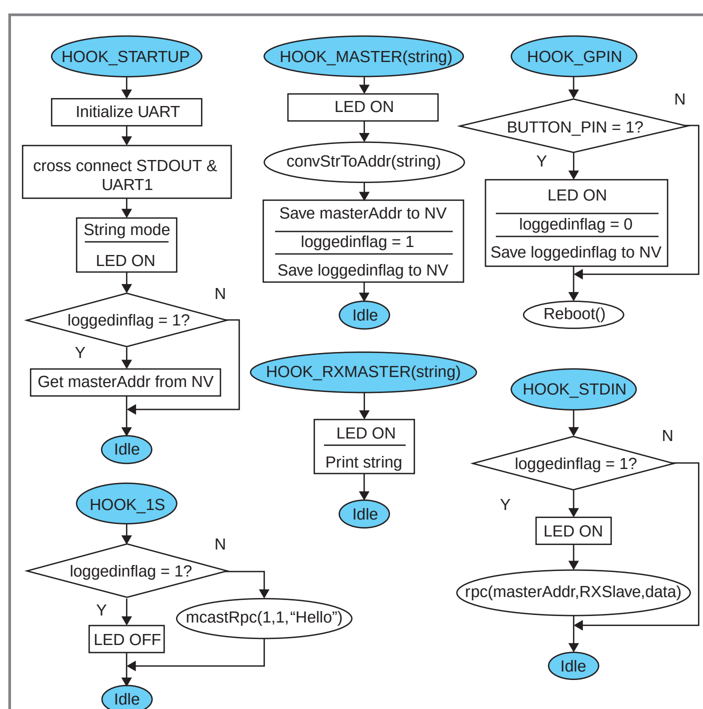


Figure 1—The Slave script for Project 1 uses "built-in" functions that include event-driven triggers (HOOKs), but also user-written functions unique to your application. This script can be downloaded to multiple SNAP engines. SNAP scripts are event-driven. HOOKs are entry points for script functions that should be executed when a specific event occurs. A number of functions are built into to the SNAP OS while other functions can be created by the user. Function scripts can call other functions. An executing function must complete before another function can be executed.

MND MASTER

On the application side of things, you may have many external serial devices offering (wireless) data to your application. Once a slave has created a union with the master, any can send (CR terminated) data to the application. A slave will use an RPC function to request that the RXSlave(string) function be executed. This function in the master's script sends the string "From=xxyyzz" to the UART1 TX pin prior to sending each string received from a slave, so your application can tell where the request came from (see Figure 2). Outgoing messages (to a particular slave) cause a HOOK_STDIN event and need to be routed, so data coming from the application looks for the string "NODE=xxyyzz." This enables the destination address of all outgoing messages to be set to "xxyyzz." Any data that follows the "NODE" string gets transmitted to the Slave address, which was previously set.

While this project enables a Master node to make unions with multiple Slave nodes and carry on personal conversations with each, I'll bet you can see how these two scripts could be combined and simplified when you only want to eliminate the wire between two serial devices. Any node could automatically make contact with another node using the RPC communication. Once a union is established, it could use the Transparent mode to pass all data back and forth. A one-to-one relationship also avoids the need to pollute the datastream with any additional ASCII data.

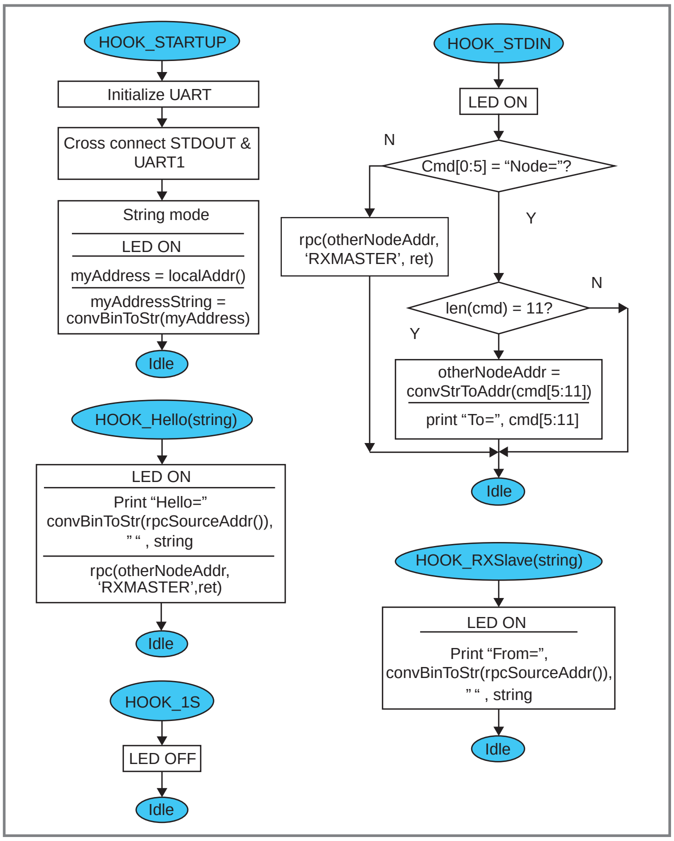


Figure 2—The Master script for Project 1 handles the creation of unions between itself and any number of Slave modules. It also labels incoming data strings with a tag identifying who is sending the data. Data strings destined for individual Slave modules are directed by including destination information in the form of a "Node=xxyyzz" string. This flowchart shows the script functions associated with the Master node. This is similar to the Slave script except for the HOOK_STDIN function, which traps user input to redirect the outgoing data to a particular Slave node (of course, this is only necessary when more than one Slave node is used).

PROJECT 2: WALL-i

The Wall-i project enables the replacement of the standard AC wall switch and outlet (or other directly connected appliance) with a digitally integrated switch and control scheme using the SNAP wireless network. This has multiple benefits. The largest benefit is being able to monitor the switch's use and the appliance's state. Your home wiring scheme comes in two basic forms, which I'll call S-A and A-S. These are both series circuits wired as a loop depending on what is the closest to the AC source, either the S(witch) or the A(ppliance). This is important to understand if you will be installing this project in your home, as it has to do with where AC (as a source) is available.

Most wall switches used around the house come in three styles: push button, slider, or toggle switch. These might directly control a wired light or appliance or indirectly control any device via an AC outlet. The wiring circuits used today mechanically interrupt the flow of current directly using a switching device capable of handling the currents necessary for the devices they handle. Most wall switches are rated for at least 15 A.

Low-voltage circuitry with less expensive (low-voltage/current) switches can be used as inputs. The control signal output can also come from the same circuitry. I'll admit that using a TRIAC or relay to control an AC appliance costs a bit more than a standard wall switch, but I think the benefits outweigh the costs. For this project I'll assume that AC is already running everywhere, so this shouldn't require rewiring the house. But, if you are starting from scratch or just looking to save money, it would make sense to run a DC source throughout the house since adding a power supply to every piece of circuitry is expensive (and bulky).

I've been tossing around a project like this for a while now but it wasn't until I experienced the power of the SYNAPSE modules that it all began to gel. I started by defining the I/O on the basic SNAP engine into three groups: digital inputs, digital outputs, and analog inputs. Most of this project is based around the digital I/O and, as you will see, Synapse modules handle both digital and analog types of control. Four input values IN1:4 are created using pairs of digital inputs. Table 1 shows that there are four possible modes of operation for each set of inputs.

It would seem that digital outputs would be cut and dried, but this is not the case. Each output has a value associated with it and this value determines the state of the output. If the value is greater than 63, then the output is a steady state based on bit 7, where 1=ON and 0= OFF. Otherwise, the output is a pulse from 1 to 64 ms. In a perfect product, this would be a pulse-width modulation (PWM) output with a frequency of 120 Hz based on and in sync with the zero crossings of the AC line. However,

Physical input	Mode	Function	Value
1A	Momentary push button	Push ON	192
1B		Push OFF	128
1A	Momentary push button toggle	Push ON/OFF	192/128
1B		Not used	
1A	Slider	Increment	0–63
1B		Decrement	0–63
1A	ON/OFF switch	ON/OFF	192/128
1B		Not used	

Table 1—Each pair of physical inputs can be configured to produce an input value in one of four ways. The first three use push buttons while the last can be a normal ON/OFF wall switch. You can tell from the values whether the input is digital (bits 7:6) or analog (percent, bits 5:0).

there is only one PWM available. I didn't want to lose an input, and servicing the zero-crossing event wouldn't be predictable. So this (percent) pulse output can be measured by an external circuit (perhaps a six-pin microcontroller), which will sync its PWM output to the line if a dimmable AC output is required.

Project 1 demonstrated a master handling multiple slaves. I didn't want some system master playing an integral role in what happens every time an input changes somewhere in the house. I wanted to use the SNAP engine's full potential for handling its own destiny. While a master can serve as a configurator, a monitor, and a gateway for remote control of any node's I/O, all the nodes should be able to converse with one another as necessary without someone else calling all the shots.

DYNAMIC LINKING

To increase flexibility between cause (Input event) and result (Output event) the connection must be configurable. When the output that is affected by an input event is local—that is, within the node itself—no communication is necessary. However, if the output is on a different node, then a command must be sent to the other node. The address of each (SNAP engine) node is pre-programmed (in NV) and consists of the last three hexadecimal digits of a node's MAC (printed on its label). To ease the configuration process for a node I added a pseudo-master node. This is necessary only for configuration of each node and not in normal operation. It can, however, serve as a link into the network to provide the "i" in Wall-i (Internet connectivity).

Each node has 12 global variables that reflect the current state of their associated functions, IN4:1, OUT4:1, and AIN4:1. While these values are established by each node, they can be read and written by any node. Each node has additional configuration variables that are initialized from NV (see Table 2).

These configuration variables determine what happens when a change of state (COS) occurs on any of the inputs. A COS always affects an Input value. If one or more internal outputs are enabled via DICx for INPUTx, the Input value is written to the Output value for that output. This links INPUTx to OUTPUTx within the same node and does not require any network communication. If one or more external outputs are enabled via DECx for INPUTx, the Input value is written to the Output value for that output. This requires that a command is sent to DNAx. DNAx is the node address of the external node INPUTx wishes to control. The output (or outputs) to be written to are determined by the DECx. The command is an ASCII string sent to DNAx via rpc(DNAx, 'RXMASTER', 'OUTx=v'), where x = which output and v = the value from INPUTx.

Anytime an input or output changes state, an additional transmission is made to the Master node (whether it's active or not). This gives any application using the master the ability to perform some action based on activity. This might be simply logging all activity, helping to debug some inappropriate action, or linking an Internet server to the network.

CONFIGURATION

Giving a Master node access to the network not only provides it with the capability to monitor and control any node, but also to configure each node's dynamic-linking abilities. Photo 2 is a screenshot of the configuration utility written to aid in configuring the whole network of nodes. When a node is powered up, if it hasn't been previously linked with the master it sends out its "Hello" message. The master creates the union (like in the previous project) and all is well. The configuration application adds this node's address to its list of nodes (if it is not already there). This list of nodes provides all the legal DNAx node addresses available for configuration purposes. This list can be saved/read as an editable text file should you want to add, delete, or change any entry. Along with each node's address, the list includes text references "labels" that are associated with all I/O for the node, IN4:1, OUT4:1, and AIN4:1. This gives you an opportunity to

Variable name	Function	NV location
IC	Input configuration	201
ASEN	Analog sampling enable	202
DIC1	Dynamic internal configuration for Input 1	203
DIC2	Dynamic internal configuration for Input 2	204
DIC3	Dynamic internal configuration for Input 3	205
DIC4	Dynamic internal configuration for Input 4	206
DEC1	Dynamic external configuration for Input 1	207
DEC2	Dynamic external configuration for Input 2	208
DEC3	Dynamic external configuration for Input 3	209
DEC4	Dynamic external configuration for Input 1	210
DNA1	Dynamic Node address for Input 1	211
DNA2	Dynamic Node address for Input 2	212
DNA3	Dynamic Node address for Input 3	213
DNA4	Dynamic Node address for Input 4	214

Table 2—These NV stored configurations enable each node input to be configured in one of four modes (see Table 1), via IC, affect internal outputs, via DICx, affect external outputs, via DECx of node DNAx. The ASEN configuration enables and disables the sampling of four analog inputs. See the text for more information.

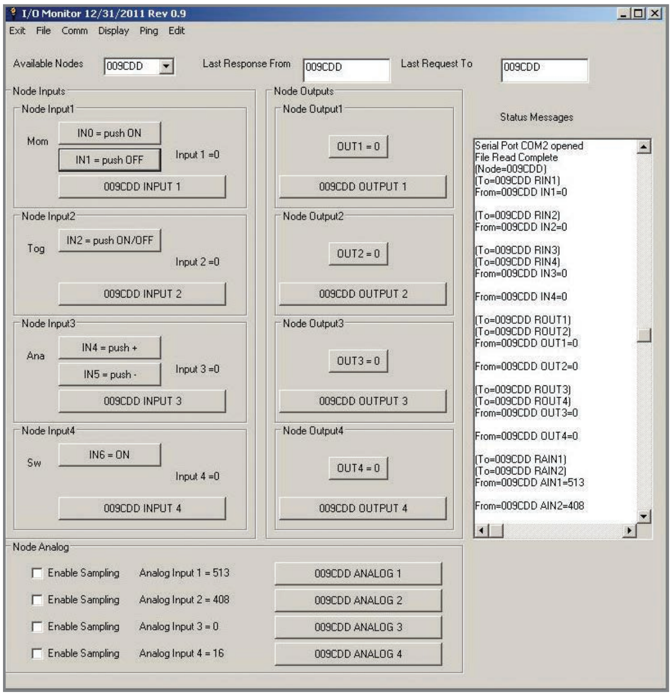


Photo 2—After the SNAPPy scripts were developed for Wall-i nodes and the configuration (Master) node, an application was needed to configure each Wall-i node added to the system. This PC application is written in Shoptalk Systems's Liberty BASIC and communicates with the Master node to set up each Wall-i node. This display screen shows real-time data from any selected Wall-i node. While the master isn't required for normal operations—that is, node-to-node communications—it can be used as the network interface to applications that might enable you to control any node via your smartphone.

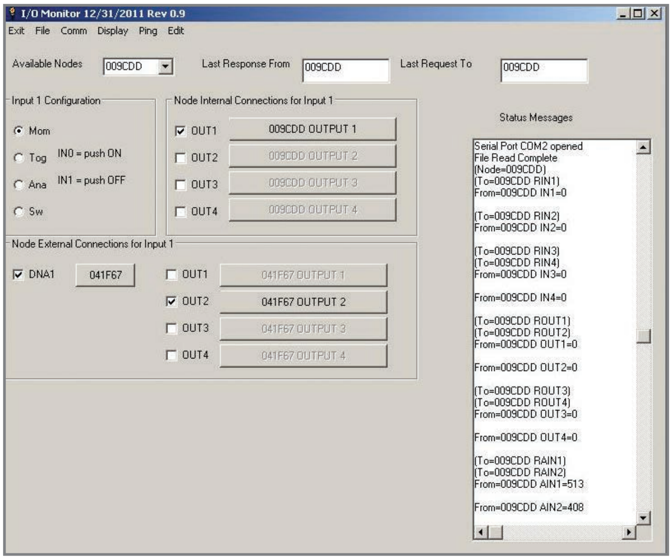


Photo 3—This screenshot shows the configuration possibilities of one input of a selected Wall-i node. Any input operation can be linked to any or all outputs either internally (on the same node) or externally (in another node).

identify each I/O with a description that might make more sense than "IN1" of address "011CDD." Perhaps "Switch @ top cellar stairs" or "Back Porch Light" would be more appropriate.

When a node is selected, the application requests all node variables from that node to identify the present state of the node. The datascreen shows the configuration mode of each input as well as the present IN4:1, OUT4:1, and AIN4:1 values. Each I/O is also labeled with its associated description. Four additional screens are available to configure the dynamic links for each INPUT (see Photo 3). The DECx, DICx, and DNAx variables are updated as changes are made. When the MASTER receives verification of a change, the screen is updated to reflect that change.

A large window on the right side of the application lists all of the communication both into and out of the network via the Master node. "To=" and "From=" strings are appended so the direction of the command can be identified.

Photo 4—Polycase has some great enclosures that include AC prongs and AC socket cutouts. The PM2425-style enclosure offers up to two controllable circuits that plug into an existing outlet.

CIRCUITRY

As I already mentioned, the SNAP engine's footprint is relatively small. The basic circuit required to use the module is also simple. Output circuitry, including a TRIAC or relay, can quickly eat up real estate. If four of these output circuits, including an AC-DC power supply, were to be combined onto a single printed circuit board (PCB) it would be prohibitively large. Retrofitting this into the existing available area, which might be as small as a single gang electrical box, would require a big hammer, with disastrous results. The AC-DC power supply alone might fill up the entire space, due to transformer size (minimum requirement approximately 1.5 VA).

So far, I've built two variations. The first combines a SNAP engine and a power supply using two push-button inputs and two TRIAC outputs fit into a Polycase PM2425-style case shown in Photo 4. The second is similar but has only a single input and TRIAC output. This circuit fits into a single gang electrical box using the Serpac Electronic Enclosures G10A-style enclosure shown in Photo 5.

You may remember that I mentioned true analog inputs as opposed to two push-button inputs configured as UP/DOWN push buttons. Analog inputs were again mentioned in the configuration section (as seen in Photo 3). These inputs are included to enable current monitoring of each appliance that is controlled by the outputs. While current measuring has not been implemented into the present prototypes, I consider this a "must have" and

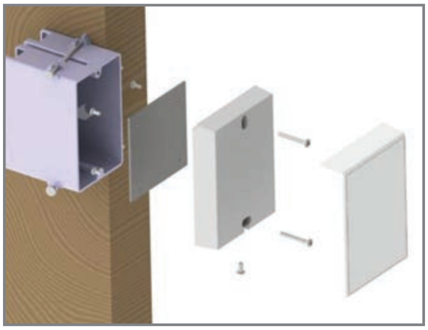


Photo 5—Another great product that can be used with single-, double-, and triple-ganged electrical boxes is Serpac's G10/20/30-style package, which replaces existing cover plates and has internal PCB mounting studs.

will continue developing the entire idea.

Isn't it time we rethink the way we wire our homes? Many appliances today already run on a lower-voltage DC. And DC-powered appliances are more suited for alternative energy storage systems—banks of batteries. When the power goes out, what do you have in place to get you through? A gas-powered generator? But what if gas is unavailable? That was the case in late October 2011 after a major snowstorm crippled the northwestern United States. Gas stations without generators couldn't pump gas. Many eyes were opened at the lack of readiness for such a disaster. Tree limb damage to the local grid was unimaginable, as were the road blockages. This led to paralyzation lasting weeks. Three months later, I still saw damaged trees ominously primed to unleash themselves across local roadways at some unforeseen moment. While this doesn't come close to global disasters like earthquakes and tsunamis, it does force one to stop and think about seeing the light, before the power is restored. As always, I'd be happy to hear your thoughts and ideas about this project or any engineering issues.

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

MAX232 Driver/receiver
Maxim Integrated Products, Inc. | www.maxim-ic.com

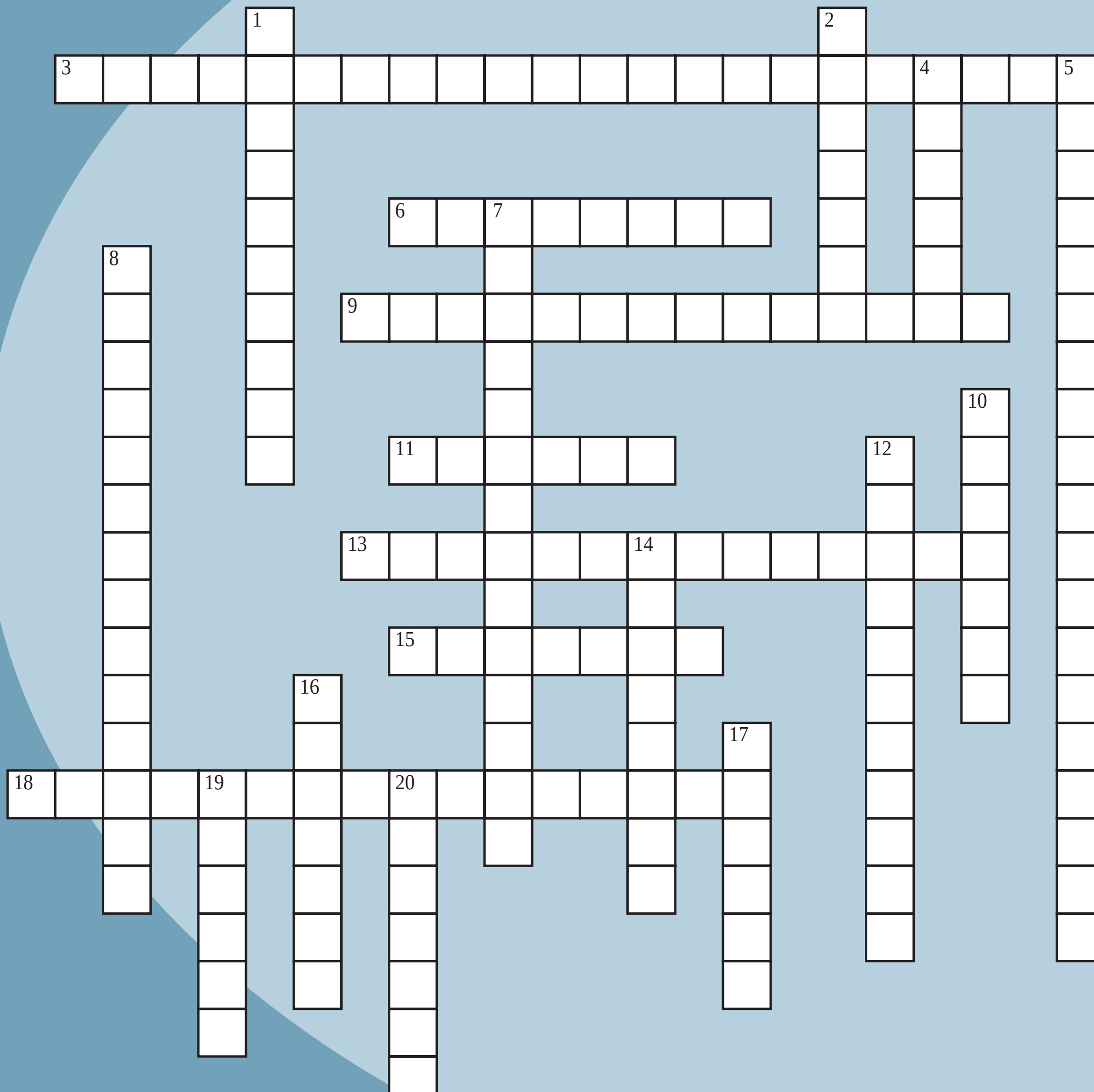
PM2425 PS Series AC wall plug enclosures
Polycase | www.polycase.com

G10A Wall-mount plastic enclosure
Serpac Electronic Enclosures | www.serpac.com

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

SNAP Network operating system and Portal wireless application development environment
Synapse Wireless, Inc. | www.synapse-wireless.com

CROSSWORD



Across

3. Isn't a digital 'scope [three words]
6. A super-efficient band of frequencies
9. Relies on nodes to capture, distribute, and reproduce data [two words]
11. An open-source programming language that relies on automatic memory management
13. A formula for linear electrical networks [two words]
15. Variable resistors, variable capacitors, or trimmable inductors, for example
18. Measures data speed [three words]

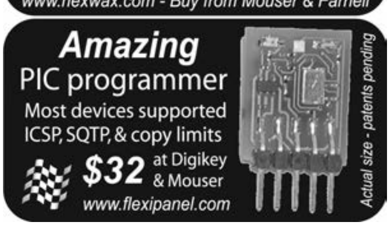
Down

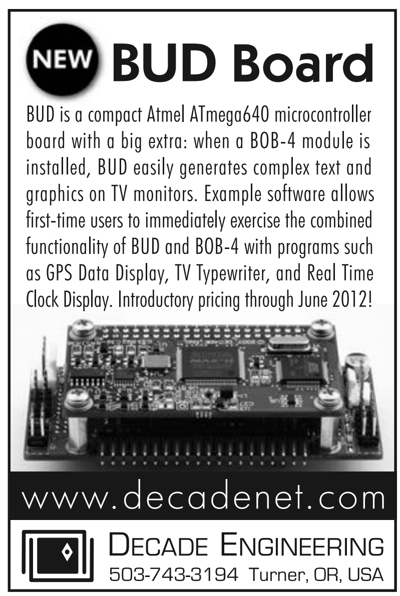
1. Electrostatic physics principle [two words]
2. Wrote 26 feature articles for *Circuit Cellar* between 1999 and 2004
4. Enables you to "get that Linux feeling on Windows"
5. Light emission caused by electrical influences
7. A comparator with two different threshold voltage levels [two words]
8. Reverses an output signal's absolute phase [two words]
10. Equivalent of amperes per volt
12. Sound wave phase
14. Represents a signal's maximum rate of change [two words]
16. Measures the damping of resonator modes
17. PCB file format
19. An amp device with three electrodes
20. Known as the "father of electric traction"

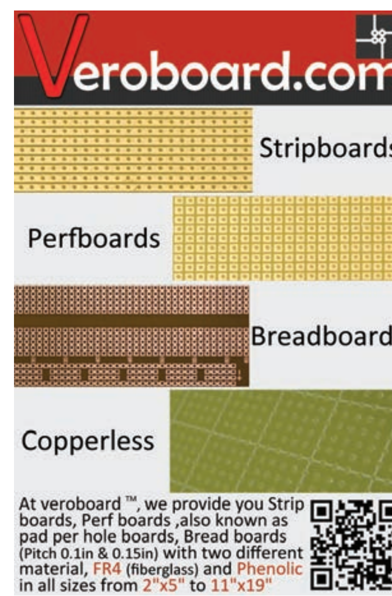
The answers will be available in the next issue.

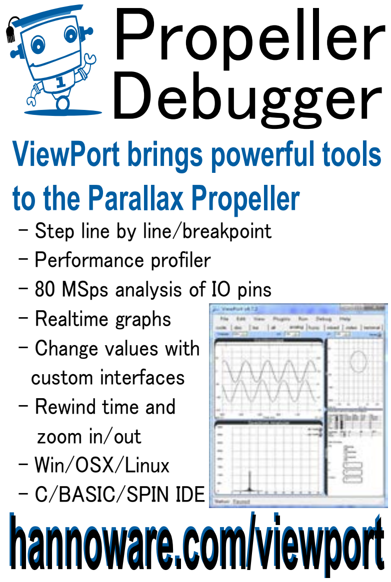
IDEA BOX

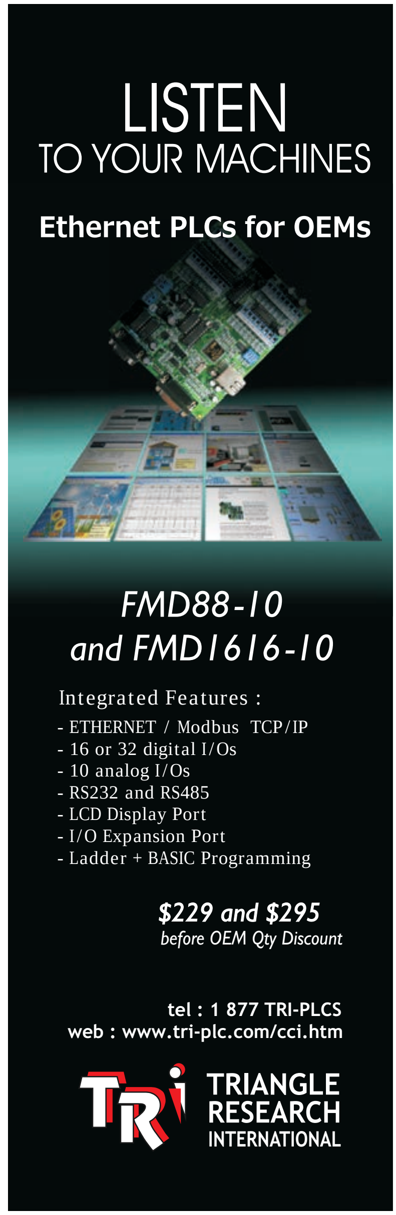
THE DIRECTORY OF PRODUCTS AND SERVICES

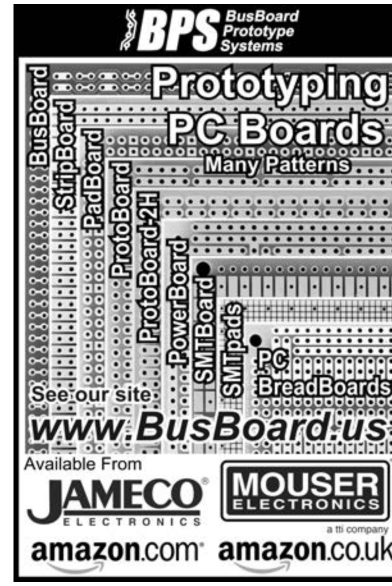




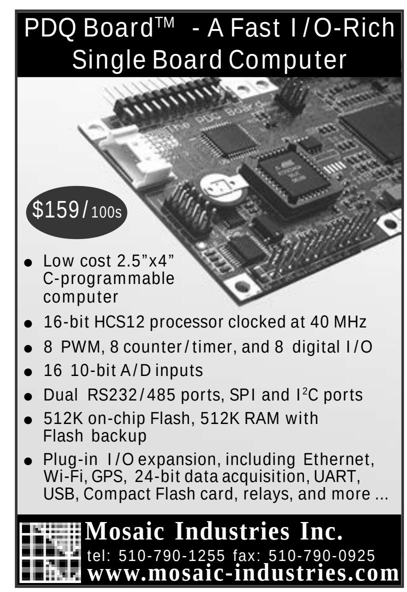


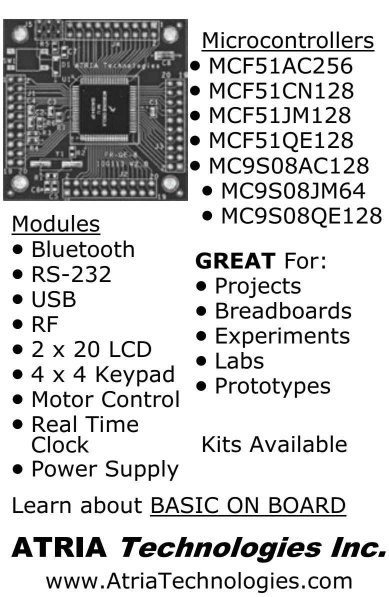

Propeller Debugger
ViewPort brings powerful tools to the Parallax Propeller
– Step line by line/breakpoint
– Performance profiler
– 80 MSps analysis of IO pins
– Realtime graphs
– Change values with custom interfaces
– Rewind time and zoom in/out
– Win/OSX/Linux
– C/BASIC/SPIN IDE
hannoware.com/viewport

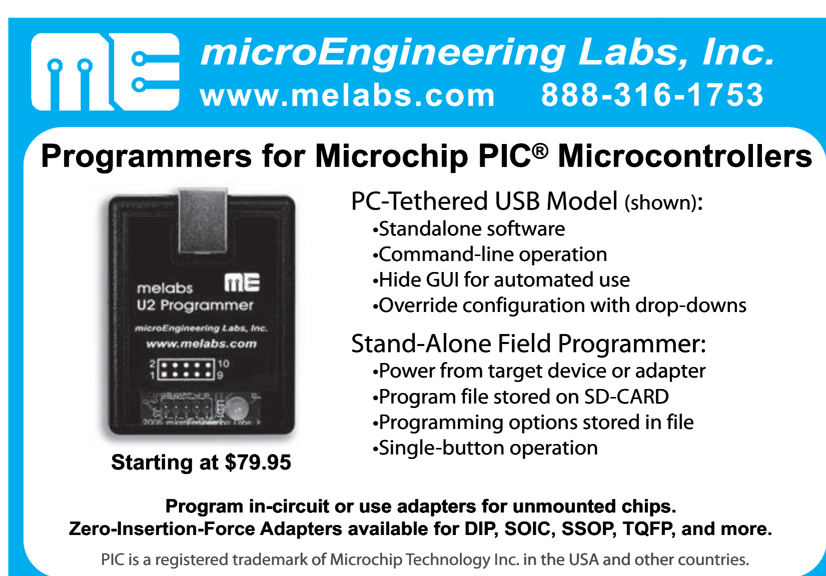

LISTEN
TO YOUR MACHINES
Ethernet PLCs for OEMs
FMD88-10
and FMD1616-10
Integrated Features :
- ETHERNET / Modbus TCP/IP
- 16 or 32 digital I/Os
- 10 analog I/Os
- RS232 and RS485
- LCD Display Port
- I/O Expansion Port
- Ladder + BASIC Programming
$229 and $295
before OEM Qty Discount
tel : 1 877 TRI-PLCS
web : www.tri-plc.com/cci.htm
TRIANGLE RESEARCH INTERNATIONAL


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


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


Microcontrollers
• MCF51AC256
• MCF51CN128
• MCF51JM128
• MCF51QE128
• MC9S08AC128
• MC9S08JM64
• MC9S08QE128
Modules
• Bluetooth
• RS-232
• USB
• RF
• 2 x 20 LCD
• 4 x 4 Keypad
• Motor Control
• Real Time Clock
• Power Supply
GREAT For:
• Projects
• Breadboards
• Experiments
• Labs
• Prototypes
Kits Available
Learn about BASIC ON BOARD
ATRIA Technologies Inc.
www.AtriaTechnologies.com


microEngineering Labs, Inc.
www.melabs.com 888-316-1753
Programmers for Microchip PIC® Microcontrollers
melabs U2 Programmer
Starting at $79.95
PC-Tethered USB Model (shown):
•Standalone software
•Command-line operation
•Hide GUI for automated use
•Override configuration with drop-downs
Stand-Alone Field Programmer:
•Power from target device or adapter
•Program file stored on SD-CARD
•Programming options stored in file
•Single-button operation
Program in-circuit or use adapters for unmounted chips.
Zero-Insertion-Force Adapters available for DIP, SOIC, SSOP, TQFP, and more.
PIC is a registered trademark of Microchip Technology Inc. in the USA and other countries.

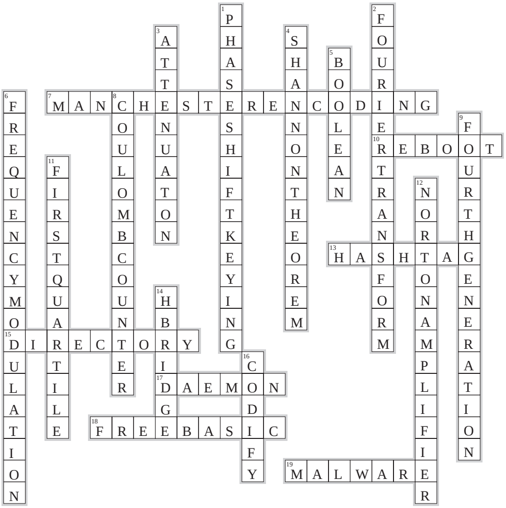
CROSSWORD ANSWERS from Issue 261

Across

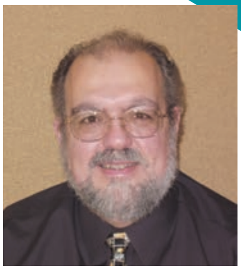
7. **MANCHESTERENCODING**—A form of BPSK [two words]
10. **REBOOT**—Restart
13. **HASHTAG**—The # label
15. **DIRECTORY**—File index
17. **DAEMON**—Takes place behind the scenes
18. **FREEBASIC**—Open-source programming language compiler
19. **MALWARE**—Bad code

Down

1. **PHASESHIFTKEYING**—Transports data by altering a reference signal's phase [three words]
2. **FOURIERTRANSFORM**—Changes a signal from the time domain to the frequency domain [two words]
3. **ATTENUATON**—Used to measure signal loss in dB
4. **SHANNONTHEOREM**—AKA, the noisy-channel coding theorem [two words]
5. **BOOLEAN**—Logic system named after George Boole
6. **FREQUENCYMODULATION**—Opposite of AM [two words]
8. **COULOMBCOUNTER**—Measures battery current [two words]
9. **FOURTHGENERATION**—4G [two words]
11. **FIRSTQUARTILE**—25th percentile [two words]
12. **NORTONAMPLIFIER**—Converts a current to a voltage. [two words]
14. **HBRIDGE**—Four switching components with the load in center
16. **CODIFY**—A way to organize



PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

Power to the People

While last month I may have implied that 8 bits is enough to control the world, there are significant things happening in high-end, 32-bit embedded processors that might really produce that inevitability. There are quite a few new system-on-chip-based, low-cost, single-board computers (SBCs) specifically designed to compete with or augment the smartphone and pad computer market. These and other full-feature budget SBCs are something you should definitely keep on your radar.

These devices typically have a high-end, 32-bit processor, such as ARM Cortex-A8, running 400 MHz to 1,000 MHz, coupled with a GPU core (and sometimes a separate DSP core) along with 128 MB to 512 MB of DDR SDRAM. These boards typically boot a full-up desktop operating system (OS)—such as Linux or Android (and soon Windows 8)—and often contain enough graphics horsepower for full-frame rate HD video and gaming.

Texas Instruments made a significant splash a few years ago with the introduction of the BeagleBoard SBC (beagle board.org, $149 at the time) with their OMAP3530 chip along with 256-MB of flash memory and 128 MB of SDRAM running Angstrom Linux on a high-resolution HDMI monitor. That board has since been superseded by the BeagleBoard-xM (1,000 MHz and 512 MB) at the same price and supplemented by the BeagleBone board. Selling for just $89, BeagleBone includes a 600-MHz AM3517 processor, 256-MB SDRAM, a 2-GB microSD card, and Ethernet (something the original BeagleBoard lacked).

All of the software for these boards is open source, and a significant community of developers has grown up around them. In particular, a lot of effort has been put into software infrastructure, with a number of OSes now ported to many of these boards, along with languages (both compiled and interpreted) and application frameworks, such as XBMC for multi-media and home-theater applications.

Another SBC that has been generating a lot of buzz lately is the Raspberry Pi board (raspberrypi.org), mainly because the "B" version is priced at just $35. Raspberry Pi is based on a Broadcom chip, which is unexpected. Broadcom traditionally only gave hardware documentation and software drivers to major customers, like set-top box manufacturers, not to an open-source marketplace. Apparently, the only proprietary piece of software for the Raspberry Pi board will be the driver/firmware for the GPU core. Unfortunately, as I write this, there are a few lingering manufacturing issues, and Raspberry Pi still awaits shipping.

Both the concept and size of an "SBC" are evolving as well. In addition to the bare development boards, a number of interesting second-level products based on these chips has begun to appear. Take a look at designsomething.org. A couple of projects in particular are Pandora's Pandora Handheld and Always Innovating's HDMI Dongle. The former is a pocket-sized computer that flips open to reveal an 800 × 480 touchscreen and an alphanumeric keypad with gaming controls. Besides the obvious applications as a video viewer, gaming platform, and "super PDA," I see huge opportunities for this box as a user interface for things like USB-based test instruments.

The Always Innovating HDMI Dongle is amazing for how much functionality they've crammed into a small package: it's no bigger than a USB thumb drive (it also needs a USB socket for power), but it can turn any TV with an HDMI input jack and USB socket into a fully functional, Android-based computer with 1080p HD video playback, games, and Wi-Fi-based Internet access. These dongles might easily become distributed home theater nodes, delivering high-quality video and audio to multiple rooms from a common file server; or, one of the other low-cost SBCs might become the brain of a robot that can see and understand the world around it using open-source computer vision (OpenCV).

While it makes an old hardware guy like me feel less useful, it's clear that the hardware—or, more specifically, the necessity to always design unique hardware—is no longer the bottleneck when it comes to powerful embedded applications. In a turn-around from decades ago, the ball is now clearly in the court of the software developers.

The applications for these boards and "thumb-thingies" are endless. Basically, they have the hardware muscle to handle anything that a smartphone or pad computer can do for much less. A lot of work has already been done on the OS and middleware layers. We just need to dive in and create the applications! Then it basically becomes a *simple matter of programming.* Of course, you know how much I personally look forward to that.

steve.ciarcia@circuitcellar.com


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